SliTaz Documentation

SliTaz docs contributors

Mar 24, 2020

Contents

author pankso, oui, linea, jozee, gokhlayeh, ceel

This site provides SliTaz GNU/Linux official and community documentation.

We believe that SliTaz users all over the world want to share their knowledge and experiences through writing. This is an open wiki that anybody can contribute to. We are very grateful to those who want to lend a hand to make SliTaz even easier to use. Welcome!

Newsletter

- *Monthly Newsletter* (page 3) — The SliTaz Monthly Newsletter, news, tips and tricks.

Guides

- *Guides* (page 79) — Small and concise guides from the community.
- *References* (page 247) — Any interesting linux article/how-to/material you may want to share.
- *Development Notes* (page 249) — Information about the development version (cooking).
- *Forum posts* (page 271) — Supplementary documentation from the user forums.

Books

- *Handbook* (page 273) — SliTaz Handbook. LiveCD usage, package management, network or system administration and specific instructions. This Handbook is a community effort to provide high quality documentation for SliTaz users. It will help you get started with SliTaz GNU/Linux and show you how to configure the system to your own needs and preferences. This is the documentation that we advise you to read, learn and consult first.
- *Cookbook* (page 369) — SliTaz Cookbook. Information regarding the management, operation and development of the distribution. Instructions on how to use the wok and package receipts, descriptions of boot scripts and rootcd files, and various tools.
- *Scratchbook* (page 397) — SliTaz Scratchbook. Describes the stages of creating the very first SliTaz distribution commencing with instructions on compiling the Linux kernel, installing the graphical server (Xvesa) and GTK applications, etc. It contains techniques requiring time and motivation that enable you to build a GNU/Linux system from source.

Manuals

All SliTaz related manuals are installed on each SliTaz distribution and can also be downloaded via the Mercurial web interface.

- *Tazinst Manual* (page 449) — SliTaz installer.
- *TazPkg Manual* (page 456) — SliTaz Package Manager.
- *Tazlito Manual* (page 471) — SliTaz LiveCD utility.
- *TazUSB Manual* (page 478) — SliTaz LiveUSB utility.
- *Cookutils Documentation* (page 480) — SliTaz Package Cooker.
- *Burnbox Manual* (page 486) — SliTaz utility to burn CD/DVDs.

Releases Errata

CHAPTER 1

Monthly Newsletter

author linea, draplater, seacat, godane

1.1 Issue 14

author linea, godane

- Written on 16th January 2010

Latest News

- SliTaz featured in Distrowatch review[1]
- SliTaz snow[2] isos released
- We will have a stand at the forthcoming http://www.solutionslinux.fr/[3]
- Newsletter is back again!

New Packages

- calcurse
- tty-clock
- nethogs
- icmptx
- plotdrop
- xournal
- apr-dpd-*
- openvas-*

[1] http://distrowatch.com/weekly.php?issue=20100111#feature
[2] https://web.archive.org/web/20100308130348/http://mirror.slitaz.org/iso/cooking-snow/README.html
[3] https://web.archive.org/web/20100211215018/http://www.solutionslinux.fr/

- snort
- lives
- gkll
- gutenpy
- python-pyprotocols
- aspell-{hu,cs,id,it,ru,sl}
- evince
- libmagic
- libical
- libmpdclient
- libunique
- libplayer
- libvalhalla
- elementary
- enna
- libffd

Updated Packages

- mpg123 ⇒ 1.10.0
- openbox ⇒ 3.4.9
- cairo ⇒ 1.8.8
- firefox ⇒ 3.5.7
- web-applications ⇒ 1.2
- postgresql ⇒ 8.3.9
- apr ⇒ 1.3.9
- apache ⇒ 2.2.14
- sarg ⇒ 2.2.6
- squid ⇒ 3.0STABLE21
- putty ⇒ 0.60-2009-09-08
- sudo ⇒ 1.7.2p2
- mc ⇒ 4.7.01
- perl-pango ⇒ 1.221
- python-pygments ⇒ 1.1.1
- python-bpython ⇒ 0.9.5.2
- python-mysql ⇒ 1.2.3.c1
- python-django ⇒ 1.1.1
- python-sqlalchemy ⇒ 0.5.6
- python-formalchemy ⇒ 1.3.1
- python-jinja2 ⇒ 2.2.1
- tuxtype ⇒ 1.8.0
- intltool ⇒ 0.40.6
- babl ⇒ 0.1.0
- gegl ⇒ 0.1.0
- gimp ⇒ 2.6.8
- couchdb ⇒ 0.10.1
- python-couchdbkit ⇒ 0.3.1
- libsmpeg ⇒ 389
- stoq ⇒ 0.9.11
- stoqdrivers ⇒ 0.9.8.2
- fpm2 ⇒ 0.76.1

- awstats ⇒ 6.95
- pure-ftpd ⇒ 1.0.27
- strace ⇒ 4.5.19
- pygtk ⇒ 2.16.0
- pycairo ⇒ 1.8.2
- pygobject ⇒ 2.20.0
- gdb ⇒ 7.0.1
- reiserfprogs ⇒ 3.6.21
- gajim ⇒ 0.13.1
- midori ⇒ 0.2.2
- ntfs-3g ⇒ 2009.11.14
- libboost ⇒ 1.41.0
- wormux ⇒ 0.8.5
- vte ⇒ 0.23.1
- sakura ⇒ 2.3.6
- jfsutils ⇒ 1.1.14
- emelfm2 ⇒ 0.7.1
- leafpad ⇒ 0.8.17
- popt ⇒ 1.15
- libmodplug ⇒ 0.8.7
- espeak ⇒ 1.42.04
- libffi ⇒ 3.0.9
- udev ⇒ 150
- cyrus-sasl ⇒ 2.1.23
- rsync ⇒ 3.0.7
- mpd ⇒ 0.15.7
- mpc ⇒ 0.19
- ario ⇒ 1.4.2
- slitaz-bootscripts ⇒ 3.2
- eet ⇒ 1.2.3
- e17 ⇒ 2009.12.02
- enlightenment ⇒ 2009.12.02
- pidgin ⇒ 2.6.5
- dnsmasq ⇒ 2.51
- get-virtualbox ⇒ 1.01
- bind ⇒ 9.6.1-P2

Improvements

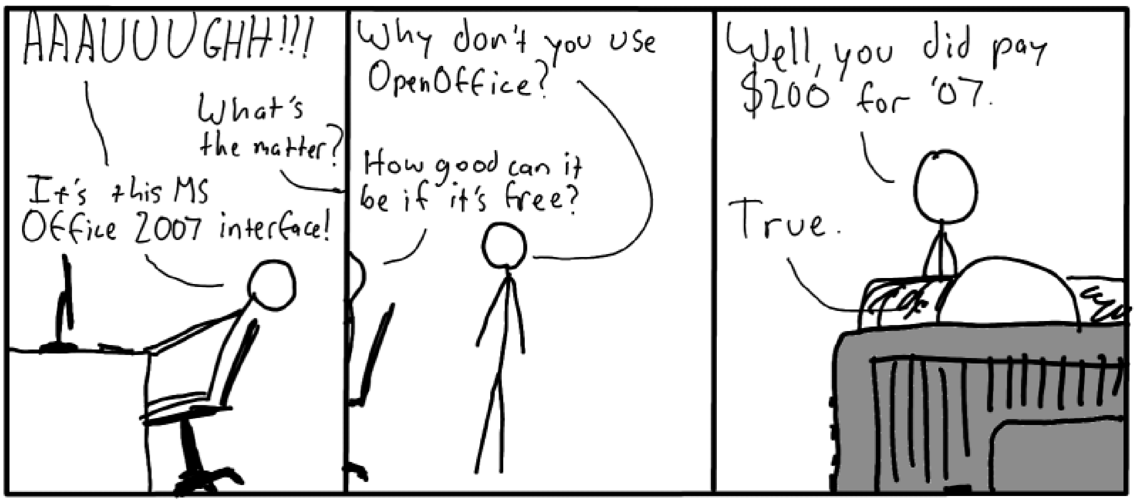
- slitaz-tools (3.4) — Lots of fixes and improvements
- tazpkg (3.2) — Box is much faster and deps auto-installed
- slitaz-boot-scripts (3.2) — WPA connections are much faster, boot time is logged and Xorg is autoconfigured at boot for any Live systems using it
- Package database for the Cooking version: 2226 (+ 835 from 2.0)

Bugs

Activity	Open	Closed
Bugs	58	50
Features	37	21
Tasks	9	38

- Based on current figures

Cartoon


AAAUUUGHH!!!
What's the matter?
It's this MS Office 2007 interface!
Why don't you use OpenOffice?
How good can it be if it's free?
Well, you did pay $200 for '07.
True.

Tips and Tricks

- The Many Uses of Screen[4]

Online

- Slitaz Linux — Zero to Distro in 30MB![5]

1.2 Issue 15

author linea, godane

- Written on 27th February 2010

Latest News

- SliTaz Cooking 20100221 released
- Newsletter translated into Chinese (thanks draplater)

[4] http://www.serverwatch.com/tutorials/article.php/3838961/The-Many-Uses-of-Screen.htm
[5] https://web.archive.org/web/20100212231315/http://linuxologist.com/reviews/slitaz-linux-zero-to-distro-in-30-mb/

New Packages

- vim-tiny
- smbfs
- libsasl
- libsasl-modules
- libcomerr3
- libkrb5support
- cgdb
- ccache
- distcc
- xorg-xf86-input-evtouch
- xorg-xf86-input-microtouch
- xorg-xf86-input-mutouch
- open-iscsi
- liblzma
- pidgin-libnotify
- xfce4-taskmanager
- xfce4-ristretto
- parole
- xfmpc
- xorg-xf86-input-void
- hal-scripts
- xorg-xf86-input-plpevtouch
- xfmedia
- dotconf
- speech-dispatcher
- yasr
- ncdu
- gfortran
- libgfortran
- octave
- fusecloop
- zsync
- tinc
- cloudvpn
- antiword
- l2tpd
- rp-lstp
- sctp-tools
- libsctp
- nat-tester
- evilvte
- lrzip
- wipe
- fsarchiver
- perl-gd

Updated Packages

- seamonkey ⇒ 2.0.2

- ncmpcpp ⇒ 0.5
- claws-mail-*
- gtkhtkl-viewer ⇒ 0.26
- rssyl ⇒ 0.26
- vala ⇒ 0.7.9
- mpd ⇒ 0.15.8
- libxml2 ⇒ 2.7.6
- mp ⇒ 5.1.3
- scite ⇒ 2.01
- beaver ⇒ 0.4.0rc1
- xvkbd ⇒ 3.1
- zim ⇒ 0.29
- mplayer-svn ⇒ 30605
- ntfs-3g ⇒ 2010.1.16
- xterm ⇒ 255
- lxpanel ⇒ 0.5.5

Improvements

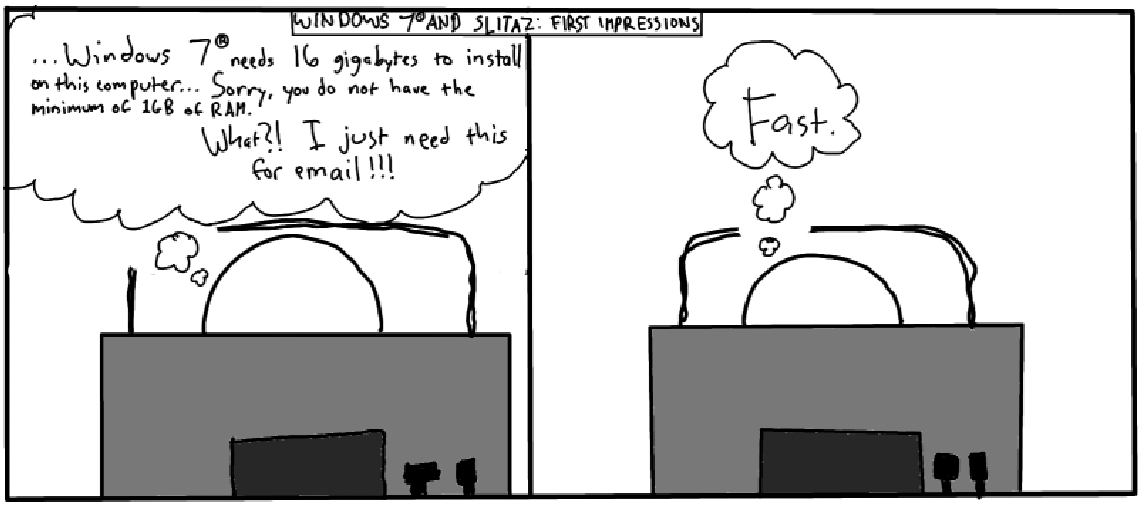
- tazlito (3.0) — Easier to customize LiveCD, GUI refactored, fixes and misc improvements
- slitaz-base-files (3.1) — 2 new scripts (man,ldd), OOO looks better and UTF-8
- slitaz-tools (3.5) — Better wifibox scan, UTF-8 locale with tazlocale, installer improved
- slitaz-doc (3.1) — Now provides a getting started guide

Bugs

Activity	Open	Closed
Bugs	68	57
Features	36	24
Tasks	8	39

- Based on current figures

Cartoon


WINDOWS 7® AND SLITAZ: FIRST IMPRESSIONS
...Windows 7® needs 16 gigabytes to install on this computer... Sorry, you do not have the minimum of 1GB of RAM.
What?! I just need this for email!!!
Fast.

Tips and Tricks

- How do I run a remote Linux desktop in Windows 7[6]

Online

- SliTaz Guide for survival (Live Linux) CD USB flash[7]

1.3 Issue 16

author linea, godane

- Written on 5th April 2010

Latest News

- SliTaz version 3.0 released

New Packages

- busybox-static
- httpfs-fuse-static
- ctags
- nss
- yad
- xorg-xf86-video-openchrome
- samba-common
- vym

[6] https://web.archive.org/web/20100402111607/http://blogs.techrepublic.com.com/window-on-windows/?p=2138
[7] http://www.fixya.com/support/r3885135-slitaz_guide_survival_live_linux_cd_usb

- perl-net-xwhois
- rgzip
- sslh

Updated Packages

- ntfs-3g ⇒ 2010.1.16
- tor ⇒ 2.1.23
- lxpanel ⇒ 0.5.5
- task ⇒ 1.9.0
- libgphoto2 ⇒ 2.4.8
- libsoup ⇒ 2.29.91
- mplayer-svn ⇒ 30817
- libdevmapper ⇒ 1.02.44
- cryptsetup ⇒ 1.1.0
- bluez ⇒ 4.62
- midori ⇒ 0.2.4
- wvdial ⇒ 1.61
- ppp-* ⇒ 2.4.5
- squid ⇒ STABLE25
- xorg-xf86-video-intel ⇒ 2.7.1
- privoxy ⇒ 3.0.16-stable
- sudo ⇒ 1.7.2p5
- lighttpd ⇒ 1.4.26
- openssh ⇒ 5.4p1
- sftp-server ⇒ 5.4p1
- xterm ⇒ 256
- nano ⇒ 2.2.3
- isomaster ⇒ 1.3.7
- mhwaveedit ⇒ 1.4.18
- gtk-gnutella ⇒ 0.96.8

Improvements

- tazlito (3.2) — box can now generate loram and loram/web
- slitaz-base-files (3.2) — release notes added
- slitaz-tools (3.8) — many improvements, po edits and fixes
- slitaz-doc (3.2) — release notes updated
- slitaz-boot-scripts (3.2) — updated
- slitaz-configs — small improvements
- slitaz-dev-tools — updated
- tazusb — ext2 and fat32 formatting added
- tazpkg — gui improvements and bug fixes

Bugs

Activity	Open	Closed
Bugs	85	73
Features	35	29
Tasks	26	43

- Based on current figures

Cartoon

Tips and Tricks

- A System Monitoring Tool Primer[8]

Online

- Greetings from SliTaz 3.0[9]

1.4 Issue 17

author linea, godane

- Written on 9th May 2010

Latest News

- SliTaz 'Summer of Documentation'

[8] https://web.archive.org/web/20100506120002/http://www.certcities.com/editorial/columns/story.asp?EditorialsID=413
[9] http://kmandla.wordpress.com/2010/03/29/greetings-from-slitaz-3-0/

New Packages

- aria2
- exo
- gtkperf
- liblinebreak
- fbreader
- enigma
- xfce4-notification
- mountlo
- speedometer
- python-urwid
- geequie
- mpc-library
- cmatrix
- ffmpeg-svn
- get-prince
- get-msttcorefonts
- gen-init-cpio
- wikiss
- groff
- linmodem-hsfmodem

Updated Packages

- evilvte ⇒ 0.4.6
- yad ⇒ 0.2.0
- freetype ⇒ 2.3.12
- lrzip ⇒ 0.45
- mpfr ⇒ 2.4.2
- gmp ⇒ 4.3.2
- binutils ⇒ 2.20.1
- linux ⇒ 2.6.33.2
- libbfd ⇒ 2.20.1
- gcc ⇒ 4.5.0
- glibc ⇒ 2.11.1
- broadcom-wl ⇒ 5.60.48.36
- pixman ⇒ 0.16.0
- vala ⇒ 0.7.10
- libgmp ⇒ 4.5.0
- autoconf ⇒ 2.65
- coreutils ⇒ 8.4
- dillo ⇒ 2.2
- lxtask ⇒ 0.1.3
- bison ⇒ 2.4.2
- cmake ⇒ 2.6.4
- cpio ⇒ 2.11
- libxml2 ⇒ 2.7.7
- abiword ⇒ 2.8.4
- libart-lgpl ⇒ 2.3.21
- mpd ⇒ 0.15.9

- ncmpcpp ⇒ 0.5.3
- dstat ⇒ 0.7.1
- openssl ⇒ 1.0.0
- openssh ⇒ 5.5p1
- libssl ⇒ 1.0.0
- xorg-*proto ⇒ 1.2.0
- libcrypto ⇒ 1.0.0
- libdrm ⇒ 2.4.20
- xorg-lib* ⇒ various
- mesa ⇒ 7.8.1
- xorg-server ⇒ 1.8.0
- sftp-server ⇒ 5.5p1
- ndiswrapper ⇒ 1.56
- xorg-* ⇒ various
- xorg-xf86-video-* ⇒ various
- zile ⇒ 2.3.15
- kismet ⇒ 2010-01-R1
- emacs-pkg-lua-mode ⇒ 20100404
- hal-info ⇒ 20091130
- libedit ⇒ 3.0
- pm-utils ⇒ 1.2.6.1
- boxbackup-* ⇒ 0.11rc8
- curl ⇒ 7.20.1
- wine ⇒ 1.1.43
- sylpheed ⇒ 3.0.2
- cairo ⇒ 1.8.10
- nvidia ⇒ 195.36
- linmodem ⇒ 2.1.80~20091225
- fuse ⇒ 2.8.4
- git ⇒ 1.7.1
- vlc ⇒ 1.0.6
- ffmpeg ⇒ 0.5.1
- mercurial ⇒ 1.5.2
- subversion ⇒ 1.6.11
- zlib ⇒ 1.2.5
- sqlite ⇒ 3.6.23.1
- tcl ⇒ 8.5.8
- conky ⇒ 1.8.0
- audacity ⇒ 1.3.12
- asunder ⇒ 1.9.3
- grsync ⇒ 1.1.0
- audiofile ⇒ 0.2.7
- lguest ⇒ 2.6.33.2
- libwebkit ⇒ 1.2.0
- webkit-web-inspector ⇒ 1.2.0
- parted ⇒ 2.2
- gparted ⇒ 0.5.2
- grub2 ⇒ 1.98
- cmake ⇒ 2.8.1
- glib ⇒ 2.25.2
- libgio ⇒ 2.25.2

- pango ⇒ 1.28.0
- bazaar ⇒ 2.1.0
- avidemux ⇒ 2.5.2
- cdrkit ⇒ 1.1.10
- readom ⇒ 1.1.10
- atk ⇒ 1.30.0
- aircrack-ng ⇒ 1.1
- gtk+ ⇒ 2.20.1
- firefox ⇒ 3.6.3
- irssi ⇒ 0.8.15
- bluefish ⇒ 2.0.0
- btrfs-progs ⇒ 0.19
- libtool ⇒ 2.2.6b
- e2fsprogs ⇒ 1.41.11
- menu-cache ⇒ 0.3.2
- openbox ⇒ 3.4.11.1
- mtpaint ⇒ 3.31
- clamav ⇒ 0.96
- tor ⇒ 0.2.1.26
- openal ⇒ 1.2.854
- bzip2 ⇒ 1.0.5
- bzlib ⇒ 1.0.5
- gaijim ⇒ 0.13.4
- gtkpod ⇒ 0.9.16
- neon ⇒ 0.29.3
- bind ⇒ 9.7.0-P1
- goffice ⇒ 0.8.2
- libgsf ⇒ 1.14.16
- transmission-* ⇒ 1.93
- glibmm ⇒ 2.24.2
- libgiomm ⇒ 2.24.2
- cariomm ⇒ 1.8.4
- pangomm ⇒ 2.26.2
- gtkmm ⇒ 2.20.3
- pkg-config ⇒ 0.23
- pmount ⇒ 0.9.20
- gnutls ⇒ 2.8.6
- perl-net-ssleay ⇒ 1.36
- dahdi-* ⇒ 2.3.0
- x11vnc-* ⇒ 0.9.10
- asterisk ⇒ 1.6.2.7
- diffutils ⇒ 3.0
- cups-pam ⇒ 1.4.2
- davfs2 ⇒ 1.4.6
- ruby ⇒ 1.9.1
- ptlib ⇒ 2.6.5
- opal ⇒ 3.6.6
- ekiga ⇒ 3.2.6
- seamonkey ⇒ 2.0.4
- linmodem-slmodem ⇒ 2.9.11-20100303
- depmod ⇒ 3.11.1

- module-init-tools ⇒ 3.11.1
- patch ⇒ 2.6.1
- wireshark ⇒ 1.2.8
- testdisk ⇒ 6.11.3
- file ⇒ 5.04
- libmagic ⇒ 5.04
- pidgin ⇒ 2.6.6
- claws-mail-* ⇒ various
- gtkhtml2-viewer ⇒ 2.27
- rssl ⇒ 0.27

Improvements

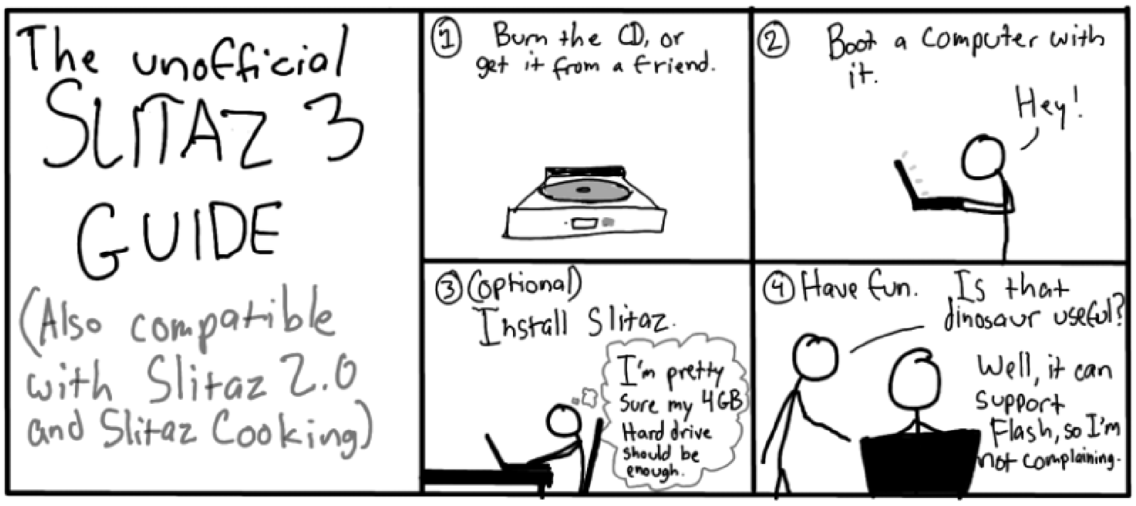
- New toolchain
- tazwok updated (3.2)

Bugs

Activity	Open	Closed
Bugs	94	73
Features	37	29
Tasks	27	43

- Based on current figures

Cartoon


The unofficial SLITAZ 3 GUIDE
(Also compatible with Slitaz 2.0 and Slitaz Cooking)
① Burn the CD, or get it from a friend.
② Boot a computer with it.
Hey!
③ (optional) Install Slitaz.
I'm pretty sure my 4GB Hard drive should be enough.
④ Have fun.
Is that dinosaur useful?
Well, it can support Flash, so I'm not complaining.

Tips and Tricks

- How-To: Grep Tricks for Linux Users[10]

[10] http://www.itworld.com/it-managementstrategy/106032/how-to-grep-tricks-linux-users

Online

- SliTaz Linux 3.0 lends credence to the phrase “Small but Powerful”[11]

1.5 Issue 18

author linea, godane

- Written on 11th July 2010

Latest News

- SliTaz moves to new FSF server

New Packages

- qt4-phonon
- php-curl
- libegl-mesa
- emacs po-mode
- gimp-dev
- libburn
- libisofs
- xfburn
- liferea
- arora
- libcanberra
- xvidcore
- barcode
- echinus
- gtk-theme-switch
- gtk-engines-rezlooks
- deadbeef
- linux-logfs
- linux-configfs
- linux-toshiba
- linux-dlm
- qiv
- normalize
- python-pybluez
- attica
- wxpython
- strigi
- gammu
- ypserv
- yp-tools
- ypbind-mt
- nis

[11] https://web.archive.org/web/20100404015556/http://www.linuxcritic.com:80/slitaz-linux-30-lends-credence-phrase-small-powerful/

- git-gui
- cpufrequtils
- python-mygpoclient
- gpodder
- turnserver
- osip
- exosip
- linphone
- bygfoot

Updated Packages

- claws-mail-extras ⇒ 3.7.6
- boxbackup ⇒ 0.11rc8
- yad ⇒ 0.2.1
- zile ⇒ 2.3.1.6
- lives ⇒ 1.3.3
- wine ⇒ 1.1.44
- aria2 ⇒ 1.9.3
- slang ⇒ 2.2.2
- libpng ⇒ 1.2.43
- tint2 ⇒ 0.9
- emelfm2 ⇒ 0.7.2
- pngcrush ⇒ 1.7.10
- optipng ⇒ 0.6.4
- yasm ⇒ 1.0.0
- nasm ⇒ 2.08.01
- gfortran ⇒ 4.5.0
- libarchive ⇒ 2.8.3
- pidgin ⇒ 2.7.1
- gettext ⇒ 0.18
- linux ⇒ 2.6.34
- lguest ⇒ 2.6.34
- iptables ⇒ 1.4.7
- emacs ⇒ 23.2
- mplayer-svn ⇒ 31179
- goffice ⇒ 0.8.3
- xvkbd ⇒ 3.2
- libxcb ⇒ 1.6
- xcb-proto ⇒ 1.6
- xorg-xf86-video-{openchrome, nv, s3, neomagic, mga}
- catalyst ⇒ 10.4
- midori ⇒ 0.2.6
- libwebkit ⇒ 1.2.1
- elinks ⇒ 0.11.7
- webkit-web-inspector ⇒ 1.2.1
- avidemux ⇒ 2.5.3
- yasm ⇒ 1.0.1
- awesome ⇒ 3.4.5
- pekwm ⇒ 0.1.12
- bastet ⇒ 0.43

- glibc ⇒ 2.11.2
- sudoku-savant ⇒ 1.3
- e2fsprogs ⇒ 1.41.12
- minicom ⇒ 2.4
- udev ⇒ 156
- nscd ⇒ 2.10.2
- task ⇒ 1.9.1
- clamav ⇒ 0.96.1
- vorbis-tools ⇒ 1.4.0
- libogg ⇒ 1.2.0
- libvorbis ⇒ 1.3.1
- libtheora ⇒ 1.1.1
- parted ⇒ 2.3
- alsa-lib ⇒ 1.0.23
- scribus ⇒ 1.3.7
- busybox ⇒ 1.17.0
- xfce4-taskmanager ⇒ 1.0.0
- geany ⇒ 0.19
- p7zip ⇒ 9.13
- ario ⇒ 1.4.4
- sudo ⇒ 1.7.2p7
- fetchmail ⇒ 6.3.17
- usbutils ⇒ 0.86
- libusb ⇒ 1.0.8
- libusb-compact ⇒ 0.1.3
- amsn ⇒ 0.98.3
- amule ⇒ 2.2.6
- libxml2-tools ⇒ 2.7.7
- osmo ⇒ 0.2.10
- dmsetup ⇒ 1.02.44
- bluez-dev ⇒ 4.62
- nscd ⇒ 2.11.2
- vte ⇒ 0.25.1
- krb5 ⇒ 1.8.2
- strace ⇒ 4.5.20
- pyqt-x11-gpl ⇒ 4.7.3
- speex ⇒ 1.2rc1

Improvements

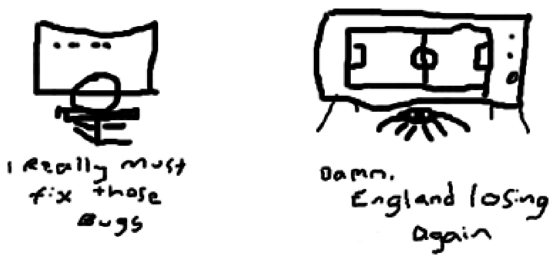
- tazwok updated (3.2.1)
- slitaz-tools updated (3.8.1)
- slitaz-boot-scripts updated (3.4.1)

Bugs

Activity	Open	Closed
Bugs	101	75
Features	37	29
Tasks	28	43

- Based on current figures

Cartoon



Tips and Tricks

- Roundup — Text Based Text Editors[12]

Online

- Slitaz Tutorials (youtube)[13]

1.6 Issue 19

author linea, godane

- Written on 29th August 2010

Latest News

- Handbook translated and updated into Russian (thanks Lexeii, Onion, Cyril)

[12] http://fosswire.com/post/2010/07/text-based-text-editors/
[13] https://www.youtube.com/watch?v=6YQrBgNGQrw

New Packages

- hostapd
- xfce4-dev-tools
- tmux
- tazwikiss
- exiftool
- libgtkimageview
- ufraw
- macchanger
- libcss
- libwapcaplet
- elfutils
- lxrandr
- lxterminal
- masqmail
- wbarconf
- unrar
- perl-rfc-rfc822-address
- perl-parse-recdescent
- perl-text-iconv
- silc-server
- nut
- uclibc-cross-compiler
- openttd
- opengfx
- sysstat
- icecast
- uget
- glew
- get-google-talkplugin
- qemu-{arm,mips,ppc,x86}
- puzzles
- ruby-gtk2
- thunderbird
- thunderbird-langpack-*

Updated Packages

- sip ⇒ 4.10.2
- busybox ⇒ 1.17.1
- pidgen ⇒ 2.7.1
- john ⇒ 1.7.6
- isof ⇒ 4.83
- chkrootkit ⇒ 0.49
- libiec61883 ⇒ 1.2.0
- logrotate ⇒ 3.7.9
- calcurse ⇒ 2.8
- nss-ldap ⇒ 265
- pam-ldap ⇒ 185
- c-client ⇒ 2007e

- exo ⇒ 0.3.107
- goffice ⇒ 0.8.6
- gnumeric ⇒ 1.10.7
- xfce4* ⇒ 4.6.2
- thunar ⇒ 1.0.2
- libxfcegui4 ⇒ 4.6.4
- libraw1394 ⇒ 2.0.5
- task ⇒ 1.9.2
- fotoxx ⇒ 10.8
- printoxx ⇒ 2.7
- zim ⇒ 0.48
- xorg-xf86-input-synaptics ⇒ 1.2.99.1
- keepass ⇒ 0.4.3
- tmux ⇒ 1.3
- py3k ⇒ 3.1.2
- xchat ⇒ 3.8.8
- xorg-xf86-input-elographics ⇒ 1.2.4
- xorg-xf86-input-plpevtch ⇒ 0.5.0
- beaver ⇒ 0.4.0
- libmng ⇒ 1.0.10
- netsurf ⇒ 2.5
- memtest ⇒ 4.10
- wine ⇒ 1.2
- wesnoth ⇒ 1.8.3
- gcc ⇒ 4.5.1
- catalyst ⇒ 10.7
- cpufrequtils ⇒ 008
- exiftool ⇒ 8.27
- dbus ⇒ 1.2.24
- apache-ant ⇒ 1.8.1
- krb5 ⇒ 1.8.3
- libpng ⇒ 1.4.3
- jpeg ⇒ 8b
- lirc ⇒ 0.8.6
- ecj ⇒ 3.6
- dbus-glib ⇒ 0.86
- dbus-python ⇒ 0.83.1
- eggdbus ⇒ 0.6
- Policykit ⇒ 0.97
- GConf ⇒ 2.31.7
- bazaar ⇒ 2.2.0
- glib ⇒ 2.25.13
- libftdi ⇒ 0.18
- scite ⇒ 2.10
- srcpd ⇒ 2.1.1
- libgdiplus ⇒ 0.18
- mono ⇒ 2.6.7
- pixman ⇒ 0.18.0
- icedtea6 ⇒ 1.8.1
- pango ⇒ 1.28.1
- freetype ⇒ 2.4.2

- zile ⇒ 2.3.19
- yad ⇒ 0.3.1
- evince ⇒ 2.30.3
- libproxy ⇒ 0.4.2
- firefox ⇒ 3.6.8
- qemu ⇒ 0.12.5
- putty ⇒ 0.60-2010-08-29
- kismet ⇒ 2010-07-R1
- libedit ⇒ 20100424-3.0
- pm-utils ⇒ 1.4.1
- powertop ⇒ 1.13
- xorg-xf86-video-neomagic ⇒ 1.2.5

Improvements

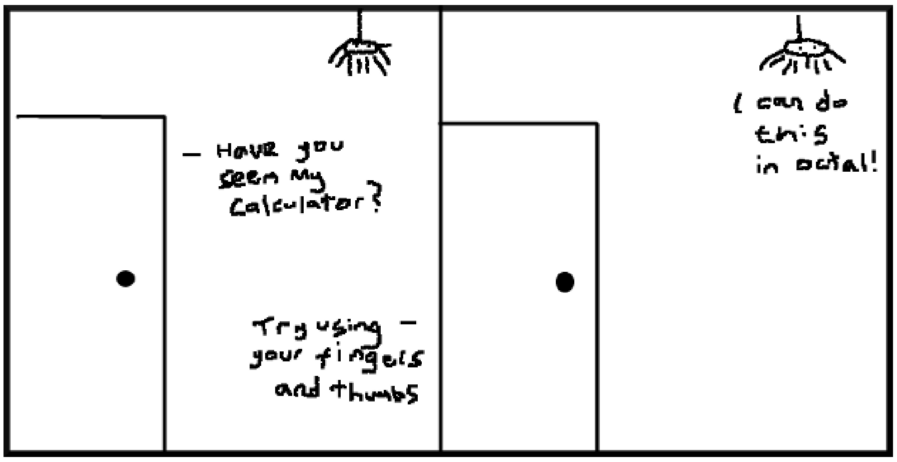
- tazbb — add stable support for log.php
- mirror-tools — add mirror-info web page

Bugs

Activity	Open	Closed
Bugs	97	86
Features	39	31
Tasks	26	33

- Based on current figures

Cartoon


– Have you seen my Calculator?
Try using – your fingers and thumbs
I can do this in octal!

Tips and Tricks

- Text editing with Nano made easy[14]

[14] https://web.archive.org/web/20100624083209/http://tuxradar.com/content/text-editing-nano-made-easy

Online

- One floppy, dozens of tools[15]

1.7 Issue 20

author linea

- Written on 16th October 2010

Latest News

- Scratchbook translated into Russian (thanks Lexeii, Onion, Cyril)
- Over 400 packages updated

New Packages

- clucene
- fbxkb
- droid-font
- uclibc
- recorder
- rt-source
- fbida
- cd-discid
- rubyripper
- grub4dos-linux
- libtirpc
- nfs-utils
- rpcbind
- surf
- libplist
- adeskbar
- polipo
- vidalia
- aspell-da
- zsnes
- slsnif
- scummvm
- frogatto
- xbill
- devede
- gen-gs
- rawstudio
- rsnapshot
- stellarium
- accessx

[15] http://kmandla.wordpress.com/2010/07/23/one-floppy-dozens-of-tools/

- iron-linux
- warzone2100
- pwgen
- ftgl
- ilmbase
- openexr
- cinepaint
- lincity-ng
- chrpath
- ldapvi
- assaultcube
- libdnet
- xcircuit
- gdk-pixbuf
- winetricks
- linux-api-headers
- btanks

Updated Packages (abridged)

- rddtool ⇒ 1.4.4
- transmission ⇒ 2.10
- libwebkit ⇒ 1.24
- vlc ⇒ 1.1.4
- midori ⇒ 0.2.8
- git ⇒ 1.7.3.1
- mercurial ⇒ 1.6.4
- abiword ⇒ 2.8.6
- apache ⇒ 2.2.16
- lighttpd ⇒ 1.4.28
- clamav ⇒ 0.96.3
- audacious ⇒ 2.4.0
- asunder ⇒ 2.0
- sudo ⇒ 1.7.4p4
- goffice ⇒ 0.8.10
- bluefish ⇒ 2.0.1
- aria2 ⇒ 1.10.3
- thunderbird ⇒ 3.1.3
- firefox ⇒ 3.6.10
- pkg-config ⇒ 0.25
- automake ⇒ 1.11.1
- tar ⇒ 1.23
- bluez ⇒ 4.75
- geany ⇒ 0.19.1
- ntfs-3g ⇒ 2010.8.8
- gimp ⇒ 2.6.11
- babl ⇒ 0.1.2
- gparted ⇒ 0.6.4
- gegl ⇒ 0.1.2
- gnumeric ⇒ 1.10.11
- cups ⇒ 1.4.4

- openssh ⇒ 5.6p1
- openssl ⇒ 1.0.0a
- buildbot ⇒ 0.8.1
- cabextract ⇒ 1.3
- pidgin ⇒ 2.7.3
- seamonkey ⇒ 2.0.8
- gphoto ⇒ 2.4.9
- gtksourceview ⇒ 2.10.4
- cdrdao ⇒ 1.2.3
- chmlib ⇒ 0.40
- chocolate-doom ⇒ 1.4.0
- aaphoto ⇒ 0.38
- dosbox ⇒ 0.74
- hardinfo ⇒ 0.5.1
- sqlite ⇒ 3.7.2
- lcms ⇒ 0.19
- lame ⇒ 3.98.4
- centerim ⇒ 4.22.9
- curl ⇒ 7.21.1
- gnomeplayer ⇒ 0.9.9.2
- mplayer ⇒ 1.0rc3
- smplayer ⇒ 0.6.9
- taglib ⇒ 1.6.3
- easytag ⇒ 2.1.6
- ddrescue ⇒ 1.13
- cryptsetup ⇒ 1.1.3
- poppler ⇒ 0.14.3
- nano ⇒ 2.2.5
- filezilla ⇒ 3.3.4.1
- zsh ⇒ 4.3.10
- xterm ⇒ 262
- pixman ⇒ 0.18.4
- wormux ⇒ 0.9.2.1
- wpa_supplicant ⇒ 0.7.3
- lxterminal ⇒ 0.1.9
- lxappearance ⇒ 0.4.0
- xine-lib ⇒ 1.1.19
- grsync ⇒ 1.1.1
- gpodder ⇒ 2.8
- beaver ⇒ 0.4.1
- fluxbox ⇒ 1.1.1
- bluefish ⇒ 2.0.2
- gtkspell ⇒ 2.0.16
- bison ⇒ 2.4.3
- m4 ⇒ 1.4.15
- gawk ⇒ 3.1.8
- ghostscript ⇒ 9.00
- epeak ⇒ 1.44.05
- jack-audio-connection-kit ⇒ 0.118.0
- iptables ⇒ 1.4.9.1
- gxine ⇒ 0.5.905

- imagemagick ⇒ 6.6.4-10
- dosfstools ⇒ 3.0.10
- dialog ⇒ 1.1-20100428
- cmake ⇒ 2.8.2
- gettext ⇒ 0.18.1.1
- gdb ⇒ 7.2
- nitrogen ⇒ 1.5.1
- readline ⇒ 6.1
- gstreamer ⇒ 0.10.30
- recordmydesktop ⇒ 0.3.8.1
- samba ⇒ 3.5.6
- wireshark ⇒ 1.4.1
- wicd ⇒ 1.7.0
- sylpheed ⇒ 3.0.3
- bzip2 ⇒ 1.0.6
- nmap ⇒ 5.21
- pcre ⇒ 8.10
- tiff ⇒ 3.9.4
- git-gui ⇒ 0.13.0
- homebank ⇒ 4.3
- lxpanel ⇒ 0.5.6
- tcl, tk ⇒ 8.5.9
- mutt ⇒ 1.5.21
- ruby ⇒ 1.9.2-p0
- mirage ⇒ 0.9.5.1
- mpg123 ⇒ 1.12.4
- gnuplot ⇒ 4.4.0
- feh ⇒ 1.9
- parcelite ⇒ 0.9.2
- awesome ⇒ 3.4.8
- subversion ⇒ 1.6.13
- make ⇒ 3.82
- conky ⇒ 1.8.1
- php ⇒ 5.2.14
- postgresql ⇒ 9.0.1
- atk ⇒ 1.32.0
- cairo ⇒ 1.10.0
- pango ⇒ 1.28.3
- gtk+ ⇒ 2.22.0
- wine ⇒1.2.1
- hplip ⇒ 3.10.9
- busybox ⇒ 1.17.3
- curl ⇒ 7.21.2
- xorg-server ⇒ 1.8.2

Improvements

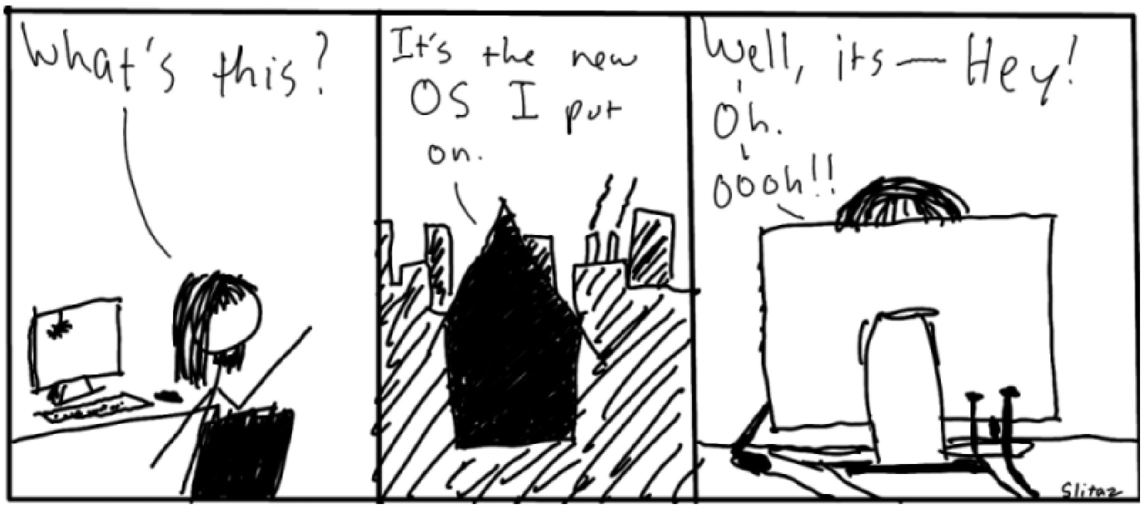
- Website updated (mailing list, artwork, sitemap)

Bugs

Activity	Open	Closed
Bugs	102	86
Features	39	31
Tasks	24	48

- Based on current figures

Cartoon



Tips and Tricks

- How To Use Bash Parameter Substitution Like A Pro[16]

Online

- Spotlight on Linux: SliTaz GNU/Linux 3.0[17]

1.8 Issue 21

author linea

- Written on 5th December 2010

Latest News

- Guides translated into Russian (thanks Lexeii, Onion, Cyril)
- Over 400 packages updated

[16] http://www.cyberciti.biz/tips/bash-shell-parameter-substitution-2.html
[17] http://www.linuxjournal.com/content/spotlight-linux-slitaz-gnulinux-30

New Packages

- libbonobo
- rarian
- libasyncns
- bin86
- hd2u
- mupen64plus
- pcsxr-svn
- ladspa
- raptor
- liblrdf
- libtar
- hydrogen
- bmpanel2
- aubio
- fftw
- frei0r-plugins
- liblo
- mlt
- fluidsynth
- ardour
- openshot
- libquicktime
- x264
- mupdf
- python-chardet
- jbig2dep
- openjpeg
- gavl
- swig
- vde2
- watchdog
- goocanvas
- portmidi
- mixxx
- singularity
- lmms
- deluge
- tinyproxy
- docbook-xml-{5,4.*}
- docbook-xsl-{ns}
- iasl
- ndisc6
- fop
- flam3
- electricsheep
- nanochess
- p4wn
- checkers
- othello

- hatenarunner
- chess3d
- sudoku
- jstetris
- virtualbox-ose
- qtconfig
- atkmm
- perl-file-tail
- txt2tags
- udhcpc6-fake
- lorcon
- libnet
- airpwn
- icewm
- cssc
- aspell-pl
- xorg-xf86-video-r128
- libfm
- pcmanfm2
- atftp
- aspell-nl
- cherokee
- linux-firmware
- stella
- sleuthkit
- linux-nfsd
- linux-autofs
- autofs
- aufs
- geoip
- wavpack
- libass
- xorg-xf86-video-vmware
- which
- fcitx
- python-notify
- decibel
- stardict
- firefox-langpack-zh_CN
- wordpress
- drupal
- perl-locale-gettext
- perl-yaml
- libsdl-perl
- libsdl-pango
- frozen-bubble
- vnc2flv
- linux-iscsi
- aescrypt
- whois
- zvbi

- meld
- libglademm
- visualboyadvance
- aiksaurus
- lyx

Updated Packages (abridged)

- xorg ⇒ 1.9.2
- libsndfile ⇒ 1.0.23
- wireshark ⇒ 1.4.2
- curl ⇒ 7.21.2
- libdrm ⇒ 2.4.22
- mesa ⇒ 7.8.2
- mpd ⇒ 1.15.15
- ncmpcpp ⇒ 5.5
- transmission ⇒ 2.12
- mtools ⇒ 4.0.14
- pyopenssl ⇒ 0.10
- pycairo ⇒ 1.8.10
- pygobject ⇒ 2.26.0
- pygtk ⇒ 2.22.0
- cdrkit ⇒ 1.1.11
- openldap ⇒ 2.4.23
- viewnoir ⇒ 1.0
- apr-util ⇒ 1.3.10
- apache ⇒ 2.2.17
- libsamplerate ⇒ 0.1.7
- dbus ⇒ 1.4.0
- udev ⇒ 163
- hal ⇒ 0.5.14
- freeimage ⇒ 3141
- vte ⇒ 0.27.2
- coreutils ⇒ 8.6
- mgetty ⇒ 1.1.37
- firefox ⇒ 3.6.12
- thunderbird ⇒ 3.0.15
- icedtea6-{jdk,jre} ⇒ 1.9.1
- libfirefox ⇒ 3.6.12
- aria2 ⇒ 1.10.5
- arora ⇒ 0.11.0
- gnutls ⇒ 2.10.2
- pidgin ⇒ 2.7.7
- ffmpeg ⇒ 0.6.1
- qemu ⇒ 0.13.0
- ffplay ⇒ 0.6.1
- nss ⇒ 3.12.8
- vlc ⇒ 1.1.15
- adeskbar ⇒ 0.4.2
- python-pygame ⇒ 1.9.1release
- libcap ⇒ 2.19

- pam ⇒ 1.1.2
- squirrelmail ⇒ 1.4.21
- valgrind ⇒ 3.6.0
- vala ⇒ 0.10.0
- syslinux ⇒ 4.03
- mono ⇒ 2.8.1
- clamav ⇒ 0.96.5
- scite ⇒ 2.22
- seamonkey ⇒ 2.0.10
- tar ⇒ 1.25
- iptables ⇒ 1.4.10
- poedit ⇒ 1.4.6.1
- fotoxx ⇒ 10.12
- dstat ⇒ 0.7.2
- remind ⇒ 03.01.10
- midori ⇒ 0.2.9
- libvorbis ⇒ 1.3.2
- mysql ⇒ 5.1.53
- aria2 ⇒ 1.10.6
- libpcap ⇒ 1.1.1
- snort ⇒ 2.9.0.2
- mtr ⇒ 0.80
- ncdu ⇒ 1.7
- inkscape ⇒ 0.48.0
- mercurial ⇒ 1.7.2
- dhcp ⇒ 4.2.0-P1
- cmake ⇒ 2.8.3
- linux ⇒ 2.6.36
- gnome-mplayer ⇒ 1.0.0
- gecko-mediaplayer ⇒ 1.0.0
- zim ⇒ 0.49
- amule ⇒ r10365
- squid ⇒ 3.1.9
- alsaplayer ⇒ 0.99.81
- zip ⇒ 3.0
- ruby-gtk2 ⇒ 0.90.5
- mc ⇒ 4.7.0.10
- qt4 ⇒ 4.7.1
- cups ⇒ 1.4.5
- bluez ⇒ 4.80
- perl-xml-parser ⇒ 2.40
- privoxy ⇒ 3.0.17
- glib ⇒ 2.26.1
- gtk+ ⇒ 2.22.1
- gphoto2 ⇒ 2.4.10
- filezilla ⇒ 3.3.5.1
- elinks ⇒ 0.13
- pcmciautils ⇒ 017
- pciutils ⇒ 3.1.7
- usbutils ⇒ 0.91
- pixman ⇒ 0.20.0

- zenity ⇒ 2.32.1
- claws-mail ⇒ 3.7.8
- xane ⇒ 0.998
- imagemagick ⇒ 6.6.5-9
- libxml2 ⇒ 2.7.8
- nano ⇒ 2.2.6
- scribus ⇒ 1.3.8
- busybox ⇒ 1.17.4
- x11vnc ⇒ 0.9.12
- uget ⇒ 1.6.1
- subversion ⇒ 1.6.15
- tor ⇒ 0.2.1.27
- gnumeric ⇒ 1.10.12
- dahdi-tools ⇒ 2.4.0
- rpm4 ⇒ 4.8.1
- hydra ⇒ 5.9
- libsoup ⇒ 2.32.2
- python ⇒ 2.7.1
- gstreamer ⇒ 0.10.31
- GConf ⇒ 2.32.1
- bind ⇒ 9.7.2-P3
- freetype ⇒ 2.4.4
- openssl ⇒ 1.0.0c
- geany ⇒ 0.19.2
- gajim ⇒ 0.14.1
- mpfr ⇒ 3.0.0p8
- poppler ⇒ 0.14.5
- pidgin-facebookchat ⇒ 1.69
- wine ⇒ 1.2.2
- twisted ⇒ 10.2.0
- git ⇒ 1.7.3.3

Improvements

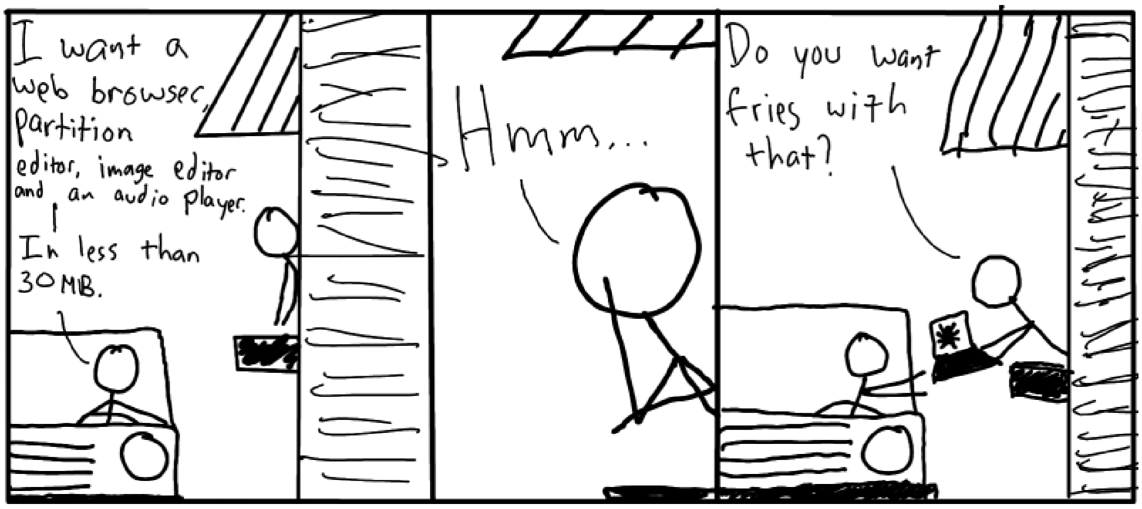
- slitaz-tools (3.9.1)
- slitaz-tools-boxes (3.9.1)
- slitaz-configs (4.0)
- slitaz-base-files (4.1)
- slitaz-doc (4.1)
- tazpkg (4.1.1)

Bugs

Activity	Open	Closed
Bugs	107	90
Features	41	32
Tasks	24	48

- Based on current figures

Cartoon


I want a web browser, partition editor, image editor and an audio player. In less than 30MB.
Hmm...
Do you want fries with that?

Tips and Tricks

- Linux Server Troubleshooting With strace[18]

Online

- SliTaz GNU/Linux 3.0 Screencast Review[19] (video[20] unavailable)

1.9 Issue 22

author linea, godane

- Written on 30th January 2011

Latest News

- Website, Wiki, Labs updated
- Newsletter translated into Spanish (thanks seacat)

New Packages

- ecm
- mdk3
- airoscript-ng

[18] https://web.archive.org/web/20101127160719/http://www.linuxplanet.com/linuxplanet/tutorials/7229/1/

[19] https://web.archive.org/web/20101203135945/http://linuxfilesystem.com/uncategorized/slitaz-gnulinux-3-0-screencast-review

[20] https://www.youtube.com/watch?v=y1a8_J8KDEU

- php-dbase
- pycurl
- pidgin-musictracker
- linapple
- dvgrab
- ogmrip
- at
- chntpw
- disktype
- fakeroot
- xorg-xf86-input-vmmouse
- caps
- alsaequal
- linux-without-modules
- TeXmacs
- bzflag
- pysqlite
- celestia
- perl-libwww
- pyrex
- pywebkitgtk
- pyorbit
- gnome-mime-data
- php-soap
- phpvirtualbox
- libmicrohttpd
- gnome-vfs
- fcitx-zm
- gnome-python
- miro
- clisp
- cowpatty
- asleap
- scrub
- postfixadmin
- igmpproxy
- wyrd
- pktstat
- get-LibreOffice
- weechat
- qrencode
- phpqrcode
- perl-smtp-ssl
- cairo-dock
- cairo-dock-plugins
- gambas2
- util-linux-ng-setterm
- yajl
- urlgrabber
- perl-text-csv
- polkit

- garcon
- libxfce4ui
- gzip
- vzctl
- vzquota
- thunar-vfs
- ovz-web-panel
- sqlite3-ruby
- xorg-makedepend
- libQtTest
- libssh
- openmpi
- shellinabox
- freerdp
- remminia

Updated Packages (abridged)

- scribus ⇒ 1.3.9
- libogg ⇒ 1.2.2
- gpodder ⇒ 2.12
- gnutls ⇒ 2.10.4
- feh ⇒ 1.11
- xterm ⇒ 267
- gst-plugins-good ⇒ 0.10.26
- cairomm ⇒ 1.8.6
- lvm2 ⇒ 2.02.81
- clutter-gtk ⇒ 0.10.8
- aria2 ⇒ 1.10.9
- fotoxx ⇒ 11.01
- whois ⇒ 5.0.10
- py3k ⇒ 3.1.3
- firefox ⇒ 3.6.13
- seamonkey ⇒ 2.0.11
- thunderbird ⇒ 3.1.7
- audacious ⇒ 2.4.3
- transmission ⇒ 2.13
- nasm ⇒ 2.09.04
- cherokee ⇒ 1.0.18
- libisofs ⇒ 0.6.40
- libburn ⇒ 0.9.0
- dhcp ⇒ 4.2.0-P2
- pcre ⇒ 8.12
- sqlite ⇒ 3.7.4
- php ⇒ 5.2.17
- docbook-xsl ⇒ 1.76.1
- zenity ⇒ 2.32.1
- gparted ⇒ 0.7.1
- xorg-server ⇒ 1.9.3
- buildbot ⇒ 0.8.2
- beautifulsoup ⇒ 3.2.0

- e2fsprogs ⇒ 1.41.14
- mixxx ⇒ 1.8.2
- curl ⇒ 7.21.3
- mysql ⇒ 5.1.54
- postgresql ⇒ 9.0.2
- postfix ⇒ 2.8.0
- freeciv ⇒ 2.2.4
- git ⇒ 1.7.3.5
- viewnoir ⇒ 1.1
- drupal ⇒ 6.20
- scite ⇒ 2.23
- udev ⇒ 1.6.5
- libdrm ⇒ 2.4.23
- terminal ⇒ 0.4.5
- evilvte ⇒ 0.4.7
- bogofilter ⇒ 1.2.2
- bluez ⇒ 4.82
- pyneighborhood ⇒ 0.5.3
- lmms ⇒ 0.4.9
- wipe ⇒ 2.3.1
- tor ⇒ 0.2.1.29
- pidgin ⇒ 2.7.9
- leafpad ⇒ 0.8.18.1
- zsh ⇒ 4.3.11
- snort ⇒ 2.9.0.3
- parcellite ⇒ 0.9.3
- busybox ⇒ 1.18.2
- netatalk ⇒ 2.1.5
- lvm2 ⇒ 2.02.79
- winetricks ⇒ 20110123
- ethtool ⇒ 2.6.36
- speedometer ⇒ 2.7
- task ⇒ 1.9.3
- units ⇒ 1.88
- cryptsetup ⇒ 1.2.0
- deadbeef ⇒ 0.4.4
- exiv ⇒ 0.20
- zile ⇒ 2.3.21
- sip ⇒ 4.12
- fsarchiver ⇒ 0.6.12
- ruby ⇒ 1.9.2-p136
- mc ⇒ 4.7.5
- vala ⇒ 0.10.2
- mercurial ⇒ 1.7.3
- wordpress ⇒ 3.0.4
- elfutils ⇒ 0.151
- yad ⇒ 0.7.2
- libwebkit ⇒ 1.2.6
- rrdtool ⇒ 1.4.5
- htop ⇒ 0.9
- cups ⇒ 1.4.6

- geany ⇒ 0.20
- shell-fm ⇒ 0.7
- bash ⇒ 4.1
- module-init-tools ⇒ 1.12
- make ⇒ 3.82
- coreutils ⇒ 0.9
- isomaster ⇒ 1.3.8
- ncmpcpp ⇒ 0.5.6
- sudo ⇒ 1.7.4p5
- wireshark ⇒ 1.4.3
- nicotine ⇒ 1.2.16
- mlt ⇒ 0.6.0
- mpc ⇒ 0.20
- gnupg ⇒ 2.0.17
- dbus ⇒ 1.4.1
- file ⇒ 5.0.5
- exo ⇒ 0.6.0
- xfce4-* ⇒ 4.8.0
- dmidecode ⇒ 2.11
- gst-plugins-{good, bad, base, ugly} ⇒ 0.10.*
- libdrm ⇒ 2.4.23
- mesa ⇒ 7.10
- linux ⇒ 2.6.37
- aufs ⇒ 20110122
- catalyst ⇒ 10.12
- orage ⇒ 4.8.0
- kismet ⇒ 2011-01-R1
- murrine ⇒ 2011-01-R1
- ntfs-3g ⇒ 2011.1.15
- vlc ⇒ 1.1.6
- openssh ⇒ 5.7p1
- memtest ⇒ 4.20
- inkscape ⇒ 0.48.1
- nmap ⇒ 5.10
- ddrescue ⇒ 1.14
- xz ⇒ 5.0.1
- mplayer ⇒ 1.0rc4
- xorg-xproto ⇒ 7.0.20

Improvements

- slitaz-configs-base added (4.0)
- slitaz-tools (4.0.2)
- slitaz-boot-scripts (3.4.3.1)
- slitaz-configs (4.1)
- slitaz-base-files (4.1.2)
- tazpkg (4.2.3)
- tazlito (3.3)

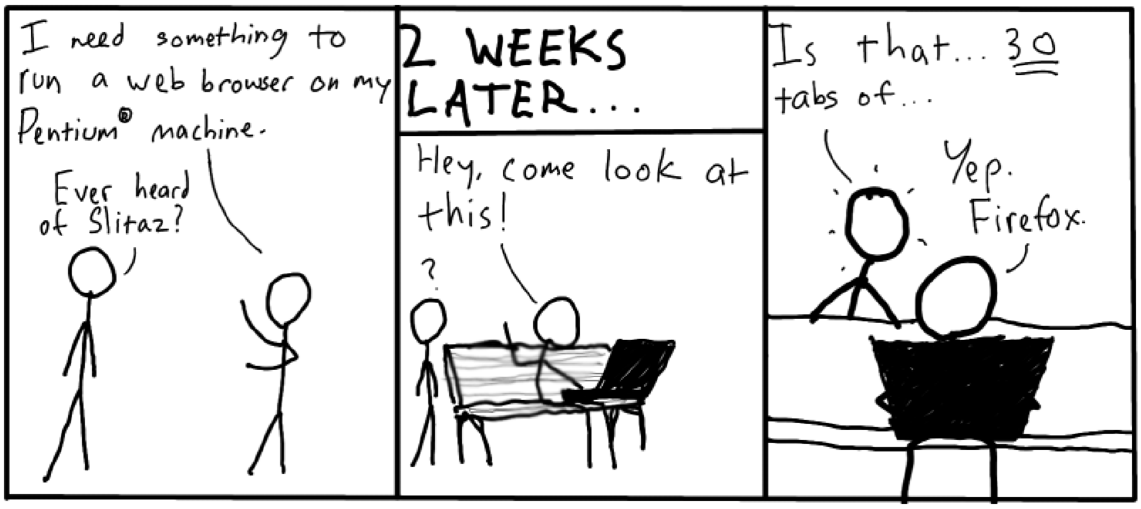
- tazchroot (0.0.4)
- libtaz (0.0.4)

Bugs

Activity	Open	Closed
Bugs	102	118
Features	44	43
Tasks	22	79

- Based on current figures

Cartoon



Tips and Tricks

- Python for Newbies – Part1[21]

Online

- Install the SliTaz operating system in VirtualBox[22] (broken link and also missing on web.archive.org[23])

1.10 Issue 23

author linea, godane

- Written on 2nd April 2011

[21] http://temporaryland.wordpress.com/2011/01/26/python-for-newbies/
[22] http://www.unixweblog.com/2011/01/installing-linux-slitaz-small-30-mb-linux-operating-system-in-virtualbox/
[23] https://web.archive.org/web/*/http://www.unixweblog.com/2011/01/installing-linux-slitaz-small-30-mb-linux-operating-system-in-virtualb

Latest News

- New Website, Community and Forum
- New Cooking 20110329 ISO released

New Packages

- nagios-{doc,dev,nrpe,nsca,plugins}
- p910nd
- emacs-pkg-{vala-mode,text-translator}
- opensp
- libofx
- libltdl
- mk-livestatus
- nagvis-{doc,flex}
- nconf
- ndoutils
- nareto
- libsvn
- perl-file-rsync
- bazzar-tools
- wbar2
- dokuwiki
- libshout
- perl-texi2html
- youtube-dl
- htmldoc
- unionfs-fuse
- dkms
- httptunnel
- ptunnel
- libtdb
- getmail
- tcpick
- ratmenu
- linux-libre
- deutex-devel
- freedoom
- par2
- backuppc
- firmware-iwlwifi-{1000,3945}
- rfkill
- ofono
- connman
- gtkaml
- lingot
- qca
- polkit-qt
- python-cython
- lfs-book

- e3
- fpc-{src,bootstrap}
- ruby-enterprise
- gource
- firmware-radeon
- metasploit
- gnu-netcat

Updated Packages (abridged)

- bluez ⇒ 4.90
- terminal ⇒ 0.4.6
- xfce4-* ⇒ 4.8.1
- gmime ⇒ 2.4.22
- thunar ⇒ 1.2.1
- vlc ⇒ 1.1.8
- midori ⇒ 0.3.3
- git ⇒ 1.7.4.2
- fribidi ⇒ 0.19.2
- uget ⇒ 1.6.2a
- babl ⇒ 0.1.4
- cherokee ⇒ 1.2.1
- sqlite ⇒ 3.7.5
- gnuchess ⇒ 5.08
- dialog ⇒ 1.1-20110118
- poppler ⇒ 0.16.2
- lirc ⇒ 0.9.0-pre1
- glpi ⇒ 0.78.2
- xchm ⇒ 1.18
- mercurial ⇒ 1.8.1
- gnumeric ⇒ 1.10.14
- gcc ⇒ 4.5.2
- coreutils ⇒ 8.10
- nspr ⇒ 4.8.7
- shared-mime-info ⇒ 0.90
- desktop-file-utils ⇒ 0.18
- xorg-xrdb ⇒ 1.0.8
- whois ⇒ 5.0.11
- stellarium ⇒ 0.10.6
- scite ⇒ 2.24
- mc ⇒ 4.7.5.1
- sylpheed ⇒ 3.1.0
- guile ⇒ 1.8.8
- pangomm ⇒ 2.26.3
- xorg-xf86-input-evdev ⇒ 2.6.0
- fotoxx ⇒ 11.02
- nss ⇒ 3.12.9
- mousepad ⇒ 0.2.16
- transmission ⇒ 2.22
- pidgin ⇒ 2.7.11
- clamav ⇒ 0.97

- wordpress ⇒ 3.0.5
- xorg-server ⇒ 1.9.4
- openssl ⇒ 1.0.0.d
- libwebkit ⇒ 1.2.7
- lyx ⇒ 1.6.9
- nmap ⇒ 5.51
- aspell-en ⇒ 7.1-0
- openldap ⇒ 2.4.24
- slang ⇒ 2.2.3
- lvm2 ⇒ 2.02.84
- util-linux-ng ⇒ 2.19
- snort ⇒ 2.9.0.4
- bazaar ⇒ 2.3.0
- irzip ⇒ 0.552
- emotion ⇒ 55225
- openjpeg ⇒ 1.4
- libsmpeg ⇒ 390
- PyQt-x11-gpl ⇒ 4.8.3
- libnet ⇒ 1.1.5
- sip ⇒ 4.12.1
- sane-backends ⇒ 1.0.22
- lmms ⇒ 0.4.10
- udev ⇒ 166
- scummvm ⇒ 1.2.1
- synergy ⇒ 1.3.6
- glew ⇒ 1.5.8
- partimage ⇒ 0.6.9
- clutter ⇒ 1.6.6
- ghostscript ⇒ 9.01
- busybox ⇒ 1.18.4
- cmake ⇒ 2.8.4
- gparted ⇒ 0.8.0
- virtualbox-ose ⇒ 4.0.4
- curl ⇒ 7.21.4
- ntp ⇒ 4.2.6p3
- valgrind ⇒ 3.6.1
- readline ⇒ 6.2
- bash ⇒ 4.2
- ruby ⇒ 1.9.2-p180
- ruby-gtk2 ⇒ 0.90.7
- bind ⇒ 9.8.0
- glibc ⇒ 2.13
- fluxbox ⇒ 1.3.1
- ario ⇒ 1.5
- mpg123 ⇒ 1.13.2
- phpvirtualbox ⇒ 4.0-3
- postfix ⇒ 2.8.2
- bluefish ⇒ 2.0.3
- tor ⇒ 0.2.1.30
- mono ⇒ 2.10.1
- orage ⇒ 4.8.1

- dnsmasq ⇒ 2.57
- samba ⇒ 3.5.8
- firefox ⇒ 4.0
- thunderbird ⇒ 3.1.8
- swig ⇒ 2.0.2
- sudo ⇒ 1.8.0
- m4 ⇒ 1.4.16
- iproute2 ⇒ 2.6.37
- wireshark ⇒ 1.4.4
- seamonkey ⇒ 2.0.13
- squashfs ⇒ 4.2
- python-pysqlite ⇒ 2.6.0
- subversion ⇒ 1.6.16
- mlt ⇒ 0.6.2
- tmux ⇒ 1.4
- nscd ⇒ 2.13
- qt4 ⇒ 4.7.2
- icu ⇒ 4.6
- samba-pam ⇒ 3.5.7
- jack-audio-connection-kit ⇒ 0.120.1
- gnome-mplayer ⇒ 1.0.2
- couchdb ⇒ 1.0.2
- freeciv ⇒ 2.2.5
- pan ⇒ 0.134
- sox ⇒ 14.3.2
- xorg-setxkbmap ⇒ 1.2.1
- emacs ⇒ 23.3
- mesa ⇒ 7.10.1
- enna ⇒ 0.4.1
- icedtea6-jdk ⇒ 1.9.7
- tar ⇒ 1.26
- openmpi ⇒ 1.5.3
- dosfstools ⇒ 3.0.11
- hydrogen ⇒ 0.9.5
- openvas-manager ⇒ 1.0.4
- openvas-administrator ⇒ 1.0.1
- gcompris ⇒ 9.6
- vala ⇒ 0.11.6
- yad ⇒ 0.9.1
- gnome-doc-utils ⇒ 0.20.4
- evince ⇒ 2.32.0
- ffmpeg ⇒ 0.6.2
- xorg-libX11 ⇒ 1.4.2
- aria2 ⇒ 1.11.1
- cairo ⇒ 1.10.2
- openal ⇒ 1.13
- xorg-xf86-video-ati ⇒ 6.14.1
- goffice ⇒ 0.8.14
- b43-fwcutter ⇒ 014
- rsync ⇒ 3.0.8
- mpd ⇒ 15.16

- task ⇒ 1.9.4
- gnutls ⇒ 2.12.0
- vim ⇒ 7.3
- gnutls ⇒ 2.12.0
- mlt ⇒ 0.7.0
- imagemagick ⇒ 6.6.8-10
- hplip ⇒ 3.11.3
- ghostscript ⇒ 9.02

Improvements

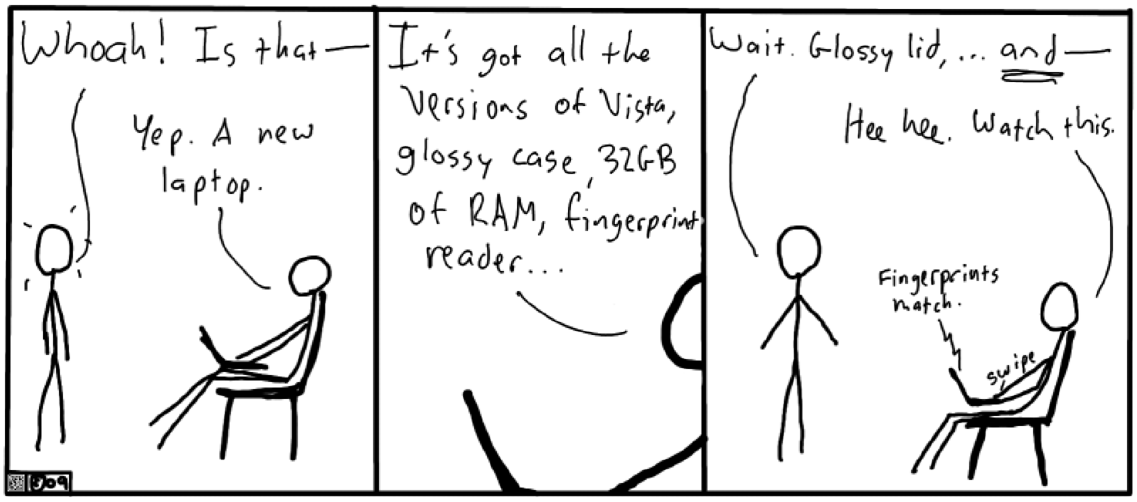
- slitaz-tools (4.0.3)
- slitaz-boot-scripts (4.2)
- slitaz-base-files (4.2)
- tazpkg (4.2.6)
- tazwok (4.2.9)
- tazlito (4.0)
- tazchroot (1.0.6)
- libtaz (1.0.4)

Bugs

Activity	Open	Closed
Bugs	65	77
Features	44	35
Tasks	9	81

- Based on current figures

Cartoon


Whoah! Is that—
Yep. A new laptop.
It's got all the Versions of Vista, glossy case, 32GB of RAM, fingerprint reader...
Wait. Glossy lid, ... and—
Hee hee. Watch this.
Fingerprints match.
swipe

Tips and Tricks

- Using the /proc filesystem[24]

Online

- Slitaz Linux 3.0 - An awesome 30 Mo Linux distribution[25]

1.11 Issue 24

author linea

- Written on 9th June 2011

Latest News

- New Italian and Danish Website (thanks yhouse, Allessandro, Giovani and Bo)
- New Cooking 20110531 ISO released

New Packages

- util-linux-ng-minix
- util-linux-ng-cramfs
- fuse-emulator
- libspectrum
- dega
- bcwipe
- tazweb
- cgames
- elfkickers
- arpwatch
- lensfun
- gens-gs
- tazdrop
- roundup
- firefox-langpack-pt-BR
- python-babel
- python-markupsafe
- postfix-mysql
- joe
- udev-light
- cookutils
- squidclamav
- squidguardmgr
- libnfnetlink
- libnetfilter_queue

[24] https://web.archive.org/web/20121127115059/http://www.petur.eu/blog/?p=320
[25] https://www.unixmen.com/slitaz-linux-30-an-awesome-30-mo-linux-distribution/

- diggerjs
- ii
- sic
- spruz
- file-tail
- fastthread
- activesupport
- activeresource
- activerecord
- actionpack
- actionmailer
- clamtk
- perl-carp-clan
- perl-bit-vector
- perl-date-calc
- perl-number-compare
- perl-text-glob
- perl-file-find-rule
- safecopy
- daemon_controller
- rack
- rake
- rails
- passenger
- glade-perl
- p0f
- usbmuxd
- cross-arm-binutils
- cross-arm-gcc
- pkcs
- truecrypt
- python-flup
- sylpheed-full
- nginx
- eeze
- xorg-server-light
- libdbusmenu-qt
- qemu-light
- slitaz-i18n-extra
- locale-sv
- locale-pt_BR-extra
- locale-es-extra
- locale-de-extra
- locale-da
- glib-networking
- xorg-server-Xfbdev
- xmlrpc-c
- firmware-rt2x00

Updated Packages (abridged)

- linphone ⇒ 3.4.3
- youtube-dl ⇒ 2011.03.29
- aria2 ⇒ 1.11.1
- filezilla ⇒ 3.4.0
- mercurial ⇒ 1.8.3
- fotoxx ⇒ 11.05
- iron-linux ⇒ 11.0.700.2
- yad ⇒ 0.12.0
- git ⇒ 1.7.5.2
- pyopenssl ⇒ 0.11
- python-django ⇒ 1.3
- python-pygments ⇒ 1.4
- python-jinja2 ⇒ 2.5
- lirc ⇒ 0.9.0
- xterm ⇒ 269
- pygobject ⇒ 2.28.3
- pygtk ⇒ 2.24.0
- glib ⇒ 2.28.5
- gtk+ ⇒ 2.24.4
- glade3 ⇒ 3.8.0
- dhcp{6} ⇒ 4.2.1-P1
- libgio ⇒ 2.28.5
- snort ⇒ 2.9.0.5
- xorg-xrdb ⇒ 1.0.9
- xorg-libX11 ⇒ 1.4.3
- nasm ⇒ 2.09.08
- cairomm ⇒ 1.10.0
- glibmm ⇒ 2.28.1
- pangomm ⇒ 2.28.1
- atkmm ⇒ 2.22.4
- gtkmm ⇒ 2.24.0
- pyneighborhood ⇒ 0.5.4
- gtk-gnutella ⇒ 0.96.9
- taglib ⇒ 1.7
- sudo ⇒ 1.8.1
- homebank ⇒ 4.4
- gpodder ⇒ 2.15
- swig ⇒ 2.0.3
- bluez ⇒ 4.93
- xfsprogs ⇒ 3.1.5
- mc ⇒ 4.7.5.2
- lftp ⇒ 4.2.2
- xfce4-panel ⇒ 4.8.3
- lilo ⇒ 23.2
- gobject-introspection ⇒ 0.10.7
- libburn, libisofs ⇒ 1.0.6
- tiff ⇒ 3.9.5
- json-glib ⇒ 0.12.4
- dmraid ⇒ 1.0.0.rc16-3

- sqlite ⇒ 3.7.6.3
- fakeroot ⇒ 1.15.1
- GConf ⇒ 2.32.3
- kismet ⇒ 2011-03-R2
- vala ⇒ 0.12.0
- polkit ⇒ 0.101
- extrema ⇒ 4.4.5
- vlc ⇒ 1.1.9
- firefox ⇒ 4.0.1
- thunderbird ⇒ 3.1.10
- syslinux ⇒ 4.04
- mpfre-dev ⇒ 3.0.1
- ejabberd ⇒ 2.1.6
- readom ⇒ 1.1.11
- grep ⇒ 2.8
- apache ⇒ 2.2.19
- postfix ⇒ 2.8.3
- apache-mod-perl ⇒ 2.0.5
- p7zip ⇒ 9.20.1
- dropbear ⇒ 0.53.1
- dbus ⇒ 1.4.8
- perl-libwww ⇒ 6.02
- virtualbox-ose ⇒ 4.0.8
- warmux ⇒ 11.04.1
- gphoto ⇒ 2.4.11
- kbd ⇒ 1.15.3
- kbd-busybox ⇒ 1.2
- weechat ⇒ 0.3.5
- xz ⇒ 5.0.2
- alsa-lib ⇒ 1.0.24.1
- alsa-utils ⇒ 1.0.24.2
- wxWidgets ⇒ 2.8.12
- wxpython ⇒ 2.8.12.0
- testdisk ⇒ 6.12
- parted ⇒ 2.4
- udev ⇒ 170
- sudo ⇒ 1.8.1p2
- qt4 ⇒ 4.7.3
- deadbeef ⇒ 0.5.0
- libsigc++ ⇒ 2.2.9
- feh ⇒ 1.14.1
- audacious ⇒ 2.5.1
- libgsf ⇒ 1.14.21
- pixman ⇒ 0.22.0
- transmission ⇒ 2.31
- midori ⇒ 0.3.6
- postfixadmin ⇒ 2.3.3
- groff ⇒ 1.21
- gstreamer ⇒ 0.10.34
- openvpn ⇒ 2.2.0
- acl ⇒ 2.2.51

- attr ⇒ 2.4.46
- libisofs ⇒ 1.0.8
- file ⇒ 5.07
- phpmyadmin ⇒ 3.4.0
- libcap ⇒ 2.21
- libevent ⇒ 2.0.11
- bind ⇒ 9.8.0-P2
- util-linux-ng ⇒ 2.19.1
- perl-datetime ⇒ 0.69
- gutenprint ⇒ 5.2.7
- lzo ⇒ 2.05
- libidn ⇒ 1.21
- mlt ⇒ 0.7.2
- openshot ⇒ 1.3.1
- gnuchess ⇒ 6.00
- libdvdread ⇒ 4.1.3
- openssh ⇒ 5.8p2
- ruby-gtk2 ⇒ 0.90.8
- ruby-gtk2 ⇒ 0.90.8
- pycrypto ⇒ 2.3
- perl-glib ⇒ 1.223
- perl-gtk2 ⇒ 1.222
- gnome-mplayer ⇒ 1.0.3
- zim ⇒ 0.52
- wordpress ⇒ 3.1.3
- ffmpeg ⇒ 0.6.3
- mono ⇒ 2.10.2
- mpg123 ⇒ 1.13.3
- espeak ⇒ 1.45.04
- ncmpc ⇒ 0.18
- ncmpcpp ⇒ 0.5.7
- scite ⇒ 2.25
- clutter ⇒ 1.6.14
- liferea ⇒ 1.7.5
- wine ⇒ 1.2.3
- mirage ⇒ 0.9.5.2
- pygobject ⇒ 2.28.4
- wireshark ⇒ 1.4.7
- mesa ⇒ 7.10.2
- libdrm ⇒ 2.4.25
- xorg-xf86-video-intel ⇒ 2.15.0
- rdesktop ⇒ 1.7.0
- rdesktop ⇒ 1.7.0
- postgresql ⇒ 9.0.4
- claws-mail ⇒ 3.7.9
- vte ⇒ 0.28.0
- glib ⇒ 2.28.7
- bazaar ⇒ 2.3.1
- qemu ⇒ 0.14.1
- ethtool ⇒ 2.6.38
- xfce4-settings ⇒ 4.8.2

- gnutls ⇒ 2.12.5
- wbar2 ⇒ 2.2.2
- vidalia ⇒ 0.2.12
- python-pytz ⇒ 2011g
- childsplay ⇒ 1.6

Improvements

- slitaz tools (4.2.3)
- slitaz-boot-scripts (4.4)
- slitaz-base-files (4.3)
- slitaz-configs (4.3.2)
- slitaz-doc (4.2)
- tazpkg (4.7)
- tazwok (4.2.14)
- tazlito (4.1)
- tazusb (3.0)
- tazweb (1.4)

Bugs

Activity	Open	Closed
Critical	7	3
Standard	15	9
Feature	5	1
Task	11	3

- Based on current figures

Tips and Tricks

- Case statement[26]

Online

- SliTaz GNU/Linux Cooking 20110531 Review[27] (video unavailable)

[26] https://web.archive.org/web/20110726060743/http://bashshell.net/shell-scripts/case-statement/
[27] http://www.youtube.com/watch?v=3jt3th_nhbE&NR=1

1.12 Issue 25

author linea

- Written on 28th July 2011

Latest News

- Wok currently frozen for RC series

New Packages

- idesk
- libmatchbox
- lsyncd
- dia
- ssfs
- ssfs-busybox
- luasocket
- zbar
- lxc
- iptraf
- keychain

Updated Packages

- mcabber-help ⇒ 0.9.10
- gob2-dev ⇒ 2.0.17
- ntop ⇒ 4.0.3
- ImageMagick ⇒ 6.7.0
- wakoopa ⇒ 2.0-2
- ncmpc ⇒ 0.19

Improvements

- slitaz tools (4.4.1)
- slitaz-boot-scripts (4.5)
- slitaz-base-files (4.4)
- slitaz-configs (4.4)
- slitaz-doc (4.2)
- tazpanel (1.1)
- cookutils (1.4)
- tazwikiss (1.3)
- tazdev (1.3)

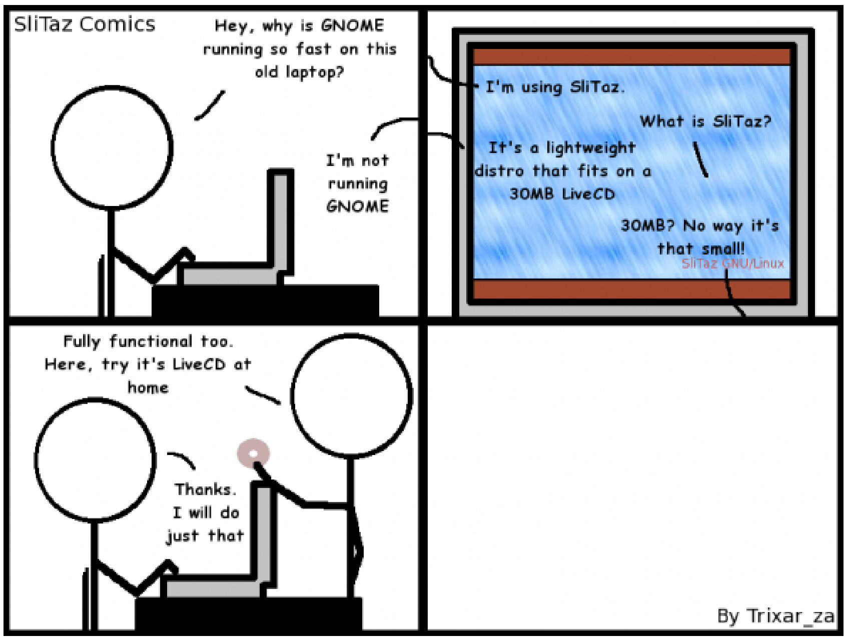
- tazdrop (4.4)

Bugs

Activity	Open	Closed
Critical	8	4
Standard	20	13
Feature	5	1
Task	11	9

- Based on current figures

Cartoon


SliTaz Comics
Hey, why is GNOME running so fast on this old laptop?
I'm not running GNOME
I'm using SliTaz.
What is SliTaz?
It's a lightweight distro that fits on a 30MB LiveCD
30MB? No way it's that small!
SliTaz GNU/Linux
Fully functional too. Here, try it's LiveCD at home
Thanks. I will do just that
By Trixar_za

Tips and Tricks

- Corntab[28]

Online

- What's cooking? It's SliTaz[29]

1.13 Issue 26

author linea

[28] https://web.archive.org/web/20110629014422/http://www.corntab.com/pages/crontab-gui
[29] http://distrowatch.com/weekly.php?issue=20110620#feature

- Written on 9th October 2011

Latest News

- Wok currently frozen for RC series

New Packages

- posixovl
- linmodem-slmodem
- hdparm
- opencc
- fcitx-googlepinyin
- libgooglepinyin
- laptop-mode-tools
- util-linux-ng-blockdev

Updated Packages

- dokuwiki ⇒ 2011-05-25a
- wordpress ⇒ 3.2.1
- iron-linux ⇒ 13.0.800.0
- xnviewmp ⇒ 0.39
- apache ⇒ 2.2.21
- wireshark ⇒ 1.6.2
- zim ⇒ 0.53
- fcitx ⇒ v4.1.2
- fcitx-googlepinyin ⇒ v0.1.3

Improvements

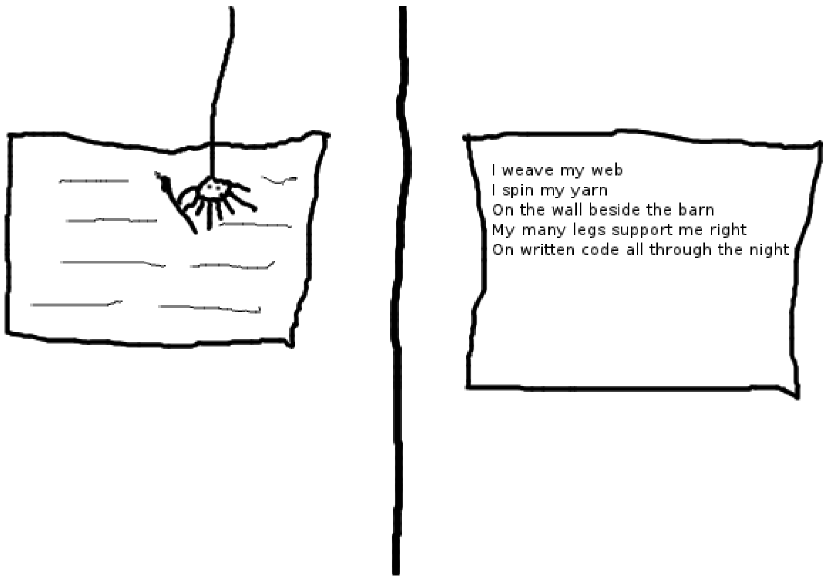
- tazwikiss (1.4)
- tazpkg (4.7.1)
- tazlito (4.2)

Bugs

Activity	Open	Closed
Critical	7	6
Standard	28	15
Feature	6	2
Task	11	9

- Based on current figures

Cartoon


I weave my web
I spin my yarn
On the wall beside the barn
My many legs support me right
On written code all through the night

Tips and Tricks

- Simple Iptables Script Generator[30]

Online

- The SliTaz Experience[31]

1.14 Issue 27

author linea

- Written on 20th January 2012

Latest News

- Various updates and packages added to repos

New Packages

- get-teamviewer
- get-wifica
- gsoap
- arping
- pciids
- usbids

[30] http://www.mista.nu/iptables/
[31] https://web.archive.org/web/20110805011916/http://rollingprogrammer.com:80/2011/06/26/the-slitaz-experience

- libdes
- gnome-games
- seed
- fbvnc
- novnc
- sdcc
- afpfs-ng
- gnome-desktop
- pamtester
- rlwrap
- qoauth
- python-xlib
- php-auth-pam
- util-linux-ng-flock
- tslib
- swfdec
- perl-net-pcap
- net-tools
- mspdebug
- avant-window-navigator
- apcupsd
- alarm-clock
- qca-ossl
- brasero
- blueman
- bleachbit
- blackbox
- billardgl
- ayttm
- collectd
- cdrtools
- catfish
- cairo-compmgr
- buildroot
- cssed
- cpuspeed
- cppunit
- compiz
- dovecot
- distccmon-gui
- directfb
- dietlibc
- dansguardian
- fbpanel
- ettercap
- enet
- editorbj
- gvfs
- guichan
- gliv
- ggseq

- get-msttcorefonts
- gejengel
- gecko-mediaplayer
- mana
- lxdm
- jnettop
- ipcalc
- icinga
- muninlite
- motion
- moosefs
- micro_proxy
- mongodb
- maxima
- protobuf
- oxine
- owfs
- ola
- nvclock
- ucarp
- tp_smapi
- thttpd
- t2
- soprano
- songwrite
- siproxd
- rsbep
- rdpdesk
- radiotray
- qtwitter
- wput
- wol
- wammu
- vsftpd
- verbiste
- mplayerplug-in
- ucl
- upx
- less
- libruby-extras
- nettle
- transmission-remote-gtk
- p11-kit
- gtk-doc
- mypaint
- protobuf-python
- python-distribute
- libraw
- libee
- libestr
- liblognorm

- libesmtp
- sagan
- barnyard2
- snort-mysql
- libhtp
- yaml
- libcap-ng
- suricata
- autoblog
- flatpress
- fwbuilder
- get-nz
- gtklp
- haproxy
- hardware-ibm-6272
- hardware-thinkpad-600e
- libextractor
- matchbox
- mp3gain
- pcmanfm2
- picoblog
- pwnat
- pysdm
- ranger
- ruby-gems
- shaarli
- tor-arm

Updated Packages (abridged)

- scite ⇒ 2.29
- python ⇒ 2.7.2
- snort ⇒ 2.9.1.2
- sudo ⇒ 1.8.3p1
- gtk+ ⇒ 2.24.8
- aria2 ⇒ 1.13.0
- bluez ⇒ 2.96
- bison ⇒ 2.5
- at ⇒ 3.1.13
- clamav ⇒ 0.97.3
- tcl ⇒ 8.5.10
- tk ⇒ 8.5.10
- tree ⇒ 1.6.0
- stellarium ⇒ 0.11.0
- audacious ⇒ 3.0.4
- inkscape ⇒ 0.48.2
- git ⇒ 1.7.7.2
- bazaar ⇒ 2.4.1
- bzflag ⇒ 2.4.0
- deadbeef ⇒ 0.5.1
- grep ⇒ 2.9

- py3k ⇒ 3.2.2
- putty ⇒ 0.61
- pciutils ⇒ 3.1.8
- usbutils ⇒ 004
- clamtk ⇒ 4.36
- pekwm ⇒ 0.1.13
- pcre ⇒ 8.13
- pangomm ⇒ 2.28.4
- pango ⇒ 1.29.4
- pcmciautils ⇒ 018
- openvpn ⇒ 2.2.1
- openssh ⇒ 5.9p1
- rsync ⇒ 3.0.9
- bind ⇒ 9.8.1
- bird ⇒ 1.3.4
- calcurse ⇒ 2.9.2
- claws-mail ⇒ 3.7.10
- cmake ⇒ 2.8.5
- dbus ⇒ 1.4.16
- dhcp ⇒ 4.2.2
- dialog ⇒ 1.1-20110707
- drupal ⇒ 7.8
- epdfview ⇒ 0.1.8
- espeak ⇒ 1.45.05
- fetchmail ⇒ 6.3.21
- gpodder ⇒ 2.19
- atkmm ⇒ 2.22.6
- gtkmm ⇒ 2.24.2
- youtube-dl ⇒ 2011.10.19
- xz ⇒ 5.0.3
- zsh ⇒ 4.3.12
- xpad ⇒ 4.0
- yad ⇒ 0.15.1
- xterm ⇒ 276
- wxpython ⇒ 2.8.12.1
- wxWidgets ⇒ 2.8.12.1
- wordwarvi ⇒ 1.00
- wget ⇒ 1.13.4
- vte ⇒ 0.28.2
- rtorrent ⇒ 0.8.9
- sqlite ⇒ 3.7.9
- lame ⇒ 3.99
- goffice ⇒ 0.8.17
- gnumeric ⇒ 1.10.17
- geany ⇒ 0.21
- glib ⇒ 2.30.1
- freetype ⇒ 2.4.7
- filezilla ⇒ 3.5.1
- acpid ⇒ 2.0.12
- cherokee ⇒ 1.2.101
- pcre ⇒ 8.20

- pam ⇒ 1.1.5
- gparted ⇒ 0.10.0
- mercurial ⇒ 2.0
- john ⇒ 1.7.8
- curl ⇒ 7.22.0
- tmux ⇒ 1.5
- firefox ⇒ 8.0
- gcompris ⇒ 11.09
- midori ⇒ 0.4.1
- lrzip ⇒ 0.608
- gnutls ⇒ 3.0.7
- tor ⇒ 0.2.2.34
- vidalia ⇒ 0.2.15
- bleachbit ⇒ 0.9.1
- htop ⇒ 1.0
- links ⇒ 2.4
- ncdu ⇒ 1.8
- seamonkey ⇒2.5
- postgresql ⇒ 9.1.2
- putty ⇒ 0.62
- wireshark ⇒ 1.6.4
- xfsprogs ⇒ 3.1.7
- ntop ⇒ 4.1.0
- vlc ⇒ 1.1.13
- scribus ⇒ 1.4.0
- p0f ⇒ 3.01b

Improvements

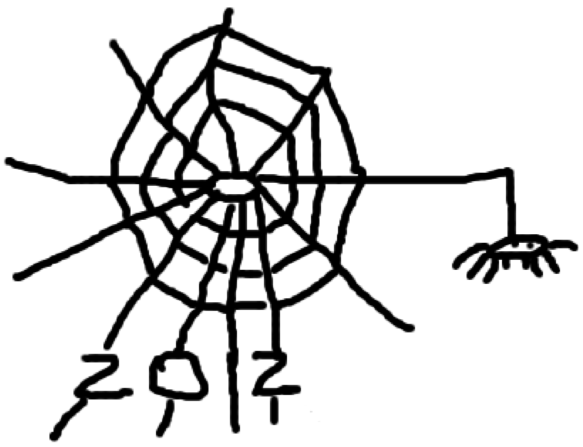
- tazpanel (1.2)
- tazpkg (4.7.2)
- tazwok (4.2.18)

Bugs

Activity	Open	Closed
Critical	6	9
Standard	25	27
Feature	7	3
Task	8	12

- Based on current figures

Cartoon



Tips and Tricks

- What is the exact difference between a ‘terminal’, a ‘shell’, a ‘tty’ and a ‘console’?[32]

Online

- Linux light - SliTaz GNU/Linux and SliTaz-Aircrack-ng[33]

1.15 Issue 28

author linea

- Written on 1st July 2012

Latest News

- SliTaz 4.0 released in April 2012

New Packages

- runcom
- xfe-extras
- memcached
- gdal
- proj
- geos

[32] http://linux-news.org/index.php/2011/10/19/what-is-the-exact-difference-between-a-terminal-a-shell-a-tty-and-a-console/
[33] http://all-things-linux.blogspot.com/2011/10/linux-light-slitaz-gnulinux-and-slitaz.html

- tsch
- Expect
- R language
- gummi
- pyradio
- dolibarr
- perl-parser
- perl-http-message
- perl-http-date
- perl-ecnode-locale
- perl-digest-md5
- perl-test-*
- SpaceFM
- c_icap
- xorg-xf86-video-fbdev
- gtk-engine-nodoka
- nmon
- perl-file-listing
- usb-modeswitch
- notification-daemon
- sarg-php
- msmtp
- pwauth
- mp3info
- libgee
- linux-arm-api-headers
- linux64-btrfs
- linux64 modules
- tea
- razor-qt
- upower
- xorg-libdmx
- xorg-dmxproto
- kmod
- sg3_utils
- libatasmart
- qiviewer
- lxappearance-obconf
- loqui
- lxpolkit
- lxisoftgun
- udisk
- xorg-server-Xephyr
- ziproxy
- softgun
- aalib
- xosd
- tuxpaint
- libpaper
- cross-x86_64-binutils
- linux-x86_64-api-headers

- cross-arm-glibc
- cross-arm-gcc
- qemu-box
- terminator
- cross-arm-binutils
- libzip
- httrack
- vlc-plugin
- xcb-util-*
- mariadb
- libconfig
- sshttp
- libqwt6-qt4
- enki
- dashel
- soundtouch
- lapack
- blas

Updated Packages (abridged)

- tor ⇒ 0.2.2.35
- apache ⇒ 2.2.22
- grsync ⇒ 1.2.1
- amule ⇒ 2.3.1
- sudo ⇒ 1.8.4
- xine-ui ⇒ 0.99.6
- glib ⇒ 2.32.1
- git ⇒ 1.7.10
- iron-linux ⇒ 17.0.1000.0
- privoxy ⇒ 3.0.19
- rrdtool ⇒ 1.4.6
- rubyripper ⇒ 0.6.2
- emacs ⇒ 23.4
- grep ⇒ 2.11
- firefox ⇒ 10.0.2
- midori ⇒ 0.4.6
- zsh ⇒ 4.3.15
- mtpaint ⇒ 3.40
- mtr ⇒ 0.82
- dillo ⇒ 3.0.2
- yad ⇒ 0.17.1.1
- hplip ⇒ 3.12.2
- swig ⇒ 2.0.4
- openssh ⇒ 5.9p1
- wireshark ⇒ 1.6.7
- tcsh ⇒ 6.18.01
- dokuwiki ⇒ 2012.01.25
- thunderbird ⇒ 10.0.2
- seamonkey ⇒ 2.7.2
- lighttpd ⇒ 1.4.30

- libpng ⇒ 1.5.9
- openssl ⇒ 1.0.1a
- libvorbis ⇒ 1.3.3
- squidclamav ⇒ 6.5
- graphviz ⇒ 2.28.0
- zim ⇒ 0.55
- nasm ⇒ 2.0.9.10
- yasm ⇒ 1.2.0
- poppler ⇒ 0.18.3
- libogg ⇒ 1.3.0b
- pidgin ⇒ 2.10.3
- transmission ⇒ 2.50
- mercurial ⇒ 2.1.2
- dovecot ⇒ 2.1.6
- tmux ⇒ 1.6
- jwm ⇒ 2.1.0
- gcompris ⇒ 11.12
- sarg ⇒ 2.3.1
- vala ⇒ 0.16.0
- logrotate ⇒ 3.8.1
- openjpeg ⇒ 1.5.0
- lxpanel ⇒ 0.5.9
- ndiswrapper ⇒ 1.57
- iptables ⇒ 1.4.13
- linux ⇒ 3.2.14
- mpc-library ⇒ 0.9
- glibc ⇒ 2.14.1
- linux-api-headers ⇒ 3.2.14
- gcc ⇒ 4.6.3
- binutils ⇒ 2.22
- mpfr ⇒ 3.1.0
- gmp ⇒ 5.0.4
- jpeg ⇒ 8c
- libevent ⇒ 2.0.17
- curl ⇒ 7.25.0
- bind ⇒ 9.9.0
- bazaar ⇒ 2.5.1
- asunder ⇒ 2.2
- aria2 ⇒ 1.15.0
- alsa-* ⇒ 1.0.25
- mjpegtools ⇒ 2.0.0
- gnupg ⇒ 2.0.19
- getmail ⇒ 4.26.0
- mc ⇒ 4.8.3
- xfsprogs ⇒ 3.1.8
- audacity ⇒ 2.0.0
- audacious ⇒ 3.2.3
- mono ⇒ 2.10.8
- mpg123 ⇒ 1.13.8
- subversion ⇒ 1.7.4
- sqlite ⇒ 3.7.12.1

- wxWidgets ⇒ 2.9.3.1
- file ⇒ 5.11
- lame ⇒ 3.99.4
- xorg-libX11 ⇒ 1.4.4
- dbus ⇒ 1.6.0
- dbus-python ⇒ 1.0.0
- gparted ⇒ 0.12.1
- parted ⇒ 3.1
- udev ⇒ 182
- polkit ⇒ 0.104
- pcre ⇒ 8.30
- busybox ⇒ 1.20.1
- vidalia ⇒ 0.2.17
- libdrm ⇒ 2.4.34
- xine-lib ⇒ 1.2.1
- Xorg server ⇒ 1.12.1
- apr ⇒ 1.4.6
- apr-util ⇒ 1.4.1
- linphone ⇒ 3.5.2
- libmtp ⇒ 1.1.1
- imagemagick ⇒ 6.7.6-7
- emelfm2 ⇒ 0.8.1
- dropbear ⇒ 2012.55
- ophcrack ⇒ 3.4.0
- gogglesmm ⇒ 0.12.6
- pciutils ⇒ 3.1.9
- broadcom-wl ⇒ 5_100_82_112
- elfutils ⇒ 0.153
- gimp ⇒ 2.8.0
- gegl ⇒ 0.2.0
- babl ⇒ 0.1.10
- vlc ⇒ 2.0.1
- ffplay ⇒ 0.10.3
- ffmpeg ⇒ 0.11.1
- tiff ⇒ 4.0.0
- libwebkit ⇒ 1.8.1
- libsoup ⇒ 2.38.1
- automake ⇒ 1.11.3
- htop ⇒ 1.0.1
- acpid ⇒ 2.0.16
- awesome ⇒ 3.4.11
- libffi ⇒ 3.0.11
- mplayer ⇒ 1.1
- bison ⇒ 2.5.1
- inkscape ⇒ 0.48.3.1
- qemu ⇒ 1.0.1
- gzip ⇒ 1.5
- gpodder ⇒ 2.20.1
- gutenprint ⇒ 5.2.8
- geany ⇒ 1.22
- php ⇒ 5.4.4

- jack-audio-connection-kit ⇒ 0.121.3
- grub2 ⇒ 2.00

Improvements

- cookutils (2.0)
- slitaz-configs (5.1)
- slitaz-boot-scripts (5.0)
- slitaz-doc (4.2.1)
- slitaz-icon (1.2)
- slitaz-menus (2.4)
- slitaz-tools (5.0)
- slitaz-tools-boxes (4.5.1)
- slitaz-base-files (5.2.3)
- tazweb (1.6.1)
- tazusb (4.2)
- tazlito (5.0)
- tazpanel (1.5.5)
- tazpkg (5.0)
- tazdev (1.5)
- tazbug (1.0)

Bugs

Activity	Open	Closed
Standard	28	14

- Based on current figures

Cartoon

Tips and Tricks

- Tiny Pluggable Linux ARM Computers Are Red-Hot[34]

Online

- SliTaz 4.0 Review — Small but fierce[35]

1.16 Issue 29

author linea

- Written on 1st December 2012

Latest News

- Website translated into Bosnian (thanks Adnan Sinanovic)

New Packages

- glm
- aircrack-ng-oui
- mediawiki
- pylorcon
- lorcon-old
- qemacs-tiny

[34] https://www.linux.com/tutorials/tiny-pluggable-linux-arm-computers-are-red-hot/
[35] http://badalhocando.blogspot.co.uk/2012/04/slitaz-40-review-small-but-fierce.html

- zoneminder
- vorbisgain
- turses
- tahoe-lafs
- shell-detector
- qtgain
- qemacs
- powermanga
- phpsysinfo
- nimrod
- lz4
- id3v2
- i3
- glusterfs
- fbgrab
- dulwich
- dar
- airstrike
- python-tweepy
- python-oauth2
- vkeybd
- freepats
- TiMidity++
- python-pyexiv2
- gigolo
- librsync
- iso-codes
- perl-dbd-pg
- pcsc-lite
- help2man
- defrag
- shake
- exfat-utils
- fuse-exfat
- dukto
- libQtDeclarative
- musl-libc
- autossh
- sratom
- lilv
- lv2
- serd
- sord
- udevil
- facter
- opus
- burp
- androguard
- ibus
- uemacs
- smx

- naim
- wepbuster
- perl-io-tty
- perl-io-stty
- perl-number-range
- perl-expect
- perl-algorithm-permute
- cmus
- icoutils
- get-playonlinux
- claws-mail-*
- partclone
- ipxe
- sozi
- check+
- iniparser
- giws
- hdf5
- szip
- libaio
- buffer
- mindi
- mondo
- grooms
- afio
- mtp-tools
- blazekiss
- python-prettytable
- python-mechanize
- python-html2text
- python-ooop
- gucharmap
- vpnc-cert
- gtk-recordmydesktop
- reaver
- pdnsd
- python-psutil
- linux64-aoe
- linux64-nbd
- mdbtools
- epdf
- gvolwheel
- pnmixer
- netrik
- nanoshot
- font-manager
- fbterm
- faenza-icon-theme
- 0install
- adminer
- blktrace

- busybox-boot
- blktrace
- codiad
- davmail
- e4rat
- fbff
- flashrom
- fsthost
- get-softmodem-driver
- lessfs
- libmikmod
- lvmts
- mikmod
- navit
- phpeasyvcs
- puppet
- qtractor
- splashutils
- systemd
- tokyocabinet
- vmware-view-open-client
- xflux
- pcsc-tools
- ccid
- cc65

Updated Packages (abridged)

- wpa_supplicant ⇒ 1.0
- busybox ⇒ 1.20.2
- shellfm ⇒ 0.8
- bison ⇒ 2.6.5
- git ⇒ 1.7.12.3
- atk ⇒ 2.4.0
- cairo ⇒ 1.12.2
- pango ⇒ 1.30.1
- phpmyadmin ⇒ 3.5.2
- libao ⇒ 1.1.0
- wordpress ⇒ 3.4.1
- drupal ⇒ 7.15
- mysql ⇒ 5.5.27
- logrotate ⇒ 3.8.2
- fail2ban ⇒ 0.8.7.1
- mercurial ⇒ 2.3
- iptables ⇒ 1.4.16.2
- e2fsprogs ⇒ 1.42.5
- wget ⇒ 1.14
- flex ⇒ 2.5.37
- mpg123 ⇒ 1.14.4
- samba ⇒ 3.6.7
- pcmanfm ⇒ 1.0

- freerdp ⇒ 1.0.1
- kmod ⇒ 11
- alsa-* ⇒ 1.0.26
- qemu ⇒ 1.2.0
- ncmpcpp ⇒ 0.5.10
- wine ⇒ 1.4.1
- dbus ⇒ 1.6.4
- iron-linux ⇒ 20.0.1150.1
- glib ⇒ 2.32.4
- nmap ⇒ 6.01
- wireshark ⇒ 1.8.2
- ardour ⇒ 2.8.14
- frogatto ⇒ 1.1.1
- zim ⇒ 2.56
- sudo ⇒ 1.8.6p3
- curl ⇒ 7.27.0
- ranger ⇒ 1.5.5
- clawsmail ⇒ 3.8.1
- xvkbd ⇒ 3.3
- sysstat ⇒ 10.0.5
- testdisk ⇒ 6.14
- patch ⇒ 2.7
- gawk ⇒ 4.0.1
- subversion ⇒ 1.7.7
- gparted ⇒ 0.14.0
- lua ⇒ 5.2.1
- midori ⇒ 0.4.7
- razorqt ⇒ 0.5.0
- ncdu ⇒ 1.9
- vpnc ⇒ 0.5.3
- ethtool ⇒ 3.6
- task ⇒ 2.1.2
- tmux ⇒ 1.7
- lzop ⇒ 1.03
- beautifulsoup ⇒ 4.1.3
- pciutils ⇒ 3.1.10
- syslinux ⇒ 4.06
- libfm ⇒ 1.1.0
- spacefm ⇒ 0.8.2
- udevil ⇒ 0.3.4
- devede ⇒ 3.23.0
- python-cython ⇒ 0.17.1
- get-flash-plugin ⇒ 1.4
- remminia ⇒ 1.0.0
- ffmpeg ⇒ 1.0
- audacity ⇒ 2.0.2
- openshot ⇒ 1.4.3
- xine-lib ⇒ 1.2.2
- grsync ⇒ 1.2.2
- enlightenment ⇒ 0.17.0-alpha2
- pcsc-lite ⇒ 1.8.6

- speedometer ⇒ 2.8
- units ⇒ 2.01
- sqlite ⇒ 3.14.1
- fetchmail ⇒ 6.3.22
- grooms ⇒ 1.05
- cryptopp ⇒ 5.6.1
- urxvt-full ⇒ 9.15
- gnome-icon-theme ⇒ 3.6.2

Improvements

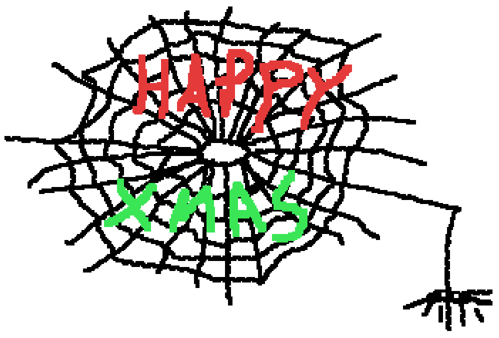
- slitaz-configs (5.1.2)
- slitaz-boot-scripts (5.2)
- slitaz-tools (5.1)
- slitaz-base-files (5.3.1)

Bugs

Activity	Open	Closed
Standard	33	31

- Based on current figures

Cartoon



Tips and Tricks

- Playing around with MPlayer[36]

[36] https://web.archive.org/web/20130117025157/http://tuts.pinehead.tv/2012/09/25/tutorial-playing-around-with-mplayer/

Online

- LinuxLive USB creator (Slitaz supported)[37]

1.17 Issue 30

author linea

- Written on 1st September 2013

Latest News

- System tool files translated into Greek and Polish (Thanks Constantine Mousafiris and Pawel Pyrczak)

New Packages

- mdocml
- gsynaptics
- get-xcas
- get-geogebra
- get-scilab
- get-algobox
- mtdev
- perl-dbd-sqlite
- cherrytree
- cocoalib
- gf2x
- giac
- ntl
- pari
- libacpi
- gputils
- firmware-rtlwifi
- xdigger
- xscavenger
- xbattle
- xorg-xhost
- ortp
- get-bitcoin
- gujin
- vice
- llvm
- fdupes
- otf-gfs
- ttf-inconsolata-dz
- ttf-mgopen
- ttf-pingwi-typography

[37] http://betanews.com/2012/08/27/linuxlive-usb-creator-now-supports-peppermint-os-three-archlinux-and-slitaz/

- ttf-open-dyslexic
- njconnect
- dd_rhelp
- firefox-langpak-it
- qutim
- libjreen
- lyx-locales
- xorg-libXaw3d
- xpaint
- libXaw3dXft
- netkit-rsh
- ttf-roboto
- croscore-fonts
- python-cssutils
- python-dnspython
- libwmf
- python-netifaces
- python-cssselect
- volumeicon
- wxHexEditor
- guvcview
- perl-md4
- pysolfc
- muparser
- raine
- alarm-clock-applet
- mp3val
- tcpreplay
- ckermit
- opencore-amr
- schroedinger
- orc
- freetuxtv
- avatar-factory
- yacpi
- libav
- neonview
- d-feet
- udisks
- libkeybinder
- n2n
- xpra
- fossil
- sockets
- libnfc
- linuxconsoletools
- python-pyweb
- w3m
- gv
- liberation-fonts
- transfig

- kexec-tools
- iperf
- extundelete
- libunique-gtk3
- gtk2-engine-murrine
- cantarell-fonts
- primateplunge
- perl-ipc-run
- perl-filesys-statvfs
- perl-lchown
- perl-unix-mknod
- perl-fuse
- lftpfs
- terminology
- dvtm
- linux-uml
- kriss_feed
- connman-tools
- python-evas
- python-elementary
- econnman
- python-phonenumbers
- py-asterisk
- coccinella
- cacerts
- compface
- tidy
- xbm2xface
- ttysnoop
- python-netaddr
- python-mock
- python-asterisk
- python-pywebdav
- get-litecoin
- miniupnpc
- get-src2pkg
- rkhunter
- lynis
- python-genshi
- fdutils
- ufiformat
- avfs
- anacron
- Clonezilla
- Drbl
- antinat
- libpano13
- manaplus
- php-docs-en
- xorg-libXxf86dga
- xorg-libXres

- xorg-libFS
- get-libaacs
- uml-utilities
- perl-cgi
- perl-config-general
- perl-http-server-simple
- perl-mailtools
- perl-mime-lite
- perl-mime-types
- monitorix
- libnl1
- python-ethtool
- xorg-xf86-video-cirrus
- xorg-xf86-video-mach64
- xorg-xf86-video-s3virge
- xorg-xf86-video-savage
- xorg-xf86-video-sis
- xorg-xf86-video-tseng
- apng2gif
- apngasm
- apngdis
- apngopt
- gif2apng
- libpng+apng
- quota
- sshguard
- iaxmodem
- gadmintools
- proftpd
- threaded-samba-scanner
- alien
- libcups
- tightvnc-java
- python-http-parser
- python-socketpool
- python-restkit
- python-six
- qupzilla
- gvpe
- xombrero
- jpegtran
- firmware-mod-kit
- python-magic
- locale-pl
- mosh

Updated Packages (abridged)

- deadbeef ⇒ 0.5.6
- sqlite ⇒ 3.7.16.2
- pidgin ⇒ 2.10.6

- urxvt ⇒ 9.16
- wine ⇒ 1.5.23
- udevil ⇒ 0.3.6
- spacefm ⇒ 0.8.4
- htop ⇒ 1.0.2
- xterm ⇒ 287
- sakura ⇒ 2.4.2
- audacious ⇒ 3.2.4
- gparted ⇒ 0.14.1
- gimp ⇒ 2.8.2
- mpd ⇒ 0.17.2
- bleachbit ⇒ 0.9.4
- vlc ⇒ 2.0.5
- binutils ⇒ 2.23.1
- atk ⇒ 2.6.0
- guvcview ⇒ 1.6.1
- gstreamer ⇒ 0.10.36
- gst-plugins-* ⇒ 0.10.36
- postrestql ⇒ 9.2.3
- emacs ⇒ 24.3
- lifera ⇒ 1.8.12
- libwebkit ⇒ 1.8.3
- nmap ⇒ 6.25
- linux ⇒ 3.2.40
- firefox ⇒ 17.0.8esr
- thunderbird ⇒ 17.0.8esr
- xine-ui ⇒ 0.99.7
- enlightenment ⇒ 0.17.4
- dropbear ⇒ 2013.58
- bluez ⇒ 4.101
- imagemagick ⇒ 6.8.4-10
- midori ⇒ 0.5.0
- dwm ⇒ 6.0
- geany ⇒ 1.23
- zlib ⇒ 1.2.8
- ophcrack ⇒ 3.5.0
- transmission ⇒ 2.5.2
- youtube-dl ⇒ 2013.05.23
- autoconf ⇒ 2.69
- automake ⇒ 1.13.2
- alsa-lib ⇒ 1.0.27.1
- git ⇒ 1.8.3
- python ⇒ 2.7.5
- php ⇒ 5.4.13
- cups ⇒ 1.6.2
- subversion ⇒ 1.7.8
- gtk+ ⇒ 2.24.18
- wicd ⇒ 1.7.2.4
- mesa ⇒ 9.1.3
- ntp ⇒ 4.2.6p5
- busybox ⇒ 1.21.1

- apache ⇒ 2.4.4
- squid ⇒ 3.3.4
- xfce ⇒ 4.10.0
- grisbi ⇒ 0.8.9
- tor ⇒ 2.3.25
- gpodder ⇒ 3.5.1
- gettext ⇒ 0.18.3
- awesome ⇒ 3.5.1
- fluxbox ⇒ 1.3.5
- valgrind ⇒ 3.8.1
- firmware-* ⇒ 20130610
- hplip ⇒ 3.12.11
- snort ⇒ 2.9.4
- dnsmasq ⇒ 2.65
- phpvirtualbox ⇒ (4.0-7)

Improvements

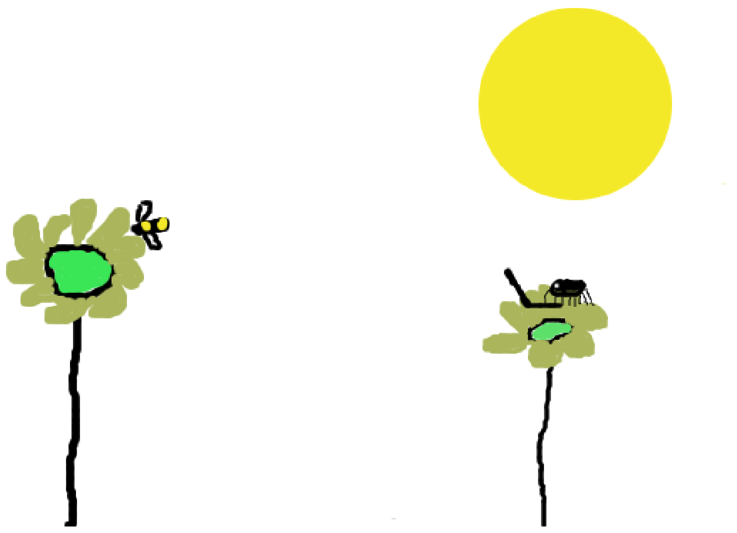
- slitaz-configs (5.1.5)
- slitaz-configs-base (5.1.3)
- slitaz-boot-scripts (5.3.1)
- slitaz-tools (5.4)
- slitaz-tools-boxes (5.4)
- slitaz-dev-tools (1.9)
- slitaz-base-files (5.4.1)
- slitaz-doc (4.4)
- tazpkg (5.1)
- tazbug (1.1)
- tazlito (5.1.1)
- tazpanel (1.6.1)
- tazusb (4.2.2)
- tazweb (1.6.4)
- cookutils (3.1.4)
- tazinst (3.90)
- tazinst-gui (3.90)

Bugs

Activity	Open	Closed
Standard	21	82

- Based on current figures

Cartoon



Tips and Tricks

- 60 Commands of Linux: A Guide from Newbies to System Administrator[38]

Online

- Slitaz's Notes — Alanyih[39]

[38] http://www.tecmint.com/60-commands-of-linux-a-guide-from-newbies-to-system-administrator/

[39] http://alanyih.blogspot.com/

CHAPTER 2

Guides

author erjo, pankso, linea, jozee, kultex, gokhlayeh, rupp, ernia, monz, seawolf, bellard, trixar_za, domcox, oui, emgi, sidini, brianperry, mojo, genesis, hgt, ceel

The SliTaz GNU/Linux quick start and complete guides by the community. You will find here specific guides to software, hardware or system configurations written by SliTaz users and contributors. These guides aim to provide concise and correct instructions. They explain a precise example of a solution and don't expect to be a general how-to.

General

- *Slitaz Newbie Guide* (page 83)
- *SliTaz Newbie Guide — II* (page 89) — Training for really Basic Newbie coming from Windows
- *README* (page 92) — Before you post a question in the Community Forum
- *Contributing to SliTaz* (page 93) — Guidelines to become a SliTaz Contributor
- *Most Annoying Bugs* (page 95)
- *FAQ* (page 95) — Frequently Asked Questions
- *Making tazpackages* (page 102) — Details on how to make packages
- *Testing Packages* (page 104) — Quality Management
- Artwork[40] — Showcase your screenshots, wallpaper, themes and other Artwork
- *Developers Corner* (page 505) — You are interested in contributing or developing your own SliTaz version? Here is your corner!
- *SliTaz Base Corner* (page 491) — A console mode SliTaz version is enough for you? See here what you may expect to do with it!

[40] http://scn.slitaz.org/groups/artwork/

Installation, flavors and virtualization

- *PXE boot* (page 104) — Boot from the web
- *Live CD for low RAM systems* (page 110)
- *Frugal install to Windows partition* (page 113)
- *Unusual install methods* (page 116)
- *LiveCD* (page 126) — How to make a SliTaz LiveCD
- *LiveUSB* (page 126) — How to make a SliTaz USB flash drive
- *Offline DVD* (page 129) — All Slitaz resources on one DVD
- *SD Card* (page 131) — How to install SliTaz to a MMC/SD flash memory card
- Online builder[41] — Create a flavor using Pizza
- *Create many-in-1 flavors* (page 133)
- Internationalization — Add language support and create your own language flavor
- *Qemu* (page 135) — SliTaz GNU/Linux with Qemu
- *VMware* (page 137) — SliTaz GNU/Linux with VMware
- *Virtualbox* (page 137) — SliTaz GNU/Linux with virtualbox
- *Net start usage and immediate installation on hard disk* (page 141)
- Grub2[42] — Install SliTaz with grub2 (and ext4)
- *Boot tree* (page 142) — What happens when you power on your slitaz box
- *Persistence and splash screen* (page 146) — Adding persistence and modifying the splash screen

Networking / LAN

- *Wifi easy* (page 148) — Easy way to configure wifi
- *Wifi hard* (page 154) — Hard way to configure wifi
- *Network.sh script* (page 157) — The /etc/init.d/network.sh file
- *VPN* (page 159) — Build your own Virtual Private Network
- *3G/Dialup* (page 161) — How to get your 3G modem working
- *Modem* (page 163) — Modem dialup with ppp or wvdial
- *Bluetooth* (page 166) — Configure bluetooth
- *Windows Networking with Samba* (page 169) — Share Windows/Linux files
- *Remote Desktop* (page 172) — NX software from NoMachine

[41] http://pizza.slitaz.org/
[42] http://blog.bodhizazen.net/linux/how-to-install-slitaz-with-ext4-and-grub2/

Software

- Recommended Software Packages — Applications to do almost anything on Slitaz
- *FlashInstall* (page 175) — How to install Adobe Flash Player
- *Super Fast Boot* (page 176) — Super fast boot time
- Burn CD / DVD — Burn CD/DVD audio/video
- Backup — How to make backups on CD / computer
- *Midori* (page 177) — Midori Tips and Tweaks
- Panels and Launchers
- Themes, Wallpaper, Icon sets
- Useful Fonts
- *Sexy Desktop* (page 185) — Desktop apps and config
- *Conky* (page 186) — Lightweight system monitor on desktop
- Audio Players
- Video Players
- Sound and Video Conversion — Sound/Video conversion, Audio creation and synthesis
- Import photos from digital camera — using gphoto2
- Dropbox
- *Tor* (page 188) — The Tor Install Guide without blah-blah
- *Conspy* (page 189) — Tiny session manager or shared console
- *Wicd* (page 192) — Lightweight network manager
- Grub and Grub2
- *Alsaequal* (page 192) — Real-time adjustable equalizer plugin for ALSA
- *Xbindkeys* (page 194) — Launch shell commands with your keyboard or mouse (independent of your X window manager)
- *TazWikiss* (page 198) — wiki software using a web server

Hardware

- *Xorg/Xvesa* (page 199) — What to do if X is will not start or has the wrong resolution
- *Nvidia* (page 206) — Install the latest non-free Nvidia driver
- ATI — Install the latest non-free catalyst ATI driver
- *Poulsbo* (page 208) — Poulsbo graphics installation
- *Webcam* (page 208) — How to get your webcam working
- *Printing* (page 209) — Howto print, CUPS install and configuration
- Scanning

- UDEV and HAL
- Troubleshoot
- *Power Management* (page 216) — ACPI (Advanced Configuration and Power Interface)
- *CPU Frequency Scaling* (page 217) — CPU Frequency Scaling instructions
- *UEFI and SliTaz* (page 220) — HOWTO do a *LiveUSB* (page 220) — *frugal install* (page 226) — full install in Boot Mode UEFI
- *Microcode* (page 232) — Updating the processor microcode

Server setup and system Administration

- Apache Webserver
- Lighttpd Webserver
- *Dropbear, Openssh* (page 235)
- Mailserver and Mailman
- *Database engine* (page 235) — SQLite and other DB management systems
- Redmine Setup
- Mercurial Installation
- *Drupal* (page 363) — Powerful CMS engine using PHP/MySQL
- Media Streaming Server
- File, Drive Encryption
- Firewall
- Router and Internet Gateway

Window Managers

- *DWM* (page 312)
- *Pekwm* (page 312)
- Fluxbox
- *Xfce* (page 312)

Kernel and SliTaz Build system

- *CustomKernel* (page 237) — Build your own custom Linux Kernel
- Linux packages — Explanation of various linux packages
- *Chroot* (page 238) — How to setup a chroot to cook packages
- *Tazwok tips* (page 240) — Tips for compiling with (the now deprecated) Tazwok
- *hal-removal* (page 241) — Documentation for removing hal in 5.0

2.1 General

2.1.1 Slitaz Newbie Guide

author oui, linea

2.1.1.1 SliTaz "the OS"

2.1.1.1.1 Introduction

SliTaz is an OS[43] available in many different actual versions and a lot of obsolete versions available for very old hardware possibly working better with old software as such. The actual versions include very small "text only versions" (just enough to start or fashioned fully to work in text mode) as well as starting directly in graphics mode. SliTaz always proposes a stable version of the OS as well as a version being in constant development in which the newest packages are to be found for testing purposes. The users have to decide which way they prefer: Graphics or not, and/or stable or testing.

2.1.1.1.2 Slitaz is free

The next consideration is how to get and (begin to) use SliTaz. As SliTaz is distributed directly, it is completely free (but not the non-free proprietory code downloadable from distributors using a download script recognizable by the name "get-something"; you may download these using the SliTaz script but you have to respect all of the possible conditions imposed by the owner), you can get SliTaz for no cost and do whatever you want with it. You can get it provided as is and all that you decide to do with SliTaz becomes your responsibility. Please be fair to the authors if you change something in his/her work or if you use it for other purposes as originally intended.

2.1.1.1.3 Get Slitaz

You will probably begin to use SliTaz by getting it online. You can also proceed differently by building SliTaz from Scratch. This way is also documented but probably rare among the fans using the distribution.

There are 3 ways to get SliTaz stuff online:

- download a SliTaz base distro directly from an internet network complete with a very short bootable application.
- download a Slitaz base distro online as a first step and then activate it using one or more steps.
- complete the environment by adding packages also downloadable online.

2.1.1.1.4 Support

Apart from software; SliTaz offers a complete scope of services:

- http://www.slitaz.org: information and news concerning the activity of the SliTaz community.

[43] http://en.wikipedia.org/wiki/Operating_system

- http://doc.slitaz.org: complete documentation on SliTaz Linux which realizes a specimen of Linux with all its main functions and without major concessions: Slitaz uses "root", "password" and "user" like most other serious Linux versions. All the stuff is detailed in the SliTaz Handbook. It can be considered as a short, but complete reference for all Linux beginners (especially if they do not work with SliTaz on a daily basis). The SliTaz doc also includes Guides being more orientated towards SliTaz and specific tasks.
- http://forum.slitaz.org: discussion point for the SliTaz community in many languages.
- http://scn.slitaz.org: SliTaz Community — read more at the web page and look on the top right hand side of the page to discover yourself the other branches of the community. Please note the IRC[44] channel as you can often get help directly on it!
- http://mirror.slitaz.org/iso: SliTaz Mirror — where you will find a lot of ISO files to start your adventure with Slitaz.

2.1.1.1.5 Activate SliTaz

By using a Slitaz CD (offers a web boot as well as a traditional ISO boot) or a Slitaz floppy (web boot only), there is not much to do! If a prepared Slitaz CD or Floppy is in the CD/Floppy drive before the PC starts, just start the PC and follow the instructions on screen!

Before you start, you can also insert an USB memory stick with a prepared file system "ext3" into one of your USB connectors. This automatically will be the place where your data is stored (Linux users say "home").

Nothing else could be more easy!

If this kind of installation can't be used, SliTaz offers a lot of other solutions adapted to other starting problems and wishes!

The next chapter will introduce you to something else!

2.1.1.2 Start SliTaz with Grub2!

2.1.1.2.1 Master Boot Record (MBR)

In each PC, at least one of the drives (if more than one) is the Master and contains the MBR where the initial startup following each shutdown is defined. In probably most PCs this special record is occupied by Microsoft's starting software. This software defines usually that the PC can only start Microsoft- or IBM-DOS or -Windows (only rarely do PCs propose a real Dual Boot).

It is not possible to boot with the Microsoft software directly into another OS!

But it is possible to rewrite that software, a BootLoader can do it (and over that preserve eventually the ability to boot Microsoft- or IBM-DOS or -Windows or free or not clones like freeDOS[45] or ReactOS[46] or more than one OS).

[44] http://webchat.freenode.net/?channels=#slitaz
[45] http://www.freedos.org
[46] http://www.reactos.org

2.1.1.2.2 Grub2 is the GNU actual standard boot loader

SliTaz holds Grub2[47] in its repositories. If not already available on your system you can install it by entering the following command in the console providing you have a valid Linux partition with a `/boot` directory (both can be empty):

```
# tazpkg get-install grub2
```

Continue in the command line (to install **`grub`** to your MBR) with:

```
# grub-install /dev/sda
# grub-mkconfig -o /boot/grub/grub.cfg
```

This main way of Grub2 installation happens generally **after** the installation of SliTaz itself (in the case of SliTaz or of the other linuxes if installed by another distribution).

Let's speak on that now?

Because it is the first step of what happens using SliTaz before Slitaz starts itself! Also bootable CDs or floppies contain their own little different boot loader. Perhaps you have Grub2 already pre-installed from a preceeding OS installation? In this case you only have to adapt your Grub2 installation! And you can start to directly use Slitaz without some CD or Floppy at the next start! If not, you just need to start Slitaz once through another way (CD, Floppy, USB installation etc.) if needed, adapt the partition of your hard disk, install Grub2 on the target partition, and continue...

And you will see in the next chapter that Grub2 is very flexible and doesn't need any OS installation at all. It can start ISO files or frugal OS versions, as well as any full installations.

More! The new Grub2 does not need a valid configuration at all. It includes a tiny command interpreter and it would be possible to start your system step by step using individual commands if you knew perfectly well how to do that and which parameters you have to enter (it is probably the best way to make your system inaccessible to all other people except real freaks!).

2.1.1.2.3 Adapt a Grub2 installation

Warning: There are 2 usages for Grub2.

Some distros (for example Debian, Ubuntu, etc.) use an extended one and require from you that you don't manually edit yourself the /boot/grub/grub.conf file! If you continue to use these distros, please read the instructions given in the docs for that distro because that distro will probably automatically overwrite all the changes you made manually if you don't respect their prescribed way!

Some distros have a simple usage of Grub2 and allow the direct adaptation of the text file /boot/grub/grub.conf. In this case you can adapt your Grub2 installation by changing the text in it. This is the configuration file for Grub2!

2.1.1.2.4 Typical contents of Grub2 as used in SliTaz

I have 3 active Slitaz versions on my hard disk. The first one, an actual *base version* (full operable, in console mode only), release 4.0, only about 7.2 Mb, will be started directly as an ISO file without any

[47] http://www.gnu.org/software/grub/

other preparation:

```
menuentry "SliTaz, ISO start, on /dev/sda5" {
set root=(hd0,5)
set isofile="/slitaz-4.0-base.iso"
loopback iso $isofile
linux (iso)/boot/bzImage from=$isofile ramdisk_size=6666 home=usb root=/
↪dev/ram0 rw autoexec=startx changes=s512.dat
initrd (iso)/boot/rootfs.gz
}
```

The second was prepared by using a concatenated file rootfs.gz (its original ISO file did contain the split file system) and is a classic "frugal"!:

```
menuentry "SliTaz, frugal, on /dev/sda5" {
set root='(hd0,5)'
linux /slitaz/bzImage pdev1=sda5 psubdir=slitaz
initrd /slitaz/rootfs.gz
}
```

This last one is for a full installation:

```
menuentry "SliTaz, full, ext4, on /dev/sda3" {
linux (hd0,3)/boot/vmlinuz-3.2-slitaz root=/dev/hda3 ro vga=normal
}
```

Note: Take care that all file names and paths are real and correctly written! Take care to write CamelCase[48] names that linux users love to use! Esp. for example in the name *bzImage*!!!

2.1.1.3 Training

I will now propose some *beginner training*.

2.1.1.3.1 Base training

Starting SliTaz base

4 ways at your disposal:

- burn the ISO on a fresh CD as an ISO (special burn mode) and start from the CD.
- download / copy the ISO into the root of a partition of your hard disk and start with Grub2.
- burn an ISO or use an old SliTaz CD: which still contains all of the CD as well as the net install option to make use of a net start of SliTaz base.
- install the net install floppy image on a floppy and make a net start using SliTaz base.

During this training, we will not install!

The CD start has a slight advantage: you can pass parameters (see CD messages: a short time after starting the CD will invite you to enter parameters and indicate to you how to see which parameters are

[48] http://en.wikipedia.org/wiki/CamelCase

available). But it has a great inconvenience: it is a dirty way to produce an ISO needing a CD for only 7 Mb irregardless of space (try perhaps if possible with a CD-RW)!

Two little menus will then appear during the start phase concerning language and keyboard.

The login appears.

Login

Yes, in SliTaz, you have to login! But the SliTaz developers have taken care to make it easy for you.

You can chose one of two ways:

- enter as user (recommended as a good habit). The standard user in our ISO is *tux* and does not need any password for SliTaz (hit only *enter* at the *password* prompt).
- enter as *root*. In linux the system administrator as well as their home directory is named *root*. Don't confuse both! If you directly enter as system administrator, login the system as *root* and repeat *root* in the *password* prompt.

Note: nobody recommends entering a linux system as *root*, it is a really a bad habit!

First steps

As the notorious proprietary OS Unix was cloned into Linux really full graphics systems were not that frequent! But it was already possible to realize in console mode extensive computer operations. Linux did inherit its wide possibilities. SliTaz proposes a very neat and powerful as well as concise environment in console mode.

2.1.1.4 Really basic newbies training

People coming fresh from Windows will probably be surprised by Linux. For this reason, I wrote some *extra training* (page 89) for them.

- You can jump this part -

2.1.1.5 Daily commands

As this page is already long, please continue *here* (page 87)!

2.1.2 Daily used Linux commands

author oui, linea

After starting our SliTaz base (see *here* (page 86)) we logged on and can see now, following a Welcome message, a short command line prompt:

```
tux@dolilive:~$ _
```

Now we can immediately start to understand who is doing what:

- **tux**, is my login name
- @, is where I am "at"
- **dolilive**, something *like do_Linux_live*
- **:**, what we chose to do is awaiting us at the prompt and it will be actived here (in absence of some concrete object that would redefine the destination)
- **~**, is my «home» (**~** is the symbol for the home directory of the user logged in!)

Warning: **~** depends on the administrative rights you actually have

If you are *root*, any data will be automatically registered in the special directory */root*. The tree of the main directory is also named the *root of the file system*. The symbol / is the first sign that you see alone or in the first place of all the other paths! It is not possible to unbind the *root of the file system* and the main directory *root* is used by the administrator *root* for all of their transactions! This directory */root* is a special directory as for example as it has to be in the same partition as the tree of main directories as well as hold the contents of the first level!

The *home* of users is a subdirectory in the main directory */home*! The path is as follows: `/home/tux`

tux can be defined differently at the installation if you don't want to be called tux! In such a case the path would be: `/home/`*`SomeWhatYouWillBe`*. Both the contents of home, as well as the contents of *`SomeWhatYouWillBe`* can also be contained in a separate partition. The automatic installer in the SliTaz Panel can help you to do that if you wish.

We can enter now at the prompt our first command:

2.1.2.1 ls

You don't need to exit

ls is a command to list files located in the actual directory. Adding parameters and/or paths it can do that in each directory where you have the rights and can do more than just showing the file names.

Please test:

```
ls --help
```

(more? see here[49])

And test now please:

```
ls /usr/bin
```

2.1.2.2 See Also

The SliTaz Handbook (page 273)

[49] http://en.wikipedia.org/wiki/Ls

2.1.3 SliTaz Newbie Guide — II

author oui

Training for really Basic Newbie coming from Windows

Linux is not only the successor of Unix: it was the clone of that already perfectly developed OS! For this reason did Linux never need to copy Windows stuff or methods and did develop itself without getting distracted from its own objectives. And all documentation concerning Unix and, later, Linux was already perfect as Linux came, so that very old documents can be deprecated but already very useful, especially translations into other languages as they often can't be published as soon as new developments.

2.1.3.1 Please remember

The **main** difference is, that Linux did never break with the use of command line: All graphic figures can be considered as (often only) graphic front ends from a powerful unified processing at commando line (=binary) level. All is considered as a "file" and will be handle so, so that the commands can generally be used on each type of objects and data.

The **second** difference is, that old version from Unix and Linux also did strongly manage the owners of all objects and data (the initial Windows did miss this performance!) so that you very often have to take caution in Linux don't to hurt the ownership of objects and data: The system refuses you access as well as processing!

Third difference is, that Linux doesn't "*reinvent the wheel*". As the system is now generally free and open source, developers can use freely the work of predecessors if it is good, fast and secure, and include such works into her new development without to need to recode somewhat! We are speaking from the as terrible considered "*dependencies*"! The commando line use of Linux meets this tendency: An application can be only a script in which programs are invoked and parameter passed and a lot of them are really so! But the consequence is of course, that when the dependencies are missing, the application does not work at all.

If you remember that you will probably avoid some difficulties to accustom in Linux coming from Windows!

2.1.3.2 Some good habits

- Accept with realism to use the commando line where it is better (and it is often better for a major reason: you see in your monitor after that messages about your actual process; even for beginners can that be really precious and avoid long and unnecessary research!)
- Never login as "*root*": it is not locked as some people are really pure administrators and need that access but it is the opened door for all kind of danger for your data and your system
- Install as beginner all dependencies even if they seem incomprehensible, improbable and memory-intensive!

For this reason will now look at

2.1.3.3 some commando line transactions

(what else after above recommandations)

2.1.3.3.1 exit

As I resented often about the difficulty to find a way to leave an application (a ridiculous situation: The application make you prisoner!), I will try to begin my explanations immediately with this important figure!

- **exit** is a main commando name to leave a superordinated transaction. It ends for example the session, and, for people starting linux without *"sessions manager"* directly ends the graphic phase of processing to return from Xwindow back to the console level

In professional linux versions, you can generally not do more than close your session — You can not leave (reboot) or halt the system. And I find it is good so, because it can be possible that the administrator did connect the command of an important object (pump that avoid, that the basement of the house be flooded by water infiltration, or your internet telephone box, nobody can call you any more, etc.). You don't know that as simple user! Thousand reason can prohibit you stop a computer...

In home computer it would be to much and the distributor did generally include (as commissioned by the administrator, private administrator are not always accustomed in such rare jobs) more easy way to do that for each. SliTaz as distributor offers to administrator or user following commands (plus, of course, special virtual keys on the desktop or menu items in the graphic environment):

- **reboot**
- **halt**
- `Ctrl+Alt+Delete`

Sometimes the password of the administrator *root*, **root** will be required. In SliTaz 1,0 and graphic session, you have to enter one of those words in the main line of the session manager (User name line) and enter the password **root** into the password field!

In the applications, there is an awful mess!

- **quit**
- **:quit**
- q
- `Alt+q`
- `M`

etc. are variants coming in consideration (and used by applications in the Slitaz Base training !)

2.1.3.3.2 su

Command parameters

Precedent commands are used without parameter in above cases. A lot of commands must have parameters plus predefined objects. For example:

```
SomeCommand --SomeParameter [ [SourceObject] DestinationsObject]
```

Tip: Some commands support more than one parameter at the same time!

Parameter –help

A great number of commands is offering an Help parameter:

```
SomeCommand --help
```

or shorted

```
SomeCommand -h
```

If you try that with the command **su** you will see a very helpful parameter **-c**:

su -c 'CommandToPass'

With this combination of the command `su -c` followed by an other command between to ' you can execute some commands reserved for the administrator *root* (the same thing does also the separate non preinstalled command `sudo`; if you prefer `sudo`, you have to install it and eventually adapt the parameter file `/etc/sudoers`. Using `su -c`, you have to reenter each time the root password. Using `sudo` only at the first use in the same session!)

```
$ su -c 'SomeWhatBeeingOnlyAble --by root'
```

You will be invite to enter the valid password from **root**! The command is in that form only for the entered line valid. You don't need to exit!

su (without any parameter, or eventually with:) [-] [user]

Warning: exit this phase of processing as soon as possible with the command `exit`!

`su` opens a sequence of following commands giving you the right from the user, if you name an user, or **root**. I have to enter the adequate password.

With those commands and variations it is easy to respect the rule never login as *root*!

man SomeCommand

Tip: quit the manpage with **q**

Will show you the manual page about the command SomeCommand. Please try

```
$ man su
```

and read it with attention!

2.1.3.4 Daily used commands

Please continue *here* (page 87)!

2.1.4 README

author kultex, linea, seawolf, trixar_za, genesis

Yes we love your questions, your problems are ours and SliTaz has a very good Support Team, **"but sometimes it's hard"** for us when you forget to use the search button or read the official documentation[50].

Answering questions takes time, and to answer them well slightly longer. This time could be better spent for other necessary things, like improving SliTaz or writing documentation.

2.1.4.1 Few things to note before asking for support

1. Please provide us with as much information as possible

Telling us that something is broken doesn't help to solve anything, because it may run perfectly OK on our systems. This means you'll have to provide us with some more information so that we can help you. This will include the steps you took before and after installing the program, your current systems specifications and which version of SliTaz you happen to be using. If a program fails to open, try running it in a terminal and copying down the error. If a piece of hardware doesn't work, see what **`lsusb`** (you may need the **`usbutils`** package), **`lspci`** and **`dmesg`** commands have to say when you plug in the device. The more information you can provide us with, the quicker we can solve your problem.

2. SliTaz is not Windows or Ubuntu

If you're expecting an easy **`Windows`** or **`Ubuntu`** replacement in a 30MB ISO, then you'll be sadly disappointed. SliTaz was designed to be lightweight, fast and small so it can run on older hardware. This means a lot of the larger packages have been stripped out to save space. It is possible to make SliTaz act and look like either **`Windows`** or **`Ubuntu`**, but don't expect every kind of functionality to work out of the box. Remember, even with **`Ubuntu`** and **`Windows`** you still have to install some other stuff before you can use them. If you wish to use a Linux distribution that works out of the box, give **`Linux Mint`** or **`PCLinuxOS`** a go. Just as a little warning — someone in the forum described SliTaz as "a no-go for newbies".

3. SliTaz isn't based on any other distribution

SliTaz was made from scratch using the **`Linux Kernel`**, **`Busybox`**, **`OpenBox`** and some **`LXDE`** tools. It has its own package management system and most of the packages are made by volunteers. SliTaz does however boast the ability to convert packages from other distributions using the **`tazpkg convert`** command.

4. We're unpaid volunteers and not your employees

Remember that you haven't paid the developers who designed SliTaz or the people online who provide the support. They don't owe you anything. So ask nicely and we'll try to help you in due time.

Here are some links to the most frequently asked questions:

[50] http://doc.slitaz.org/en:start

“Can’t login, not even as root” or “startx does not work” or “failed to execute login command”

http://vanilla.slitaz.org/index.php?p=/discussion/comment/5812/#Comment_5812
http://doc.slitaz.org/en:guides:xorg-xvesa

“How to change the desktop resolution”

http://doc.slitaz.org/en:guides:xorg-xvesa

“Adding a new user”

http://vanilla.slitaz.org/index.php?p=/discussion/comment/2808/#Comment_2808

“Copy / paste troubles with xterm”

Install sakura terminal as root:

```
tazpkg get-install sakura
```

See also

Frequently Asked Questions (page 95)

2.1.5 Contributing to SliTaz

author seawolf, claudinei, linea, pankso, domcox

There are so many ways you can help out with the SliTaz community. If you have a little free time, anything from artwork to package testing would be greatly appreciated!

Post a quick message to the forum or mailing list if you’re unsure about anything. We don’t bite!

Tip: We have a Community Network[51] where you can write blog posts, share wallpapers and images, and more. We encourage anyone that contributes to SliTaz to use this platform to show off their hard work!

2.1.5.1 Developers & Hackers

Packaging

If you are comfortable with the Linux compiling process (**`configure`**, **`make`**, **`make install`**), you can help by creating SliTaz packages. We have an automated tool to build packages from source

[51] http://scn.slitaz.org/

called Tazwok[52]. All SliTaz packages contain a *receipt* (page 370) — a text file where you control the building process using some variables and functions. You can browse the wok[53] to see some receipts and understand how they work. Please take time to read the Developer's page beforehand and practice locally with some new or existing packages.

When you feel ready to start cooking, read the packages cooklist[54], pick one and test. You can submit your receipts to the Mailing List or contact a SliTaz dev, so that we can review the work and give you access to the repos.

Package Testing and Debugging

You can help SliTaz by testing packages and reporting bugs on the Bug Tracking System[55]. We have some Package Testing Guidelines[56]. You can use the testing project on the Labs, the forum or the mailing list to report any missing dependencies, unexpected package behavior, or wrong configurations, etc.

2.1.5.2 Writers & Translators

Take a look at the *SliTaz Documentation Guidelines* (page 244), pick your page and go! At this stage, we just need solid contributions which will then be reviewed and updated. We're undergoing a Summer of Documentation[57], in which we're concentrating on all the docs. You'll be in good company.

There's many areas in which you can help:

Are you experienced with SliTaz? Go for the Guides. They take users through a process and range from customising your desktop to getting networking set-up. If you've had success with a procedure or feel one could be improved, this is the place for you!

Are you a general Linux user? Take a look at the Handbook. A lot of general introduction to SliTaz, Linux or available software can be written there. Some people find it difficult to write technically, others struggle with more generically informative writing. Both are needed, in describing the topic and scope before brief instructions.

Do you speak non-English? Translations are very important to the global audience SliTaz attracts. We need speakers of the above languages to ensure non-English readers experience better translations than automated services can provide! Those are the locales supported by the SliTaz system. We are aiming to add more in the future — if you can help, please do!

2.1.5.3 Artists & Designers

Wallpapers[58] make a huge impact on the first impression users receive. Some reviews barely comment further on artwork than the wallpaper!

[52] http://doc.slitaz.org/en:cookbook:wok
[53] http://hg.slitaz.org/wok
[54] http://labs.slitaz.org/projects/distro/wiki/Pkgscooklist
[55] http://bugs.slitaz.org/
[56] http://labs.slitaz.org/wiki/packages
[57] http://listengine.tuxfamily.org/lists.tuxfamily.org/slitaz/2010/04/msg00063.html
[58] http://community.slitaz.org/image/tid/2

2.1.5.4 PR & Journalism

(Could we put together some blurbs or short pieces of writing for the press & community? Maybe contacts for them?)

2.1.6 BUGS

author stork, linea

The problems listed in this page refer to the Slitaz 3.0 environment.

2.1.6.1 Burning a CD with `Burnbox`

To burn a CD, you have to be root (it will NOT work from your default TUX account, even if TUX has been added to the `cdrom` group).

In **`Burnbox`**; enclose the file and folder names between quotes if they contain a space (eg `"My Documents"`), otherwise it will fail to find the document.

2.1.6.2 `Abiword`

With **`AbiWord`** you MUST install **`glibc-locale`** too, it was forgotten as a dependency, but it is needed for **`AbiWord`** to work properly.

2.1.6.3 Chat entry of the Internet menu

This should install **`pidgin`** and **`skype`**. Practically it has sometimes (but not always) installed **`pidgin`**, and always failed to install **`Skype`**. Install **`Pigin`** and **`Skype`** (well, **`get-skype`**) from the package manager.

2.1.7 Frequently Asked Questions

author seawolf, linea, jozee, genesis, hgt

Listed here are the most frequently asked questions from the user forums. Please check here to see if your problem has been fixed before starting a new forum thread.

These are organised into per-scenario groups — from system-level through to the desktop. Within each, a symptom describes a give-away identification of the problem; a widely-working solution is then outlined.

General

- *Log-In Problems* (page 97)
- *Shutdown via SLiM* (page 98)
- *Terminals (Copy/Paste etc.)* (page 98)

Package or Software

- *An Application Cannot Find Libraries/Files* (page 99)
- *A Program Doesn't Run or Quits Unexpectedly* (page 100)
- *Can I install a package on SliTaz from another distribution?* (page 100)

Hardware

- *Mouse* (page 101)

Questions Still Unanswered?

No problem! We are happy to hear your question over at our Support Forum[59] — or even the Mailing List[60] if it is a more lengthy discussion — once you have looked through our documentation[61]!

Answering questions takes time however, to answer them well slightly longer. This time could be better spent for other necessary things, like improving SliTaz or writing documentation.

Your first post should include the following things:

- detailed information about your hardware
  - e.g the manufacturer and model of your computer, motherboard or (even better) detailed chip information. The latter can be found by running the following command and uploading the generated `SysInfo.txt` file:

    ```
    echo '=== PCI Devices: ===' >> SysInfo.txt && lspci >> SysInfo.
    ↪txt &&
    echo '=== USB Devices: ===' >> SysInfo.txt && lsusb >> SysInfo.
    ↪txt &&
    echo '=== Kernel Modules: ===' >> SysInfo.txt && lsmod >> SysInfo.
    ↪txt
    ```

- precise information about your SliTaz installation:
  - which version to you have? Stable or Cooking?
  - are you using the standard ISO or a specific flavor (XVESA, Lo-RAM etc.)?
  - is it installed on a HD or on an USB stick?
  - do you know which file-system you used?
- identify when your problem appeared:
  - was it just after installation or after an update?
  - were there any changes to your normal usage pattern or configuration?

This should provide enough system information for help to be provided.

[59] http://forum.slitaz.org/
[60] http://www.slitaz.org/en/mailing-list.html
[61] http://doc.slitaz.org/en:start

2.1.7.1 Cannot Login to Desktop

author jozee, linea, kultex, mojo

2.1.7.1.1 Symptoms

- **SLiM**, the SliTaz Login Manager, fails with the message:

  ```
  failed to execute login command
  ```

Warning: Verify the SliTaz install partition is not full and formated with a linux filesystem such as ext3. Using fat32 or ntfs filesystems cause login failure.

2.1.7.1.2 Explanation

The following files must exist, this can be verified with the **`ls -la`** command:

- `.Xdefaults`
- `.xinitrc`
- something else?

2.1.7.1.3 Solution

You should copy the default files from the template located in the `/etc/skel` directory. This happens automatically when a new user is created with the SliTaz Control Box, but not when using the command-line utilities. Occasionally users experience these files being removed or modified/broken.

Switch to the root (super) user:

```
su root
```

Change to the affected users `/home` directory:

```
cd /home/USERNAME
```

Set the shell options to allow the dot (.) to be included in file names:

```
shopt -s dotglob
```

Copy all files, recursively:

```
cp -r /etc/skel/* /home/USERNAME
```

Change ownership of all files and directories in the user's home to that of the affected user:

```
chown -R USERNAME:USERGROUP /home/USERNAME/*
```

Restore the shell options:

```
shopt -u dotglob
```

The essential files should now be restored!

EDIT:

```
shopt -u dotglob
```

is not working in `slitaz-3.0.iso` — instead of this run in addition:

```
chown -R USERNAME:USERGROUP /home/USERNAME/.[a-zA-Z0-9]*
```

2.1.7.2 Halt, reboot or shutdown via slim

author hgt, linea

2.1.7.2.1 Symptoms

Slim, the SliTaz Login Manager asks for a password when a special command like `halt`, `reboot` or `shutdown` was entered in place of a user name.

2.1.7.2.2 Explanation

The required password is that from user `root`!

2.1.7.2.3 Solution

Enter the root password and the appropriate action for `halt`, `reboot` or `shutdown` as defined in `/etc/slim.conf` with

```
halt_cmd    /sbin/poweroff
```

and

```
reboot_cmd    /sbin/reboot
```

Will be performed and no login.

2.1.7.3 XTerm — Cannot Copy/Paste

author seawolf

2.1.7.3.1 Overview

There are a variety of terminal emulators available for SliTaz and even more for Linux. SliTaz comes with the venerable **`XTerm`**, a basic but functioning program.

Tip: You may experience some problems with any graphical program, some more serious than others. By running them via a Terminal, any output will appear in the Terminal window so you can identify any problems. Just find its command, type it and press `Enter`.

2.1.7.3.2 Symptoms

- I cannot find the copy/paste feature
- Right-click for a menu does not work as expected

2.1.7.3.3 Explanation

`XTerm` does not have a right-click menu — or any other menu for that matter — but selection is activated by using the mouse buttons.

2.1.7.3.4 Solution

Selection, copy and paste is achievable using the mouse buttons. The three mouse buttons works thus:

- *left click* defines where the selection begins
- *middle click* defines where the selection ends & copies it to clipboard
- *right click* pastes the clipboard contents to the current cursor, not mouse, position

2.1.7.4 An Application Cannot Find Libraries/Files

author jozee, linea

2.1.7.4.1 Symptoms

- When run in a Terminal, an application fails with the message:

```
[name]: error while loading shared libraries: [library].so.*: cannot open shared object file: No such file or directory
```

2.1.7.4.2 Explanation

The program is missing some files it needs to run. This is caused by the package information missing a reference to another, or not including the file.

2.1.7.4.3 Solution

Check if the file required is available in another package. Use the package manager (**`TazPkg`**) to search for the file:

- start the package manager
- *Recharge* the package lists, if necessary
- click the *Search* tab
- in the *Search* box, enter the file-name (`library.so`) and click *Files*
- install the package (pick that with the closest name if there is more than one) by double-clicking the entry and clicking *Install Package*

Alternatively, run the following command from a terminal:

```
tazpkg search-pkgname library.so
```

. . . and install the appropriate package with:

```
tazpkg get-install [package]
```

If no packages are given, report this as a bug over at the Labs[62] as it needs to be included, or the dependency removed during the build.

2.1.7.5 A Program Doesn't Run or Quits Unexpectedly

author jozee, linea

2.1.7.5.1 Symptoms

- Nothing happens when an application is run from the Launch menu.
- An application quits without warning.

2.1.7.5.2 Explanation

One or more of problems may have caused this. Run the program in a Terminal to see any error messages the program may deliver.

2.1.7.5.3 Solution

Launch a Terminal (command line) and run the application by typing in its name. See the following FAQs on solving common problems.

2.1.7.6 Packages from Other Distributions

author unknown

[62] http://labs.slitaz.org

2.1.7.6.1 Symptoms

- There is a useful package that is not yet available in the SliTaz repositories.
- Can I use packages from other distributions on my SliTaz installation?

2.1.7.6.2 Explanation

Packages in a general sense are simply a collection of files and meta-data, compiled from source code to suit a particular environment and installation. Software binaries from other distributions can sometimes run if the libraries that power it exist for SliTaz. It is an easy but not-always-accurate way of testing packages before writing SliTaz *Receipts* (page 370) for a SliTaz version of the package.

The SliTaz Package Manager, Tazpkg, can convert[63] packages from the Debian, RedHat, Slackware and Arch Linux formats, by unpacking them and using the meta-data information inside to create a Tazpkg.

2.1.7.6.3 Solution

Simply run the following command in a Terminal:

```
tazpkg convert {filename}
```

The converted Tazpkg will be created after auto-detection of the original format.

Important: There may be occasions where the generated package does not function. This could be a result of mismatching libraries or missing dependant software. Similarly, the binaries may under-perform as they may have been compiled for a different environment and have to adapt. For these reasons, it is highly advisable to compile the software from source code and create a Tazpkg proper.

2.1.7.7 Mouse

author genesis, linea, mojo

2.1.7.7.1 I can’t change the mouse for left-handed people

Symptoms

I can’t change the mouse for left-handed people using the computer.

Explanation

You need to install the **`xorg-xmodmap`** package to configure the mouse buttons.

[63] http://hg.slitaz.org/tazpkg/raw-file/tip/doc/tazpkg.en.html#convert

Solution

Install the **xorg-xmodmap** package. In a terminal, type:

```
# tazpkg -gi xorg-xmodmap
```

To configure the mouse for left-handed:

```
# xmodmap -e "pointer = 3 2 1"
```

To configure the mouse for right-handed:

```
# xmodmap -e "pointer = 1 2 3"
```

To check the configuration of your mouse:

```
# xmodmap -pp
```

To make the changes permanent for your user, create a new file called `xmodmap.desktop` in the `~/.config/autostart` directory with the following content:

```
[Desktop Entry]
Type=Application
Name=Left hand mouse
Exec=xmodmap -e "pointer = 3 2 1"
NotShowIn=XFCE;Razor;LXDE;
```

Logout and Login to X session and the mouse buttons are configured for left-handed people.

To make the changes permanent for all users at the same time, repeat the procedure above as *root*, creating a new `xmodmap.desktop` file in the `/etc/xdg/autostart` directory.

References:

- Mouse left-handed[64]
- Configure left-handed mouse[65]

2.1.8 Making tazpackages

author brianperry, linea

Before starting on making slitaz packages, we'll begin with a quick overview of how a package installs programs:

When you click on a package to install (in **tazweb**), you tell the system to automatically download the package to `/var/lib/tazpkg/5.0/packages/`. It then automatically un-cpio's itself to `~/fs/` (or `/home/tux/fs/` when using the absolute path).

Then, it automatically puts all of the files you included in `/fs/` into the correct place on the hard disk, overwriting the old files bearing the same name, ... It knows where to place them as you use the exact same path of the files as the path the files are installed to on your harddisk (so if for instance, you want the package to copy a file called `foo.bar` to `/usr/share`, put `foo.bar` into `/fs/usr/share/` in the package).

[64] http://forum.slitaz.org/topic/mouse-left-handed
[65] http://forum.slitaz.org/topic/help-configure-left-handed-mouse#post-1792

After that, it executes the compile rules, genpkg rules, ... (assuming you packed it with **tazwok**; if packed with **tazpkg** it won't call these rules !)

So, with this, we're all ready now to make a tazpackage ourselves:

- put all the files/folders you need in /fs/path_of_files/; make sure the fs contents only contains binaries, scripts, configuration files, ... It should not contain source files, license, readme's, ... Compress the files as **cpio**, and then as **lzma** (do this using the cpio-command **cpio -o -H newc --quiet**, and the lzma-command **xz --format=lzma fs.cpio.lzma fs.cpio** or alternatively with a gui-based compression utility (like **xarchiver**, ...)
- put the file size details of the cpio.lzma folder in the receipt (you can find these by right-clicking → properties in **PCManFm** on the file). Also put in the following lines (replace "text" with your commands; for more rules, see http://doc.slitaz.org/en:cookbook:receipt):

```
compile_rules()
{
   text
}

genpkg_rules()
{
   text
}

post_remove()
{
   text
}
```

- make the files.list file by **cd**-ing to your tazpkg's fs folder, and then using following commands:

```
find . -type f -print > ../files.list
find . -type l -print >> ../files.list
cd ..; sed -i s/'^.'/''/ files.list
```

- make the md5 file by typing the command **md5sum filename**, or by using a program (on windows, you can use **winmd5**, **FCIV**, **certutil**, **hashtab**, **hashcheck**, ...)
- compress the fs.cpio.lzma file together with the receipt, files.list, md5sum, and description.txt into a cpio file renamed as filename.tazpkg. Do so by using the cpio compress command: **cpio -oH < fs.cpio.lzma receipt files.list md5sum description.txt > filename.tazpkg** or using a gui-based compression utility (like **xarchiver**, ...)

Alternatively, you can also make the package by using the **tazpkg pack** command; to see what that does exactly, see http://hg.slitaz.org/tazpkg/file/3af642cd5e69/modules/pack. In general it will make the md5 checksum and the filelist, and update the filesizes declared in the receipt. If you have already made the description and receipt it will also make the finished tazpackage in /home/tux (make sure you also pre-made the fs folder with the files therein for it to make a working tazpackage). Note that the wok too can be of use here (see http://doc.slitaz.org/en:oldcookbook:wok).

Once you have made a package, you can put it on a personal ftp site/webspace, or you can upload it to the slitaz package database using **mercurial**. See http://doc.slitaz.org/en:oldcookbook:wok on how to do this.

Also note you may also want to read:

- http://hg.slitaz.org/tazpkg/raw-file/tip/doc/tazpkg.en.html
- http://hg.slitaz.org/cookutils/raw-file/tip/doc/cookutils.en.html

2.1.9 Testing Applications

author seawolf, jozee

Important: Not started yet — this is only a suggestion to discuss in the forum

As a result of discussions in the forum[66], this is the page for documentation — please take part and report any troubles with packages here.

system-tools

Package	Maintainer	Tester	Status	Problems	Link
Midnight Commander	~	kultex	tested	ok	~
pcmanfm	~	~	not tested	~	~

x-window

Package	Maintainer	Tester	Status	Problems	Link
xorg-xf86-video-intel	~	kultex	tested	should be updated to 2.7.1; update in progress	http://hg.slitaz.org/wok/rev/c86710039a5f
xorg-xf86-video-nv	~	kultex	started	~	~

2.2 Installation, flavors and virtualization

2.2.1 PXE

author jozee, linea, seawolf, bellard

The *Preboot eXecution Environment* (or *PXE*, pronounced 'pixie') is the process of booting a computer from a network connection. It is comparable to booting a LiveCD from a remote CD drive.

This network boot method requires:

- a server to store files running DHCP and TFTP (each could be on a separate server);
- a client with a PXE boot-loader, stored in the BIOS firmware. It maybe disk-less.

[66] http://forum.slitaz.org/index.php/discussion/668/testing-packages/#Item_23

2.2.1.1 PXE Server Set-Up

A PXE server comprises:

- a DHCP server to accept clients;
- a DHCP boot-file to configure them;
- a TFTP server to deliver an OS.

2.2.1.1.1 Quick start with the Live CD

The SliTaz LiveCD can be used as a PXE server. To begin the process, launch the Netbox application (from the System Tools menu).

- From the *Static IP* tab, click *Start*.
  - This box will be the DHCP server. It can't use DHCP to get an IP configuration.

  Tip: Since SliTaz 3.0 the Netbox application is now split into Netbox and Serverbox. If you are running a recent SliTaz version, please read **Serverbox application** instead of **Server tab** below

- From the *Server* tab, select the *INETD* sub-tab and ensure the `tftpd` line is **uncommented** in `/etc/initd.conf`. This is the default behaviour. Click *Start*.
  - This will launch the TFTP server, which will deliver the SliTaz LiveCD across the network.
- From the *Server* tab, select the *PXE* sub-tab.
- Edit the configuration to add your boot options.
  - This will update the DHCP server configuration automatically.
- From the *Server* tab, select the *DHCP* sub-tab. Check that the configuration aligns with your network. The previous step has added the lines `boot_file` and `siaddr`. Click *Start*.
  - This will launch the the DHCP server. If clients to do not receive an IP address, check this configuration.
- Ensure the files `bzImage` and `rootfs.gz` are stored in the `/boot` directory of the LiveCD.

2.2.1.1.2 Customize your PXE server

- You can have multiple PXE configurations for the different client groups, see PXElinux wiki[67].
- You can store `/home` on a client local drive only (like tazusb does), example append `/etc/fstab` with :

```
/dev/hda1     /home    ext3     defaults    0   0
```

- Since SliTaz 3.0, you can have a hydrid installation on (some) clients. These clients have SliTaz installed with some huge packages like libreoffice. They boot with PXE and most of the system

[67] http://syslinux.zytor.com/wiki/index.php/PXELINUX#How_do_I_Configure_PXELINUX.3F

runs in RAM except the huge software linked to the hard disk (could be a network disk too). Example, append to /etc/init.d/local.sh:

```
mount -t nfs -o ro bootserver:/slitaz  /media/slitaz
tazpkg link libre-office /media/slitaz
```

- Since SliTaz 3.0, you can stack multiple initramfs in the pxelinux configuration file — An easy way to upgrade SliTaz and keep your customizations, example:

```
label slitaz
kernel /boot/bzImage
append initrd=/boot/rootfs.gz,/boot/configs/extra-packages.gz,/boot/
↪configs/special-configuration.gz rw root=/dev/null vga=normal␣
↪autologin
```

- Example of a PXE server configuration: The SliTaz web boot[68] server http://mirror.slitaz.org/pxe/ (start with pxelinux.cfg/default[69])

2.2.1.1.3 Test the PXE server with QEMU

- Install qemu

```
# tazpkg get-install qemu
```

- Launch the VM

```
# qemu -boot n -bootp /pxelinux.0 -tftp /boot
```

2.2.1.2 PXE Client Set-Up

Most recent machines with on-board Ethernet have a PXE-capable BIOS. Look for this feature in BIOS menus and the BIOS boot menu and ensure it is activated. It may require you press a key, such as F12, during the boot process.

If your computer does not support PXE booting, you can use SliTaz as a client instead. Create a bootable CD-ROM or floppy disk with the *Boot Floppy/CDROM* tool found in the *System Tools* menu.

In the *PXE Network* tab click *Write floppy*. Use this to boot the client computer.

Tip: Is your Ethernet card not recognised? See ROM-O-Matic[70]

2.2.1.3 Web Booting

The SliTaz LiveCD has configuration settings to start your computer via the Internet. This is useful for using a newer version of SliTaz from older media.

You can start the automatic process with the following command at the SliTaz LiveCD boot-splash:

[68] http://boot.slitaz.org/
[69] http://mirror.slitaz.org/pxe/pxelinux.cfg/default
[70] http://rom-o-matic.net/

```
web
```

That’s it!

You can find more information on using an Internet connection to boot your computer at the SliTaz Web Boot home-page[71].

You will need a DHCP server to get an IP address, netmask, gateway address, as per a normal network connection — a standard home router should be sufficient for this.

2.2.1.3.1 PXE boot without DHCP server: Web Boot & Command Line

If you have no device that can function as a DHCP server, you need an IP address with a netmask, gateway address and, optionally, a DNS address.

```
title Slitaz Web
  kernel /boot/gpxe ip=192.168.0.12/24 gw=192.168.0.1 dns=192.168.0.1␣
↪url=http://mirror.slitaz.org/pxe/pxelinux.0
```

You can modify the URL thus:

```
title Slitaz Web
  kernel /boot/gpxe ip=192.168.0.12/24 gw=192.168.0.1 dns=192.168.0.1␣
↪ip=192.168.0.12/24 gw=192.168.0.1 dns=192.168.0.1 url=http://mirror.
↪slitaz.org/pxe/pxelinux.0
```

Note that only the following keywords are recognised:

- `ip=`
- `gw=`
- `dns=`
- `url=`
- `nodhcp` (useful to avoid a DHCP timeout error)

2.2.1.4 Advanced Web Booting Configuration

The Web Booting process can be embedded into routers and other devices, as well as being customised.

2.2.1.4.1 Embedded Web Boot, with PXE boot PROM (PXE forwarder)

Configure a PXE server with http://download.tuxfamily.org/slitaz/boot/gpxe.pxe as the boot file, a 42Kb second stage loader. This was successfully tested with an OpenWRT[72] router:

- install http://mirror.slitaz.org/boot/mips/tftpd (mips version) in /jffs/usr/sbin
- install gpxe.exe in /jffs/boot
- add dhcp bootfile option in dnsmasq config file

[71] http://boot.slitaz.org/
[72] http://openwrt.org/

```
# echo "dhcp-boot=gpxe.pxe" >> /tmp/dnsmasq.conf
```

Launch the tftp server for your lan (say 192.168.0.1/24)

```
# /jffs/usr/sbin/tftpd 192.168.0.1 /jffs/boot
```

Tip: You can avoid the tftp server installation and use the SliTaz tftp server directly:

```
# echo "dhcp-boot=gpxe.pxe,mirror.slitaz.org" >> /tmp/dnsmasq.conf
```

2.2.1.4.2 Modifying the Default GPXE Web Boot URL

The URL is stored at offset 519 in 255 bytes max.

- Show the current URL with:

```
$ dd bs=1 skip=519 count=255 if=gpxe 2> /dev/null | strings
```

- Change the URL with:

```
$ echo -n "http://myurl.org/myboot" | cat - /dev/zero | dd␣
↪conv=notrunc bs=1 seek=519 count=255 of=gpxe
```

- Change the URL and IP configuration with:

```
$ echo -n "ip=192.168.0.10/24 gw=192.168.0.1 dns=192.168.0.1␣
↪url=http://myurl.org/myboot" | cat - /dev/zero | dd conv=notrunc␣
↪bs=1 seek=519 count=255 of=gpxe
```

- Remove URL to behave as a normal GPXE with:

```
$ dd if=/dev/zero conv=notrunc bs=1 seek=519 count=255 of=gpxe
```

2.2.1.4.3 Hack the gpxe.pxe Default Web Boot URL

The URL is stored at offset 5 in 255 bytes max.

Show the current URL with:

```
$ dd bs=1 skip=5 count=255 if=gpxe.pxe 2> /dev/null | strings
```

Change the URL with:

```
$ echo -n "http://myurl.org/myboot" | cat - /dev/zero | dd conv=notrunc␣
↪bs=1 seek=5 count=255 of=gpxe.pxe
```

Remove the URL and behave as a normal `gpxe.pxe` with:

```
$ dd if=/dev/zero conv=notrunc bs=1 seek=5 count=255 of=gpxe.pxe
```

2.2.1.4.4 Using Redundancy with Web Boot Servers

Comma separated URL lists are supported.

The PXE client will try to load the first URL. If the load fails, it will try the next URL, and so on.

Example with current Slitaz Web boot servers:

```
$ echo -n "http://mirror.slitaz.org/pxe/pxelinux.0,http://mirror.switch.ch/
↪ftp/mirror/pxe/pxelinux.0,http://download.tuxfamily.org/slitaz/pxe/
↪pxelinux.0" | cat - /dev/zero | dd conv=notrunc bs=1 seek=519 count=255␣
↪of=gpxe
```

2.2.1.5 Why use PXE? The VNC example

Let's say that your company is working on some very sensitive data. You don't want people copying anything on to removable media such as USB keys. Only a few users can use this data.

- PXELINUX chooses a special configuration by the MAC address in `pxelinux.cfg/`*`client-mac-address`*
- It checks the md5 (or sha256) password of the user boot entry with menu.c32
- It sends a kernel and an initramfs with a **fbvnc** package built by http://tiny.slitaz.org/ (total size < 1.44MB)
- The client boots in 1 to 5 seconds with a VNC framebuffer client
- The VNC server can send any OS display
- The client has no media driver and can use 20 year old hardware (may avoid theft risk)
- The target OS can run in a VM: more scalable and easier to maintain than multiple desktops
- No data is stored on the client machine. It may also have no disk. It only needs an ethernet card
- Of course, the sessions in the target OS must have a connection timeout and need a username and a password. . .

2.2.1.5.1 Increase security a bit

The VNC listens to the network without a password (fbvnc has no authentication support) and the VNC traffic is not encrypted on the network.

- Build an initramfs with a **fbvnc-ssh** package on http://tiny.slitaz.org/
- On the server, VNC should listen on localhost only
- The SSH public key of the client is installed in `$HOME/.ssh/authorized_keys` on the VNC server
- The VNC traffic can be compressed in the SSH tunnel (fbvnc supports raw frames only)

2.2.1.5.2 A quick demo

The menu *Tiny SliTaz* → *Tiny VNC* of the SliTaz Web Boot[73] launches the VNC client without ssh (you need a VNC server running on your network…).

Tip: You can directly download the kernel[74] and then the initramfs[75] and test it on your network or with qemu

---

Tip: Try with the cmdline argument `vga=ask` first. This will find the best VESA mode to use (example `vga=0x33B`)

2.2.2 Live CD for Low-RAM Systems

author jozee, seawolf, linea, bellard

The minimum RAM requirement for the SliTaz core Live CD is 160MB (128MB for 1.0). Many graphical applications won't run with such a low amount of memory; so, using the text-mode `screen=text` boot option is recommended.

The packages **`slitaz-loram`**, **`slitaz-loram-http`** or **`slitaz-loram-cdrom`** can be used to build a LiveCD for systems with RAM larger than 64MB, 32Mb or 24MB respectively. These can be installed directly on the host system, rather than specified in the packages list for the LiveCD you are building.

- **`slitaz-loram`** will compress the `/usr` tree and the system will still run in RAM. It will not use a CD-ROM, harddisk or USB key
- **`slitaz-loram-http`** will get the `/usr` tree from an ISO image stored in http://mirror.slitaz.org/ built using **`slitaz-loram-cdrom`**, and pass the `tiny` keyword while booting from the web
- **`slitaz-loram-cdrom`** will move the `/usr` tree onto the CD-ROM

`/usr` will be read-only. If the package **`funionfs`** or **`aufs`** is installed you will have read-write access to `/usr`.

These packages patch `/etc/init.d/rcS` to mount `/usr` and install two scripts in `/etc/tazlito`:

- `loram.rootfs` is called by **`tazlito gen-distro`** to compress or move `/usr`
- `loram.extract` is called by the **`slitaz-installer`** to uncompress or move `/usr` and install the same distribution as the SliTaz core LiveCD

Let's build a `slitaz-loram-cdrom.iso`!

2.2.2.1 slitaz-loram-cdrom

We boot the LiveCD and install the **`slitaz-loram-cdrom`** package:

[73] http://boot.slitaz.org/
[74] http://mirror.slitaz.org/pxe/tiny/vnc/bzImage.gz
[75] http://mirror.slitaz.org/pxe/tiny/vnc/rootfs.gz

```
# tazpkg get-install slitaz-loram-cdrom
```

tazlito gen-distro will then create an ISO with the packages listed in `/etc/tazlito/distro-packages.list` or `./distro-packages.list`. Since some more packages are installed, we remove these files to force **tazlito** to use all of the installed packages.

```
# rm -f /etc/tazlito/distro-packages.list ./distro-packages.list
```

Now we can build the ISO image…

```
# tazlito gen-distro
```

…and burn it.

```
# wodim dev=1,0,0 /home/slitaz/distro/slitaz-hacked.iso
```

2.2.2.2 Variations of slitaz-loram & slitaz-loram-cdrom

slitaz-loram compresses `/usr` with *cromfs* by default, which gives a higher compression ratio but is very slow. You can use *squashfs* instead:

- refuse to install *cromfs* during the **slitaz-loram** installation

  ```
  # yes n | tazpkg get-install slitaz-loram
  ```

- install **squashfs** with its dependencies

  ```
  # yes y | tazpkg get-install squashfs
  ```

slitaz-loram-cdrom moves `/usr` uncompressed to the CD-ROM and produces a 90MB ISO. If you install *cromfs* or **squashfs**, `/usr` will be compressed on the LiveCD and the ISO size will be around 30 megabytes.

2.2.2.3 Let's build a `slitaz-loram-cdrom-sqfs.iso`!

Install the package **slitaz-loram-cdrom** and **squashfs** (sqfs) on the host system:

```
# tazpkg get-install slitaz-loram-cdrom
# yes y | tazpkg get-install squashfs
```

Now we repeat the latter points of the above process:

```
# rm -f /etc/tazlito/distro-packages.list ./distro-packages.list
# tazlito gen-distro
# wodim dev=1,0,0 /home/slitaz/distro/slitaz-hacked.iso
```

2.2.2.4 slitaz-loram-cdrom and Large Memory Systems

When the **slitaz-loram-cdrom** LiveCD detects enough memory during boot, `/usr` is copied from the CD-ROM to RAM. You can eject and/or use the CD drive. The system behaves as a regular LiveCD in this case:

- a slitaz LiveCD (`/usr` was not compressed on the CD-ROM)
- a slitaz-loram LiveCD (`/usr` was compressed on the CD-ROM by **`squashfs`** or *cromfs*)

2.2.2.5 slitaz-loram-cdrom and Tiny Memory Systems

The boot command line is usually:

```
boot: slitaz args...
```

SliTaz boots on a 9MB RAM system with the boot command:

```
boot: loram single root=/dev/hdc
```

Where `/dev/hdc` is the CD-ROM device, the loram boot entry avoids RAM disk creation and CD-ROM detection.

Tip: Note that on a system with such a low amount of memory, the first thing to do is add swap!

You need 10MB to use the boot scripts with:

```
boot: loram root=/dev/hdc
```

In this case you can add arguments like `kmap=`, `config=`, etc.

2.2.2.6 slitaz-loram Auto-Extraction

Each slitaz-loram* flavor can be extracted into RAM at boot time (if enough memory is available) by using the boot argument `extract-loram`. You will get a core flavor running without read-only restrictions for `/usr`.

For example, assuming you boot the slitaz-loram-cdrom-sqfs:

```
boot: slitaz extract-loram
```

You will get:

- `/usr` read-only squashfs on a CD-ROM with a smaller RAM size
- `/usr` read-only squashfs in RAM with a medium RAM size (like slitaz-loram)
- `/usr` read-write tmpfs in RAM with a larger RAM size (like slitaz-core)

2.2.2.7 Build a slitaz-loram with tazlitobox

Since SliTaz 3.0, you can now build a slitaz-loram LiveCD more easily:

- launch **`tazlitobox`**
- click on the *Low RAM* tab
- select *The filesystem is always in RAM* (for slitaz-loram) or *The filesystem may be on a CD-ROM* (for slitaz-loram-cdrom)

- fill the ISO input with your SliTaz flavor (3.0 or more recent)
- update the ISO output
- click *build ISO*

The filesystem root (`/`) is compressed (not `/usr` only) and mounted read-write thanks to the **aufs** package. The **mount** and **df** commands will show a strange output, handy to detect that this kind of loram is running.

You can also use the following command in text mode:

```
# tazlito build-loram original.iso loram.iso
```

or:

```
# tazlito build-loram original.iso loram-cdrom.iso cdrom
```

Meta flavors are supported; you can loramize a slitaz-3in1.iso!

2.2.2.8 And what to do with only 8MB RAM?

Try Tiny SliTaz: http://tiny.slitaz.org/!

2.2.3 Winboot

author jozee, linea, bellard, mojo

2.2.3.1 Frugal install to Microsoft Windows partition

Install Grub4dos[76].

This article will assume that your `C:` drive is `(hd0,0)`. You need at least 160MB to run SliTaz cooking (128MB for Slitaz 1.0).

Tested with Windows XP on FAT32 and on NTFS.

For Windows XP, 2000 or NT, copy grldr[77] into `C:\` and append to `C:\boot.ini`:

```
C:\grldr="Grub4DOS"
```

2.2.3.2 Traditional frugal install

Copy bzImage[78] and rootfs.gz[79] (both <30MB) into `C:\boot` and append to `C:\menu.lst` the following text:

```
title SliTaz cooking
  kernel (hd0,0)/boot/bzImage rw root=/dev/null vga=normal
  initrd (hd0,0)/boot/rootfs.gz
```

[76] http://grub4dos.sourceforge.net/wiki/index.php/Grub4dos_tutorial#Installation
[77] http://mirror.slitaz.org/boot/grldr
[78] http://download.tuxfamily.org/slitaz/boot/cooking/bzImage
[79] http://download.tuxfamily.org/slitaz/boot/cooking/rootfs.gz

Works fine with Windows XP on FAT32 and on NTFS.

2.2.3.3 ISO image install

Copy slitaz-cooking.iso[80] (<30MB) into `C:\boot`. Run defragmentation on `C:`. And append to `C:\menu.lst` the following text:

```
title SliTaz cooking
  map (hd0,0)/boot/slitaz-cooking.iso (hd1)
  map --hook
  kernel (hd1)/boot/bzImage rw root=/dev/null vga=normal
  initrd (hd1)/boot/rootfs.gz
```

Works fine with Windows XP on FAT32 and on NTFS. Using version 0.4.3. The file `C:\boot\slitaz-cooking.iso` MUST be defragmented.

Tip: Anyway, you can always boot from an ISO image on a fragmented filesystem with:

```
title SliTaz cooking
  map --mem --heads=0 --sectors-per-track=0 (hd0,0)/boot/slitaz-cooking.
↪iso (hd1)
  map --hook
  kernel (hd1)/boot/bzImage rw root=/dev/null vga=normal
  initrd (hd1)/boot/rootfs.gz
```

You need 160MB or 256MB of RAM to run the `slitaz-cooking.iso`. If you have less RAM try: slitaz-cooking-base.iso[81] or slitaz-cooking-justX.iso[82] or slitaz-loram.iso[83] instead:

```
title SliTaz cooking base
  map (hd0,0)/boot/slitaz-cooking-base.iso (hd1)
  map --hook
  kernel (hd1)/boot/bzImage rw root=/dev/null vga=normal
  initrd (hd1)/boot/rootfs.gz
```

See http://community.slitaz.org/wiki/Loram

2.2.3.4 Web boot install

Copy gpxe[84] (<200KB) into `C:\boot` and append to `C:\menu.lst` the following text:

```
title SliTaz web
  kernel (hd0,0)/boot/gpxe
```

Gpxe is provided by the SliTaz Live CD too:

[80] http://download.tuxfamily.org/slitaz/iso/cooking/slitaz-cooking.iso
[81] http://mirror.slitaz.org/iso/cooking/flavors/slitaz-cooking-base.iso
[82] http://mirror.slitaz.org/iso/cooking/flavors/slitaz-cooking-justX.iso
[83] http://mirror.slitaz.org/iso/cooking/flavors/slitaz-loram.iso
[84] http://download.tuxfamily.org/slitaz/boot/gpxe

```
title SliTaz web
  map (hd0,0)/boot/slitaz-cooking.iso (hd1)
  map --hook
  kernel (hd1)/boot/gpxe
```

This will boot the latest available version of SliTaz. See http://boot.slitaz.org/. Works fine with Windows XP on FAT32 and on NTFS.

2.2.3.5 Tuning the boot process

Additional boot parameters may be appended. For example:

```
title SliTaz cooking
  map (hd0,0)/boot/slitaz-cooking.iso (hd1)
  map --hook
  kernel (hd1)/boot/bzImage rw root=/dev/null vga=extended lang=fr_FR␣
↪kmap=fr-latin1 laptop autologin config=/dev/hda1,boot/slitaz.sh
  initrd (hd1)/boot/rootfs.gz
```

The file `/boot/slitaz.sh` in `/dev/hda1` will be executed at the end of the boot script:

```
initrd (hd1)/boot/rootfs.gz (hd0,0)/boot/extra-softwares.gz (hd0,0)/boot/
↪my-config-files.gz
```

The compressed cpio archives `/boot/extra-softwares.gz` and `/boot/my-config-files.gz` will be loaded after the official initramfs `rootfs.gz`. This is faster than using the previous `/boot/slitaz.sh` script. Example:

```
# find /etc/rcS.conf /etc/daemons.conf /etc/dropbear /home/tux/.ssh | cpio␣
↪-o -H newc | gzip -9 > /boot/my-config-files.gz
```

2.2.3.6 Automated Graphical Approach using UNetbootin

A SliTaz Frugal Install or a Live USB installation can be performed using UNetbootin[85].

Warning: Unetbootin will not work with the main 4.0 release slitaz-4.0.iso or slitaz-rolling.iso which are core 4-in-1 ISO's with multi-rootfs.gz without manually editing the boot menu or combining the multi-rootfs.gz into one.

Please use slitaz-4.0-core.iso[86] or another single rootfs.gz flavor found here[87].

Either the standard version (Windows[88] | Linux[89]) can be used (select SliTaz from the Distribution list), or a custom SliTaz version (Windows[90] | Linux[91]) may also be used.

[85] http://unetbootin.sourceforge.net/
[86] http://mirror.slitaz.org/iso/4.0/flavors/slitaz-4.0-core.iso
[87] http://mirror.slitaz.org/iso/4.0/flavors/
[88] http://unetbootin.sourceforge.net/unetbootin-windows-latest.exe
[89] http://unetbootin.sourceforge.net/unetbootin-linux-latest
[90] http://unetbootin.sourceforge.net/unetbootin-windows-latest.exe
[91] http://unetbootin.sourceforge.net/unetbootin-linux-latest

To perform a Frugal Install, select *Hard Disk* under installation type; to create a Live USB, select *USB Drive* under installation type.

2.2.4 Unusual install methods

author bellard, linea, babaorum, mojo

The *slitaz-installer* (page 286) installs SliTaz in a partition or a hard drive as most Linux distributions do. However, there are many other ways to install SliTaz…

The following configurations are using the SliTaz 4.0 boot loader **grub4dos-linux**.

Most kernel command line arguments are processed by /init[92] during the *boot process* (page 142).

2.2.4.1 Frugal install

You don't need a special partition, the system runs in RAM like a Live CD…

See *Traditional frugal install* (page 113) and *ISO image install* (page 114)

…it can be tuned to your needs a little bit.

See *Tuning the boot process* (page 115)

You can also use a LORAM flavor created with **tazlitobox** and *The filesystem is always in RAM*…

…or *The filesystem may be on a small CD-ROM*. Install the CD-ROM files `/boot/bzImage` and `/boot/rootfs.gz` and copy the `/rootfs.gz`. Say into `/this/directory`. Now get the label of the partition. Say mypartition:

```
# blkid
```

And append the param `loram=` to the cmdline:

```
kernel (hd1)/boot/bzImage rw root=/dev/null vga=normal␣
↪loram=LABEL=mypartition,this/directory
```

Tip: You can also use a device name (`loram=/dev/hda2,this/directory`). The device name may vary with the kernel version (`hda` or `sda`) and USB keys.

2.2.4.2 USB key install

2.2.4.2.1 Tazusb

This is a mix between a frugal and traditional install. The system runs fully in RAM but the `home` directory is always on the key. You can modify the system (configure, install packages) and then save the new system on the key 8-). See tazusb manual[93]

Tip: You can create the USB key from the ISO image

[92] http://hg.slitaz.org/slitaz-boot-scripts/file/tip/init
[93] http://hg.slitaz.org/tazusb/raw-file/tip/doc/tazusb.en.html

```
# tazusb gen-iso2usb slitaz-cooking.iso
```

2.2.4.2.2 Hybrid ISO

This install method will **erase all of your key** and install a **unmodifiable** :-/ SliTaz. You can create a custom system with **`tazlito`** or **`tazlitobox`**. Each SliTaz ISO image is *hybrid* (page 130).

Tip: Use **`fdisk`** to get back the remining free space on the USB keys. Simply add a partition(s).

```
# dd if=slitaz-cooking.iso of=/dev/usbkey
# fdisk /dev/usbkey
```

2.2.4.3 Floppy install

2.2.4.3.1 The last resort install

Imagine you have a very old PC with a floppy drive and a hard disk. No CD-ROM, no network card, and no USB. The hard disk works only with this machine. You can't plug it into your friend's PC or into an USB disk box.

2.2.4.3.2 Prepare a floppy set

Get a floppy set from http://mirror.slitaz.org/floppies/. The base subset should be sufficient (6 floppies).

Tip: Your may use one floppy only with tiny slitaz[94] for a manual install (without the slitaz-installer)

Build a data floppy set from the ISO image:

```
# echo "slitaz.iso" | cpio -o -H newc | split -b 1440k /dev/stdin iso
# dd if=isoaa of=/dev/fd0
# dd if=isoab of=/dev/fd0
# ...
```

Tip: Some data floppy sets[95] can be generated on SliTaz mirrors

2.2.4.3.3 Transfer the ISO image onto hard disk

Boot from the SliTaz floppy set, mount a hard disk partition in `/mnt` and restore the data floppy set in `/mnt`:

[94] http://pizza.slitaz.org/tiny/
[95] http://mirror.slitaz.org/floppies/#fdiso

```
# cd /mnt
# dd if=/dev/fd0 of=fdiso01
# dd if=/dev/fd0 of=fdiso02
  ...
# cat fdiso* | cpio -i
# rm fdiso*
```

Tip: If space is critical, replace the last 2 lines with:

```
for i in fdiso*; do cat $i; rm -f $i; done | cpio -i
```

Now you can use any install method from an ISO image. Example:

```
# mount -o loop,ro slitaz.iso /media/cdrom
# slitaz-installer
```

Tip: With tiny slitaz[96], you don't have the slitaz-installer. You can start a frugal install:

```
# mkdir boot && cp /media/cdrom/boot/bzImage /media/cdrom/boot/rootfs.gz␣
↪boot && rm slitaz.iso
```

and optionally a standard install:

```
# unlzma -c boot/rootfs.gz | cpio -id
```

The problem: you have no boot loader!

Boot the SliTaz generic boot floppy[97] with the *SliTaz frugal* menu entry. Now you can install the **`GRUB`** bootloader on the hard disk.

2.2.4.4 Loop install

If you want to install SliTaz on a disk (not a frugal install), and you don't want to create a partition for SliTaz, but you have enough room in a feature-poor filesystem (FAT32 or NTFS)...

... create a loop file and install SliTaz into it!

The problem is: what size? 200MB should be the minimum. Imagine, you could like it and install many more packages!

2.2.4.4.1 Loopfile creation

You can create the loop file with **`mountbox`** (click *loop*, enter the file name, then click *create*, enter the size, the units, click *create*) or with the command line:

```
# dd if=/dev/zero bs=1M count=200 of=slitaz.fs
```

[96] http://pizza.slitaz.org/tiny/
[97] http://mirror.slitaz.org/boot/floppy-grub4dos

You now need to create a filesystem in this loopfile:

```
# yes | mke2fs -j slitaz.fs
```

Later, if the loopfile is too small you can extend it (assuming you don't boot from the loopfile, but a Slitaz Live CD for example):

```
# dd if=/dev/zero bs=1M count=100 >> slitaz.fs
# resize2fs slitaz.fs
```

2.2.4.4.2 Root filesystem files installation

Copy files from the `rootfs.gz` archive of a CD-ROM into the loopfile:

```
# mount /dev/cdrom /media/cdrom
# mount -o loop,rw slitaz.fs /mnt
# unlzma -c /media/cdrom/boot/rootfs.gz | ( cd /mnt; cpio -idmu )
# umount -d /mnt
# umount /media/cdrom
```

2.2.4.4.3 Boot setup

Get a **preinit** ISO file with same version (the kernel version must match the modules version in the root filesystem). The partition storing the loopfile (say `/dev/hda1`) and its path into the partition (say `/data/slitaz.fs`) is defined by the `mount` and `loopfs` arguments:

```
title SliTaz cooking
  map (hd0,0)/boot/slitaz-preinit.iso (hd1)
  map --hook
  kernel (hd1)/boot/bzImage mount=/dev/hda1 loopfs=data/slitaz.fs
  initrd (hd1)/boot/rootfs.gz
```

Tip: The loop install does not use exotic packages from preinit. You can use any SliTaz flavor (except lorams).

Tip: You can built an up-to-date **preinit** ISO anytime with

```
# tazlito get-flavor preinit
# tazlito gen-distro
```

```
title SliTaz cooking in loop file
  map (hd0,0)/boot/slitaz-cooking.iso (hd1)
  map --hook
  kernel (hd1)/boot/bzImage mount=/dev/hda1 loopfs=data/slitaz.fs
  initrd (hd1)/boot/rootfs.gz

title SliTaz cooking in RAM (like the Live CD)
  map (hd0,0)/boot/slitaz-cooking.iso (hd1)
```

(continues on next page)

(continued from previous page)

```
  map --hook
  kernel (hd1)/boot/bzImage rw root=/dev/null autologin
  initrd (hd1)/boot/rootfs.gz
```

Or, you can replace the device name of the `mount` variable by the UUID or LABEL returned by **`blkid`**:

```
title SliTaz cooking
  map (hd0,0)/boot/slitaz-preinit.iso (hd1)
  map --hook
  kernel (hd1)/boot/bzImage mount=a4b346ee-4c7b-46aa-9fd4-6bc39ab4fa96␣
↪loopfs=data/slitaz.fs
  initrd (hd1)/boot/rootfs.gz
```

Tip: You can extract the `bzImage` and `rootfs.gz` from the ISO image to avoid **`map`** commands and defragmentation.

2.2.4.5 Subdirectory install in a Posix filesystem

If you want install SliTaz on a disk (not a frugal install), and you don't want to create a partition for SliTaz, but you have room in a filesystem for another Unix and you don't know how much space to reserve for SliTaz…

… create a subdirectory and install SliTaz into it!

2.2.4.5.1 Root filesystem files installation

Simply install SliTaz file in a subdirectory (say `/var/slitaz`) of another Linux partition:

```
# mkdir /mnt/var/slitaz
# unlzma -c /media/cdrom/boot/rootfs.gz | ( cd /mnt/var/slitaz ; cpio -
↪idmu )
```

2.2.4.5.2 Boot setup

Like a loop install, you need a preinit ISO file with a matching version. The partition (say `/dev/hda1`) and the path into the partition are defined by the `mount` and `subroot` arguments:

```
title SliTaz cooking
  map (hd0,0)/boot/slitaz-preinit.iso (hd1)
  map --hook
  kernel (hd1)/boot/bzImage mount=/dev/hda1 subroot=var/slitaz
  initrd (hd1)/boot/rootfs.gz
```

Both notes in *Loop install* (page 118) section about `bzImage` extraction and UUID/LABEL also apply here.

Tip: The subdirectory install does not use exotic packages from preinit. You can use any SliTaz flavor (except lorams).

Tip: The subdirectory install can be easily tested with a Raspberry Pi[98] running Raspbian[99] with the tazbian[100] script. This script creates a **`raspbian`** package from the latest SliTaz tarballs found on the mirror[101]. The installation of this package

```
$ sudo dpkg -i slitaz-<VERSION>-1_armhf.deb
```

will install SliTaz in `/var/os/slitaz` and setup a multiboot. It does not remove rasbian or alter partitions.

2.2.4.6 Subdirectory install in a non-Posix filesystem

You want to install SliTaz in a subdirectory but the filesystem (NTFS[102] or VFAT[103]) does not fully support UNIX features.

Use posixovl[104]!

2.2.4.6.1 Root filesystem files installation

You need to mount the target subdirectory (say `/slitaz`) with **`posixovl`** **before** installing the files.

```
# mkdir /mnt/slitaz
# mount.posixovl /mnt/slitaz
# unlzma -c /media/cdrom/boot/rootfs.gz | ( cd /mnt/slitaz ; cpio -idmu )
```

Tip: Windows users can extract the archive http://mirror.slitaz.org/iso/4.0/slitaz-4.0.zip and look at the file `\slitaz\boot\install.txt`

2.2.4.6.2 Boot setup

Like a loop install, you need a *preinit* ISO file with a matching version. The partition (say `/dev/hda1`) and the path into the partition are defined by the `mount`, `subroot` and **posixovl** arguments:

```
title SliTaz cooking
  map --mem --heads=0 --sectors-per-track=0 (hd0,0)/boot/slitaz-preinit.
↪iso (hd1)
```

(continues on next page)

[98] http://en.wikipedia.org/wiki/Raspberry_Pi
[99] http://en.wikipedia.org/wiki/Raspbian
[100] http://hg.slitaz.org/slitaz-arm/raw-file/tip/rpi/tazbian
[101] http://mirror.slitaz.org/arm/rpi/
[102] http://en.wikipedia.org/wiki/NTFS
[103] http://en.wikipedia.org/wiki/VFAT
[104] http://mirror.slitaz.org/pkgs/?package=posixovl

(continued from previous page)

```
  map --hook
  kernel (hd1)/boot/bzImage mount=/dev/hda1 subroot=slitaz posixovl
  initrd (hd1)/boot/rootfs.gz
```

Both notes in *Loop install* (page 118) section about `bzImage` extraction and UUID/LABEL also apply here.

2.2.4.6.3 Extra setup

You want to see the host partition while running SliTaz like UMSDOS[105] does with `/DOS`.

Create the mount point:

```
# mkdir /mnt/slitaz/Windows
```

And update the boot arguments:

```
title SliTaz cooking
  map (hd0,0)/boot/slitaz-preinit.iso (hd1)
  map --hook
  kernel (hd1)/boot/bzImage mount=/dev/hda1 subroot=slitaz posixovl␣
↪bindfs=.,slitaz/Windows
  initrd (hd1)/boot/rootfs.gz
```

2.2.4.7 LVM install

The Logical Volume Manager can manage (add disks, replace disks…) and logically freeze any disks for backup (snapshots) without disrupting service. See LVM[106]

2.2.4.7.1 LVM partition setup

A small amount of storage (depending on the disk activity, likely between 1% and 15%) is used by snapshots to hold frozen data during a backup. Assuming we use the `sda1` partition with 5% reserved for snapshots:

```
# tazpkg get-install lvm2
# modprobe dm-mod
# pvcreate /dev/sda1
# vgcreate slitaz /dev/sda1
# lvcreate -l 95%VG slitaz -n root
# mke2fs -j /dev/mapper/slitaz-root
# tune2fs -c 0 -i 0 /dev/mapper/slitaz-root
# mount /dev/mapper/slitaz-root /mnt
```

2.2.4.7.2 Root filesystem files installation

Similar to a loop install:

[105] http://en.wikipedia.org/wiki/UMSDOS
[106] http://en.wikipedia.org/wiki/Logical_Volume_Manager_(Linux)

```
# unlzma -c /media/cdrom/boot/rootfs.gz | ( cd /mnt ; cpio -idmu )
```

2.2.4.7.3 Boot setup

Like a loop install, you need a **preinit** ISO file with a matching version. The argument **lvmroot** holds the volume name:

```
title SliTaz cooking
  map (hd0,0)/boot/slitaz-preinit.iso (hd1)
  map --hook
  kernel (hd1)/boot/bzImage lvmroot=slitaz-root
  initrd (hd1)/boot/rootfs.gz
```

2.2.4.8 RAID install

2.2.4.8.1 Hardware RAID

Full hardware RAID[107] is transparent for SliTaz. The disk array is seen as a single disk and nothing special has to be done to install SliTaz.

2.2.4.8.2 Semi hardware RAID

Creation & installation

The RAID[108] array is built with the BIOS menus. SliTaz needs the driver **dmraid** to see the array and not only each hard disk:

```
# tazpkg get-install lvm2
# tazpkg get-install dmraid
# dmraid -s              # shows raid infomation
# modprobe raid1         # could be raid0, raid456 or raid10
# dmraid -ay             # activates the array in /dev/mapper
# mount /etc/mapper/myraid /media
# unlzma -c /media/cdrom/boot/rootfs.gz | ( cd /mnt ; cpio -idmu )
```

Boot setup

Like a loop install, you need a **preinit** ISO file with a matching version. The argument **dmraid** holds the volume name:

```
title SliTaz cooking
  map (hd0,0)/boot/slitaz-preinit.iso (hd1)
  map --hook
  kernel (hd1)/boot/bzImage dmraid=myraid
  initrd (hd1)/boot/rootfs.gz
```

[107] http://en.wikipedia.org/wiki/RAID
[108] http://en.wikipedia.org/wiki/RAID

2.2.4.8.3 Software RAID

The array does not need the BIOS and can be fully administered remotely!

Tip: You should tune the **preinit** flavor to your needs. Enable the **dropbear** startup in /etc/rcS.conf and maybe install a VPN. If the software RAID does not start on startup, you will be able to fix it remotely. . .

Creation & installation

Example for mirroring (raid1) devices /dev/sda3 and /dev/sdb3:

```
# tazpkg get-install lvm2
# tazpkg get-install mdadm
# echo y | mdadm --create /dev/md0 --level=1 --raid-devices=2 /dev/sda3 /
↪dev/sdb3 --bitmap=internal --assume-clean
# modprobe raid1
# mdadm --assemble --scan
# mount /dev/md0 /media
# unlzma -c /media/cdrom/boot/rootfs.gz | ( cd /mnt ; cpio -idmu )
```

Boot setup

Like a loop install, you need a preinit ISO file with a matching version. The argument **softraid** holds the device name:

```
title SliTaz cooking
  map (hd0,0)/boot/slitaz-preinit.iso (hd1)
  map --hook
  kernel (hd1)/boot/bzImage softraid=/dev/md0
  initrd (hd1)/boot/rootfs.gz
```

2.2.4.9 Crypto install

Note: Encrypts the whole root filesystem[109], not just the /home partition. Important, because the files in /tmp, /var/tmp may betray your work. The swap does too unless you use a file instead of a partition (like /tmp/swapfile; this will be encrypted too because this file is in the root filesystem)

2.2.4.9.1 LUKS

LUKS[110] replaces the Cryptoloop[111] and Loop-AES formats now.

[109] http://en.wikipedia.org/wiki/Disk_encryption
[110] http://en.wikipedia.org/wiki/LUKS
[111] http://en.wikipedia.org/wiki/Cryptoloop

Creation & installation

Create the encrypted device with **mountbox** (*crypto* button). You may have to accept the missing packages installation. Select the device (say /dev/sda3) and click the *create* button. Now you can start to format it:

```
# mke2fs -j /dev/mapper/crypto-sda3
# tune2fs -c 0 -i 0 /dev/mapper/crypto-sda3
# mount /dev/mapper/crypto-sda3 /media
# unlzma -c /media/cdrom/boot/rootfs.gz | ( cd /mnt ; cpio -idmu )
```

Boot setup

Like a loop install, you need a preinit ISO file with a matching version. The argument cryptoroot holds the volume name:

```
title SliTaz cooking
  map (hd0,0)/boot/slitaz-preinit.iso (hd1)
  map --hook
  kernel (hd1)/boot/bzImage cryptoroot=sda3
  initrd (hd1)/boot/rootfs.gz
```

2.2.4.9.2 Loop-AES compatibility

This deprecated format needs the same **boot setup** as LUKS.

2.2.4.10 Mixed install

You can mix several above methods using one device access and/or one filesystem access.

Device access	Filesystem access
mount=	subroot=
*raid=	loopfs=
lvmroot=	cryptoroot=
*raid= + lvmroot=	loopfs= + cryptoroot=

2.2.4.10.1 Example 1 : RAID + LVM

2.2.4.10.2 Example 2 : Loop + crypto

2.2.4.10.3 Possible improvements?

Add network support: nbd/iscsi + RAID 1 net&local + crypto

2.2.4.11 PXE: No install!

You can *setup a PXE server* (page 105) (well… you need to configure your server) or a *PXE forwarder* (page 107), see the *Embedded Web Boot with PXE boot PROM* (the SliTaz team has configured the server for you[112] 8-))

2.2.5 LiveCD

author seawolf, brianperry

This manual is intented to help you on making a SliTaz LiveCD. Once you made a customized SliTaz ISO, or after simply having downloaded a regular SliTaz ISO from the SliTaz website, you can go ahead and burn it to a CD. However, as the writer of this guide has found, not all programs always work. So here's a list of programs that are found not to work:

- **`Nero Burning ROM 6`** (Windows)

These programs do work:

- **`imgBurn`** (Windows, or Linux using Wine)
- **`ISO Master`** (Linux; should work, see here[113])
- **`Graveman`** (Linux; should work, see here[114])
- **`k3b`** (Linux)
- **`wodim`** (Linux)

2.2.6 Live USB

author jozee, linea, pankso, totoetsasoeur, mojo, fantomas, brianperry, bellard

This page describes how to make a SliTaz live usb stick.

2.2.6.1 From Windows

A SliTaz Live USB installation can be performed using:

- LinuxLive USB Creator[115]
- Universal USB Installer[116]
- **`UnetBootin`**
- **`TazUSB`**

[112] http://boot.slitaz.org
[113] http://doc.slitaz.org/en:handbook:utilities#iso-master-create-and-edit-isos
[114] http://www.slitaz.org/en/doc/scratchbook/base-system.html#mkiso
[115] http://www.linuxliveusb.com/
[116] http://www.pendrivelinux.com/universal-usb-installer-easy-as-1-2-3/

Warning: SliTaz with xorg-server and persistent `/home` requires a Linux filesystem (ext2,ext3) for regular users to login to an X-session.

Fat32 or NTFS filesystems cause SLiM login manager error: `Failed To Execute Login Command`

Tip: Solution:

SliTaz runs in RAM so after booting it no longer needs the flash drive. This enables you to reformat the flash drive to a Linux filesystem and reinstall SliTaz.

---

1. Install SliTaz with LinuxLive USB Creator[117] or Universal USB Installer[118]
2. Copy the `slitaz.iso` to the root of the flash drive.
3. Boot into SliTaz with the flash drive,login root, password: root
4. Mount[119] the flash drive,copy the `slitaz.iso` to `/root`
5. Use **From SliTaz** instructions above to format the flash drive ext2, reinstall SliTaz using ISO file as install source.

Warning: **`Tazusb.exe`** and **`Unetbootin`** do not work with any `cooking.iso`, `rolling.iso` or `slitaz-4.0.iso` (core 4-in-1).

They will work with slitaz-4.0-core.iso[120] or another single `rootfs.gz` flavor found here[121].

Warning: **`Tazusb.exe`** does not work with any `cooking.iso`, `rolling.iso` or `slitaz-5.0.iso`.

The `.iso` file must be renamed to an `.exe` file, i.e.:

1. boot into Windows,
2. download the latest `slitaz-rolling.iso`,
3. go to the downloaded file directory,
4. rename the file `slitaz-rolling.iso` to `slitaz-rolling.exe`,
5. double-click on the file `slitaz-rolling.exe` to run it,
6. follow the instructions given by the application (2-3 steps):
    1. Do you want to create a boot key: => answer yes,
    2. Step 1: unplug the USB stick,
    3. Step 2: plug the USB stick in and wait for Windows to mount it,

[117] http://www.linuxliveusb.com/
[118] http://www.pendrivelinux.com/universal-usb-installer-easy-as-1-2-3/
[119] http://doc.slitaz.org/en:handbook:commands
[120] http://mirror.slitaz.org/iso/4.0/flavors/slitaz-4.0-core.iso
[121] http://mirror.slitaz.org/iso/4.0/flavors/

4. wait for the confirmation window,

7. reboot,
8. press `Fn` key to choose alternative boot device, slitaz boot panel must be displayed,
9. select the language,
10. `RETURN` to boot.

2.2.6.1.1 Remarks when using `UNetbootin`

You can find UNetbootin for Windows[122] at this page. Note that you need to select SliTaz ISO from the *Distribution list* in UNetbootin. To create a Live USB, select *USB Drive* under installation type. You can also choose to do a frugal install. To perform a Frugal Install, select *Hard Disk* under installation type.

2.2.6.1.2 Remarks when using `TazUSB`

The binary file `TazUSB.exe` is a Windows executable that installs an ISO file on a USB drive.

- download **`TazUSB`** executable for Windows: tazusb.exe[123] (the sources are available here tazusb.nsi[124])
- plug in your USB drive and format it in **FAT32** (formatting might not be required),
- launch **`TazUSB`** application by double-clicking on `TazUSB.exe` executable file,
- choose the language,
- select the destination USB drive (**double-check that the letter refers to the correct drive**),
- select the SliTaz **iso**

2.2.6.2 From Linux

2.2.6.2.1 From any Linux distro

- You can use **`unetbootin`**; download from UNetbootin for Linux[125].

2.2.6.2.2 From Slitaz linux

Download > Burn > Boot a SliTaz ISO image

Verify the install target, format will delete everything.

[122] http://unetbootin.sourceforge.net/unetbootin-windows-latest.exe
[123] http://mirror.slitaz.org/packages/windows/tazusb.exe
[124] http://hg.slitaz.org/tazusb/raw-file/tip/win32/tazusb.nsi
[125] http://unetbootin.sourceforge.net/unetbootin-linux-latest

```
$ su root
# fdisk -l
# tazusb format /dev/sdxx
```

LiveCD as install source

```
# tazusb gen-liveusb /dev/sdxx
```

ISO file as install source

```
# tazusb gen-iso2usb slitaz.iso /dev/sdxx
```

Note that the `/dev/sdxx` part of the command above specifies the location to where you need to write the ISO's files to; it is not the source media where the ISO is on! With the ISO file as install source option, you should hence specify the exact location where the SliTaz ISO source file is located. If it is located at `/home/tux` (while running SliTaz from a non-live version), write

```
# tazusb gen-iso2usb /home/tux/slitaz.iso /dev/sdxx
```

If the source file is found on an external hard disk, mention the folder in which the media's files are outputted to — check this with **PCManFM** (in practice the folder can be something like `/media/disk`). So, the command you'd need to write with the latter would be something like

```
# tazusb gen-iso2usb /media/disk/slitaz.iso /dev/sdxx
```

Once done, wait… and then reboot!

2.2.6.2.3 SliTaz 4.0 kernel panic (not booting)

In Slitaz 4.0 the rootfs file is divided into 4 parts which is a problem with **tazusb.exe** and **UNetbootin**. For SliTaz to work with **tazusb.exe** and **Unetbootin** you must download the core flavor of SliTaz[126] consisting of a single file rootfs.

2.2.6.3 Sources

- how to install slitaz 4.0 on usb[127]

2.2.6.4 See also

- *Live USB* (page 126)

2.2.7 All SliTaz resources on one DVD

author jozee, linea

A bootable DVD image of all available packages version is generated every day. See http://www.slitaz.org/en/get/. Both *cooking* or *stable* versions are built; they contain:

[126] http://mirror.slitaz.org/iso/4.0/flavors/slitaz-4.0-core.iso

[127] http://oldpapyrus.wordpress.com/tag/how-to-install-slitaz-4-0-on-usb/

- a Slitaz boot
- a copy of the website
- a snapshot of the wok
- all official packages

It is designed to be used without an internet connection.

2.2.7.1 Hybrid ISO

The ISO image should be burned onto a DVD using:

```
# wodim -v dev=/dev/dvd packages-cooking.iso
```

It can also be written onto a USB key (say the key is `/dev/usbkey`). USE WITH CAUTION this will erase all your data.

```
# dd if=packages-cooking.iso of=/dev/usbkey
```

After writing the USB key, you can create partitions to get back the remaining storage.

```
# fdisk /dev/usbkey
```

The USB key is bootable. If your BIOS doesn't support USB boot, you can boot **plop** from http://www.slitaz.org/en/get/#floppy floppy.

2.2.7.2 Burnable on CD-ROM

The files on the ISO image are ordered in 5 groups:

- the boot files
- the loram boot files
- the website
- the wok
- the packages

The four first groups fit in the first 100-200Mb of the ISO image. If you don't need the packages, you can burn it onto a mini CD-ROM (8cm/210Mb)

```
# wodim -v dev=/dev/cdrom packages-cooking.iso
```

The packages are stored in alphabetical order. **915resolution** should likely be present but **zsh** surely not. The directories are stored at the beginning of the image and are always present. If you try to load a non-present package (**zsh**) you will get an I/O error.

2.2.7.3 Dual boot

Both *core* and *loram* (*loram-cdrom* to be exact) boot files are stored in the ISO image. The default boot entry (slitaz) will select which to start according to your RAM size. You can force the choice with *core* or *loram* keyword:

```
boot: loram screen=1280x1024
```

With core boot you can:

- unmount the DVD (**`umount /cdrom`**)
- test (install) packages in RAM

You can't do that with loram boot because it has `/usr` mounted read-only from the DVD. You can install SliTaz on a hard disk with both boot files.

2.2.7.4 Auto install

During boot the following is performed:

- the DVD is mounted in `/cdrom`, the wok is onto `/home/slitaz/wok`
- a pseudo tazpkg recharge links packages from the DVD
- the SliTaz web site is installed in file:///cdrom/website/index.html

Note: USB key users should read USB key instead of DVD

2.2.8 Installing SliTaz to a MMC/SD-card

author monz, linea, jozee

SliTaz can be installed to a MMC/SD flash-memory card (the kind used in digital cameras) in either of two ways: in a LiveUSB version as used on a "memory-stick" (also called "thumb-drive", "flash-drive", or a variety of other names), or as a regular install as used on a hard-drive. There are advantages and disadvantages to both methods.

2.2.8.1 LiveUSB-type installation

A memory-stick uses NAND flash memory which eventually wears out after about 100,000 read/write cycles. SliTaz tries to minimize this wear, and thus prolong the life of the memory-stick, by storing the entire main root-file-system ("rootfs") as a compressed image in one file (`rootfs.gz`), then uncompressing it into the machine's memory (RAM) when booting. Only the `/home` directory is normally written onto the memory-stick, unless the user issues this command:

```
# tazusb writefs [compression]
```

Where "[compression]" can be "lzma", "gzip", or "none". This command re-writes the `rootfs.gz` file onto the memory-stick, so that changes made to the system can be saved and used again at the next boot-up. The big advantage to using this method is obviously that it prolongs the life of the USB device. The big disadvantage is that booting takes a long time because of the use of compression, and the more packages you install to your rootfs system the longer it takes to boot. The rootfs can also be saved without compression (i.e.,

```
# tazusb writefs none
```

), which will allow it to boot much faster. This was not such a good option when SliTaz 1.0 was released in 2008, because memory-sticks had much less storage space then, but with 4 GB memory-sticks common now, it's not as much of a problem. This same method can be used to install SliTaz to a MMC/SD card.

2.2.8.2 Regular "hard-drive"-type installation

The other option is to treat the MMC/SD card as if it were a real hard-drive, and simply use:

```
# gparted
```

to format the card and set up partitions like on a regular hard-drive, and

```
# slitaz-installer
```

to do the actual installation. Be sure to also allow the installer to install GRUB onto the card, so that you can boot from it the normal way, same as you would from a regular hard-drive.

2.2.8.3 Example as used on my own Asus Eee-PC

The SliTaz installation which I put onto a MMC/SD-card (16 GB) was for my Asus Eee-PC 701, leaving the Eee's original Xandrox Linux installation intact on its SSD (solid-state drive), and keeping the MMC/SD-card with SliTaz always in the card-reader slot. The Eee's BIOS needed to be changed to allow the SD-card to be the first boot device, so that the machine would boot into GRUB before anything else. The `/boot/grub/menu.lst` is setup to allow me to boot from either Xandros or SliTaz on the GRUB menu:

```
# /boot/grub/menu.lst: GRUB boot loader configuration.
#

# By default, boot the first entry.
default 0

# Boot automatically after 8 secs.
timeout 8

# Change the colors.
color yellow/brown light-green/black

# To boot newest slitaz from : /dev/sdb5
#
title  SliTaz GNU/Linux (cooking - kernel 2.6.30.6)
  root (hd0,4)
#  kernel /boot/vmlinuz-2.6.30.6-slitaz rootdelay=10 root=/dev/sdb5
# the kernel line used to be necessary
# but apparently a later upgrade of SliTaz commented it out
# GRUB boots into SliTaz with no problem using only the root (hd0,4)␣
↪command

# To boot Asus eee pc Xandros
#
title Xandros (kernel vmlinuz-2.6.21.4-eeepc)
  root (hd1,0)
```

(continues on next page)

(continued from previous page)

```
  kernel /boot/vmlinuz-2.6.21.4-eeepc quiet rw vga=785 irqpoll i8042.
↪noloop=1 root=/dev/sda1
  initrd /boot/initramfs-eeepc.img
```

The first 3 partitions on my MMC/SD-card were originally used as storage for various parts of the Eee's Xandros system, with the 4th partition formatted as an Extended Partition and further divided into two, with Partition 5 used for the SliTaz install and Partition 6 for Linux-Swap. But because this SliTaz installation has become my main one for the Eee, as I added more packages I finally decided to move parts of the SliTaz filesystem to other partitions and use this layout:

```
$ df -h
Filesystem                Size      Used Available Use% Mounted on
rootfs                    1.4G    413.3M    913.1M  31% /
/dev/root                 1.4G    413.3M    913.1M  31% /
tmpfs                   500.7M         0    500.7M   0% /dev/shm
/dev/sdb2                 3.0G      1.4G      1.6G  46% /usr
/dev/sdb3                 2.5G      2.1G    323.0M  87% /home
/dev/sdb1                 7.4G      5.2G      1.9G  73% /home/shared
```

The `/home/shared` directory is one which has documents shared by the Xandros system.

The Eee-PC was already set-up to recognize and read the MMC/SD-card. But one day I accidentally moved the whole contents of `/boot` to another directory and thus was unable to boot SliTaz. When I inserted the card into a Toshiba Satellite A215-S5850 laptop (my biggest, most modern, and main laptop), I was surprised to find that the machine did not see the MMC/SD-card. It was necessary to add a kernel package:

```
# tazpkg get-install linux-mmc
```

and then to load this module:

```
# modprobe mmc_block
```

After that, SliTaz was able to read the contents of the MMC/SD-card, and I was able to move the `/boot` files back to where they belonged and fix the system. If the output of

```
# lsmod
```

does not show `mmc_block`, then that module must be loaded manually. To have SliTaz always load it at boot-up, simply add it to the `LOAD_MODULES` line of `/etc/rcS.conf`.

2.2.9 Create many-in-1 flavors

author bellard, linea

Slitaz LiveCDs run fully in RAM. The feature set depends on the available RAM size. You can build an auto adaptive LiveCD to ensure that the right feature set is launched according to the detected RAM size. You need to:

- define a flavor list (or an ISO list) where each flavor is nested in the previous one (like russian dolls),
- define the minimum RAM quantity necessary for each flavor.

The many-in-1 ISO may be a little bit larger than the largest flavor. Sometimes the sizes are identical.

We will build the official slitaz-3in1 flavor which boots:

- the *core* flavor with 160MB or more
- the *justx* flavor with 96MB to 160MB
- the *base* flavor with 32MB to 96MB
- and displays an error message with less than 32MB...

The ISO images `slitaz-3.0-3in1.iso` and the largest flavor `slitaz-3.0.iso` share the same size: 31457280 bytes. (caused by the 1MB padding: it would be 123351 bytes larger otherwise).

2.2.9.1 The best way: create a flavor

A flavor holds all the necessary information to build a LiveCD. The build bot[128] will automagically keep the flavor file (`.flavor`) up to date and this tiny file will also be used by **`tazlito`** to (re)generate the ISO image.

flavors[129] are like software sources for the flavor file: the build bot[130] is a make tool and **`tazlito pack-flavor`** the compiler. Providing flavors is like providing source files and providing ISOs is like providing executables only.

A meta flavor has no rootfs, rootcd or packages.list. The `ROOTFS_SELECTION` in the receipt (see http://hg.slitaz.org/flavors/file/324757d594ef/core-3in1) gives the nested flavors according to the ram sizes from the largest to the smallest.

```
ROOTFS_SELECTION="160M core 96M justx 32M base"
```

The rootfs and the rootcd are taken from the largest flavor.

2.2.9.2 The easy way: using tazlitobox

Launch **`tazlitobox`**, select tab *meta*. Enter the ISO file in *ISO input* and add the according RAM size in *RAM needed* for each flavor with the + button. Fill the *ISO output* name and then click *Build ISO*.

Note: In recent SliTaz versions **`tazlitobox`** has moved into **`tazpanel`** in the menu *boot* → *live* → *Build a meta ISO*

2.2.9.3 The cmdline way: using tazlito

Usage: **`tazlito merge size1 iso size2 rootfs2 [sizeN rootfsN]...`**

The largest flavor is given as an ISO file to get the additional rootcd files and will get the resulting ISO.

```
tazlito merge 160M slitaz-core.iso 96M rootfs-justx.gz 32M rootfs-base.gz
```

[128] http://bb.slitaz.org/
[129] http://hg.slitaz.org/flavors
[130] http://bb.slitaz.org/

2.2.9.4 Is it so useful?

OK we can boot the live CD with less RAM but nowadays the computers have a lot of RAM. And booting in text mode is not so sexy…

- booting **live** a (very) old machine with less RAM may be in text mode. But the same PC can have a graphic desktop if SliTaz is installed on hard disk. The SliTaz *base* flavor can install the distribution. Without the russian dolls trick there is no way to use these machines with the default SliTaz cdrom.
- some people maybe don't like the default application set. They can boot a simpler flavor live (say *justx*), add their preferred packages, edit the config files and save the result with **`tazlito writeiso`**

Note: You can do something similar online with http://pizza.slitaz.me/.

2.2.10 QEMU

author stork, linea

2.2.10.1 Introduction

QEMU provides a virtual machine that will run within XP. Refer to the QEMU documentation for a detailed description of the virtual hardware.

It allows to test an operating system environment without actually installing it and without rebooting out of XP; e.g. you can test Slitaz this way, before deciding if you want to install it on your machine.

Compared with Virtual Box (which I am also using regularly), QEMU is initially much faster and easier to set up. You will also have less problems if you are somewhat memory constrained (which you are, if you have less than 1.0 or 1.5 GB RAM). Performance wise, both virtualization platforms looked similar to me (although way slower than a native install).

If you just want to try SliTaz within your XP environment, prefer the QEMU solution described here (unless VirtualBox is already installed and set up on your machine).

2.2.10.2 Getting started

2.2.10.2.1 Requirements

This works on XP with SliTaz 3.0 (should work with all versions of SliTaz).

2.2.10.2.2 Install QEMU

Go to Pendrivelinux.com and download the QEMU PC hardware emulator installer (`QPU804.exe`). This executable is actually a self-extracting archive. When you run it you just need to tell the installer where to unzip the contents (which may be on your hard disk or on a flash drive). The result will be a folder named `QPU804`.

The QEMU setup can execute any ISO that you put in that folder. Copy the SliTaz ISO to that folder. You now have a virtualized SliTaz installation that can run alongside Windows.

2.2.10.2.3 Start QEMU

Execute `QPU804.bat` (double click on it from an explorer window) to start your virtual machine (it will advertise it is running Ubuntu, but really it will run the ISO you copied to that folder, `SliTaz.iso` in this case).

You will see your virtual machine booting from the ISO file, as if it would be booting from a SliTaz LiveCD. Refer to other parts of the SliTaz documentation for a detailed presentation of what you can do from there.

2.2.10.2.4 Getting in and out of the QEMU Window

To move the mouse out of the QEMU screen window press `Ctrl`, keep it pressed and press the `Alt` key simultaneously. To go back, just left click in the window.

2.2.10.2.5 Shared Folder

Open a **`PCManFM`** window: the SliTaz partition labeled "QEMU VVFAT" refers to the same physical (shared) location as the XP "QPU804shared folder" folder.

2.2.10.2.6 Screen resolution

My screen resolution was 800×600. You may want to read the QEMU documentation and play around with the available options from both QEMU and SliTaz to get a higher resolution.

To get a full screen SliTaz, try adding the `-full-screen` option to the qemu command line of the `QPU804.BAT` file.

2.2.10.3 I want to keep the changes I made in SliTaz

2.2.10.3.1 Private data

Copy any private data created in SliTaz to the shared directory: it will survive a virtual machine reboot.

2.2.10.3.2 Keeping system changes

Two ways to do this:

- create a new ISO with your changes, or
- install SliTaz in a qemu virtual drive

Create an ISO file with your system changes

In the SliTaz menu select *System Tools* → *Create a LiveCD* (refer to the LiveCD section of the documentation).

The writeISO panel will create a bootable image of SliTaz with any changes you made, and will store this image file as `/home/slitaz/distro/slitaz.iso`.

In the SliTaz environment, copy-paste that `slitaz.iso` file to the shared folder. Then from XP you will be able to copy the `slitaz.iso` file from the `QPU804\shared` folder to the `QPU804` folder, replacing the existing ISO file (delete the old `.iso` file if it has a different name). Next time you start QEMU, it will boot from your new ISO which brings you back to the same environment.

Install Slitaz in a qemu virtual drive

Alternatively, refer to the QEMU documentation to start QEMU with a "virtual disk" file, where you will be able to install SliTaz by using the menu item *System Tools* → *SliTaz Installer*.

(Not tested with SliTaz but tested with other systems in the past)

2.2.11 VMWare

author jozee, linea

2.2.11.1 Test SliTaz GNU/Linux with VMware

What you will need:

- SliTaz GNU/Linux Virtual Appliance[131] The Virtual appliance file.
- 7-Zip[132] Needed to decompress the file.
- VMware Player[133] Run the virtual machine on your Windows system.

1. Download and install 7-Zip.
2. Download and install the VMware Player.
3. Download the SliTaz GNU/Linux file and decompress it with 7-zip.
4. Now you can run the SliTaz virtual machine (`slitaz.vmx`) with VMware Player.

Note: When starting the virtual machine for the first time, VMware Player detects that files have been moved from another location. Keep the default option in the dialog box and click on the *OK* button.

2.2.12 Virtualbox

author jozee, linea, bellard, magicienap, shreknz, emgi, genesis, mojo

[131] http://dl.free.fr/getfile.pl?file=/Dz6bYtdE/slitaz.7z
[132] http://www.7-zip.org/fr/download.html
[133] http://www.vmware.com/products/player/overview.html

2.2.12.1 SliTaz GNU/Linux with VirtualBox

You can now play with SliTaz GNU/Linux on Windows with VirtualBox. Required:

- SliTaz GNU/Linux Image[134], the virtual machine.
- Oracle VirtualBox[135] software, binaries for Windows.
- 7-Zip[136], needed to decompress the file.

2.2.12.2 Step 1 — Downloads and installation.

- Download and install 7-Zip — necessary to decompress the files.
- Download and install the Oracle VitualBox software.
- Download and decompress the virtual machine somewhere on your local hard drive.

2.2.12.3 Step 2 — Play the virtual machine.

Before running SliTaz GNU/Linux virtual machine you need to configure a new virtual machine in the VirtualBox software.

1. Run virtualbox *Start* → *Programms* → *Oracle xVM VirtualBox* → *VirtualBox*.
2. Add a new Machine: Click on *New* toolbar button.
3. Click *Next* in the "Create New Virtual Machine" dialog box.
4. Give a name to your new virtual machine (i.e: `SliTaz GNU/Linux`), and select `Linux 2.6` in *OS Type* list and click *Next*.
5. Select the amount of memory for the virtual machine. 256MB default should be sufficient.
6. In the "*Virtual Hard Disk*" dialog box click on the *Existing* button to use your SliTaz virtual disk.
7. The Virtual Disk Manager start. Click the *Add* toolbar button, locate and select the VDI file (i.e: `slitaz-1.9-x86.vdi`).
8. Click on *Next* and *Finish*.

If you wish to set up a network using the Host-only adapter, follow these instructions:

1. Click on the *Settings* toolbar button
2. Click on *Network* in the side menu
3. Select the *Network Adapter* (usually *Adapter 1*) and set it for Host-only networking
4. Click on the *Advanced* arrow
5. Change the *Adapter Type* to *PCnet-FAST III*
6. Click on *OK*

[134] http://virtualbox.wordpress.com/2008/06/27/slitaz-gnulinux-is-here/
[135] http://www.virtualbox.org/wiki/Downloads
[136] http://www.7-zip.org/fr/download.html

The default Intel adapter seems to work fine for NAT, but not for Host-only networking.

You can now play with your new SliTaz GNU/Linux virtual machine.

Note: VirtualBox can use VMware virtual disks as well.

Note: If you experience resolution problems (no larger than 800×600) check the following forum post[137]

2.2.12.4 Virtualbox-OSE

You can install the open source edition of virtualbox (free software) in SliTaz with the package **`virtualbox-ose`**:

```
# tazpkg get-install virtualbox-ose
```

This is a restricted version without USB support.

2.2.12.5 SliTaz get-virtualbox

You can install virtualbox (full, but non-free version) in SliTaz with the **`get-virtualbox`** package:

```
# tazpkg get-install get-virtualbox ; get-virtualbox
```

Other virtualization (free software) is also available such as lguest or qemu.

2.2.12.6 Installing Virtualbox Guest Additions

Tip: This procedure is intended for Slitaz 5.0.

This procedure was tested with:

- VirtualBox 4.3.20 installed on Host, including the Extensions Pack;
- slitaz-5.0-rc2.iso dated 20140519;
- slitaz-rolling.iso 32-bit dated 20150201.

See guest additions for SliTaz 4.0 in the forum[138] also.

Virtualbox Guest Additions needs to be manually installed by Slitaz.

Steps:

1. In the SliTaz virtual machine make sure you have a working internet connection. Sometimes you need to stop/restart eth0 using TazPanel (Slitaz Panel) the first time to get a connection.
2. Open Package Manager and recharge the package list.

[137] http://forum.slitaz.org/topic/screen-resolution/page/2
[138] http://forum.slitaz.org/topic/installing-virtualbox-guest-additions-in-slitaz-40-target#post-10791

3. Install the following packages:
   - linux-module-headers (3.2.53)
   - mesa-dri
   - bzip2
4. Mount the VBoxAdditions ISO using the VirtualBox menu option *Devices* → *Insert Guest Additions CD image…*
5. In a terminal, change directory to the mount point for the VBoxAdditions iso, e.g.

```
$ cd /media/cdrom/
```

6. Run as root the following command:

```
# sh ./VBoxLinuxAdditions.run
```

   Ignore text about scripts added to `/etc/init.d` (installer does not recognize your Linux Distribution etc). Ignore text about kernel headers for current running version not found.
7. Open a terminal as root and issue the following commands:

```
# adduser -h /var/run/vboxadd -G daemon -S -s /bin/false vboxadd >/
↪dev/null 2>&1
# addgroup -S vboxsf >/dev/null 2>&1
```

8. Open as root `/etc/rcS.conf` and add "vboxguest vboxsf vboxvideo" to the `LOAD_MODULES` line, e.g.

```
LOAD_MODULES=" vboxguest vboxsf vboxvideo"
```

9. Open as root `/etc/slim.conf` and change the `login_cmd` line to start VBoxClient-all before it executes `~./xinitrc`, e.g.

```
login_cmd VBoxClient-all & exec /bin/sh -l ~/.xinitrc %session</file>
```

   Tip: Instead of editing `/etc/slim.conf`, you could add "VBoxClient-all &" without the quotes to a new line above the `CASE 1$` line in every users `~/.xinitrc` file.

10. Reboot.

To check that the vbox modules are loaded, open a terminal and type:

```
$ lsmod | grep vbox
```

… that should show:

```
vboxguest
vboxsf
vboxvideo
```

(as well as `drm` using `vboxvideo`)

Also, you can install the **mesa-demos** package to get the **glxgears** and **glxinfo** utilities. After you install **mesa-demos**, open a terminal and type:

```
$ glxinfo | grep render
```

… that should show:

```
direct rendering: Yes
OpenGL renderer string: Chromium
```

glxinfo and **glxgears** will show a false error:

```
libGL error: failed to load driver: vboxvideo
```

… which according to the VirtualBox bugtracker is due to the way vboxvideo hooks itself into the Mesa library instead of being loaded in the normal way by Mesa. So ignore it… If you have `OpenGL render = Chromium` and `Direct rendering = Yes` then accelerated 3D is supported.

Clipboard sharing Host<->Guest works if you enable it in the VirtualBox menus or in your virtual machine settings.

Shared folders work (The author manually mounts the shared folders as and when he needs them). For instance, if you wanted to mount the (hypothetical) share called `pubdoc` and access it as a desktop folder called `Docs` (make sure the folder exists first) then you issue the following command in a terminal as root:

```
# mount -t vboxsf pubdoc /home/tux/Desktop/Docs
```

References: Slitaz Forum topic[139]

2.2.13 Web start and possible immediate installation

author linea

2.2.13.1 Special figure in Slitaz!

Slitaz is one of the extremely rare OS's available fully prepared for the web start in live mode using a very small starting program. The starting program can be saved on a old floppy disk or very little special ISO burned on a CD-ROM.

But each starting ISO of Slitaz with the live version of "base", "justX" or "stable" or "cooking" contains also this start program being used through the command line options at starting time of the ISO.

The special advantage of this kind of start is to work with the most actual cooking version and not with the version of the CD.

The same technical ability can of course be used within a closed community.

[139] http://forum.slitaz.org/topic/installing-virtualbox-guest-additions-in-slitaz-50#post-34521

2.2.13.1.1 Where can I find the starting image for a starting floppy disk?

http://mirror.slitaz.org/boot/floppy-grub4dos

Please read more here: http://boot.slitaz.org/

2.2.13.1.2 Where can I find the small ISO for a small starting CD rom?

You can use the little flavor version slitaz-2.0-base.iso[140] for that.

Only enter please "web" as command line option!

2.2.13.1.3 Starting from the net

The start from net don't differ from usual start from the CD.

Only enter "web" as command line option. The starting program don't follow the CD any more but search the more actual file on the web.

2.2.13.1.4 Install Slitaz on the hard disk after the net start

Of course here is an important difference compared with the usual start:

There is no CD in the drive where the installation program can find the row files.

...

2.2.13.1.5 Regular net start in my community net (school etc.)

Install the adequate Slitaz version on the own server

...

Limit access if required

...

Prepare an adequate ISO for a own starting check card mini CD

...

2.2.14 Boot tree

author linea, bellard

[140] http://mirror.slitaz.org/iso/2.0/flavors/slitaz-2.0-base.iso

2.2.14.1 From BIOS to `/etc/init.d/rcS`

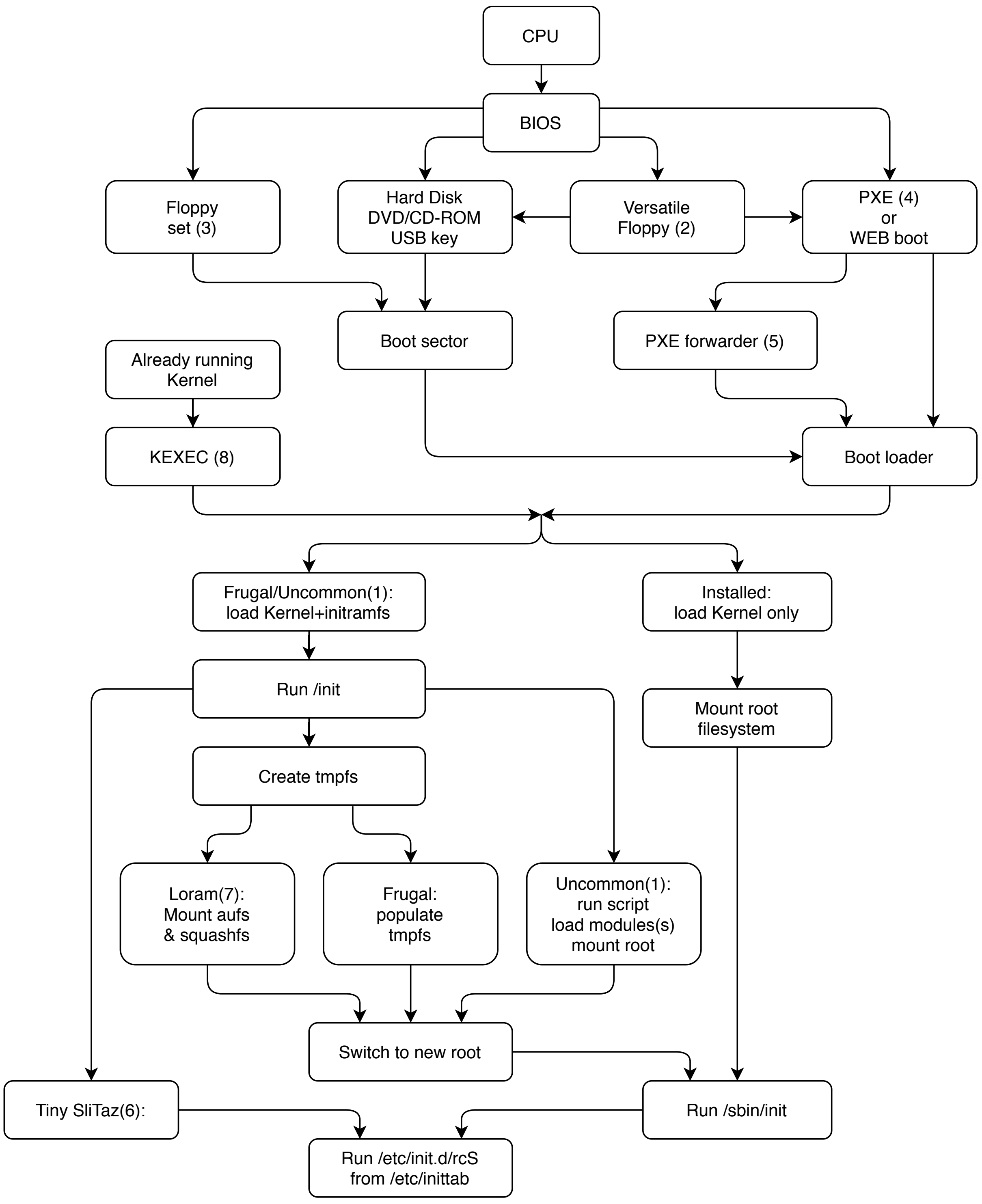


Fig. 1: Boot tree

1	*Uncommon* (page 116) is LVM, RAID, crypto, loop or subdir mount with preinit rootfs.
2	Versatile boot floppy[141]
3	Boot floppy set[142].
4	LAN *PXE* (page 104) or WEB boot[143].
5	*gpxe.pxe* (page 107).
6	Tiny slitaz, see http://tiny.slitaz.org/
7	*Lowram CD* (page 110).
8	**`kexec`** command from the kexec-tools[144] package.

2.2.14.2 SliTaz ISO image boot tricks

The CD-ROM image has a hybrid format from version 5.0.

- It boots from a CD-ROM drive according to the *El Torito* specification as usual (BIOS or UEFI)
- It boots from a memory card / USB key using the syslinux hybrid format (BIOS) or using a FAT partition mapped in the ISO image (UEFI).
- It launches a USB boot key creation utility from Windows (32 bits).
- It can boot from DOS (real mode or virtual 8086 with EMM386) directly:

```
C:\> ren slitaz.iso slitaz.exe
C:\> slitaz.exe
```

Tip: The files `bzImage`, `memtest` and `ipxe` can boot with DOS too:

```
C:\> ren bzimage bzimage.exe
C:\> bzimage.exe root=/dev/hda3 autologin

C:\> ren memtest memtest.exe
C:\> memtest.exe

C:\> ren ipxe ipxe.exe
C:\> ipxe.exe http://myserver.org/boot.php
```

- It is easily customizable with the **`iso2exe`** (compatible with **`taziso`** or *tazpanel* → *boot* → *ISO mine*) tool:

```
iso2exe -a "lang=fr_FR kmap=fr-latin1 tz=Europe/Paris" -i myconfig.gz␣
↪slitaz.iso
iso2exe -l slitaz.iso
iso2exe -r slitaz.iso custom.append custom.initrd
taziso slitaz.iso getcustomconf
taziso slitaz.iso isomd5
```

Tip: You can tune the boot process with your own `/init` script:

```
#!/bin/sh

sed -i 's,^RUN_DAEMONS=",RUN_DAEMONS="dropbear ,' /etc/rcS.conf
cat >> /etc/init.d/local.sh <<EOM
/my/special/inits.sh
EOM
exec /init "$@"
```

Update the kernel command line:

[141] http://mirror.slitaz.org/boot/floppy-grub4dos
[142] http://mirror.slitaz.org/floppies/
[143] http://boot.slitaz.org/
[144] http://pkgs.slitaz.org/search.sh?package=kexec-tools

```
iso2exe -a "rdinit=/myinit lang=fr_FR kmap=fr-latin1 tz=Europe/Paris"␣
↪-i myconfig slitaz.iso -f
```

2.2.15 Persistence & splash

author linea, brianperry

This document details how you can add persistence to your live SliTaz distro, and how you can modify the splash screen. There are different ways on how you can do add persistence (see How to make Slitaz USB persistent[145]). Unless you decide not to keep the **SLiM** login manager in SliTaz however, all methods first require you to have the live SliTaz distro placed on a media (USB stick, . . .) that is formatted in ext2 or ext3 (if you don't want to keep the **SLiM** login manager, FAT32 can also be used instead). FAT32 has the advantage that you can view/mount the media in Windows too, and you can format it in Windows simply by means of the windows explorer application. If you don't need this functionality though, you might want to use ext2 or ext3 however, as it is a more secure filesystem.

2.2.15.1 Method 1

By typing `slitaz home = sdxx` or `slitaz home=UUID` at the command line at the splash/boot screen. See *Slitaz Parameters* (page 276). This option hence involves typing in this line every time you want persistence. It's also only useful when keeping the SLiM login manager in place, as you can't specify to (immediately) log in as user or root (either with or without persistence).

The `home=sdxx` or `home=UUID` basically just tells SliTaz which disk it is where it needs to write the system changes to (which are being written away as a `rootfs.gz` file, see below). It should btw be noted here that `home=UUID` is preferred over `home=sdxx` or even `home=usb` (the latter is the same as `home=sda1`). This, as depending on which drives you connect at any given time, your drive name may change, which will give problems (this happens as depending on the access speed of the drive (speed depending on the type of drives connected/inserted, i.e. SATA drive, USB drive, . . .), Linux will assign a different drive name). Then, to effectively write away the changes, you need to type in

```
# tazusb writefs gzip
```

(for fast compression, average filesize),

```
# tazusb writefs lzma
```

(for slow compression, smaller filesize) or

```
# tazusb writefs none
```

(for no compression, larger filesize). Each of these will write everything to a `rootfs.gz` archive to be loaded the next time you boot.

One other way to write away the changes are to right-click on desktop *SliTaz Live → TazUSB Writefs (gzip)*. With this latter method, you will however then also need to rename the `rootfs.gz` file (i.e. to `rootfs5.gz`) and move the file from the root folder (`/`) to the `/boot` folder on your USB stick drive. Finally, you need to manually add this file to `syslinux.cfg` (or `extlinux.conf`, see below).

[145] https://superuser.com/questions/123399/how-to-make-slitaz-usb-persistent

2.2.15.2 Method 2

Another way to tell SliTaz the disk it needs to write to is by simply modifying or adding an entry to the `extlinux.conf` file (which is located at `/home/boot/extlinux`). See also How to make changes persistent on a usb key live installation?[146]. This method is the only method that doesn't require typing in commands every time you boot and require persistence (more automated). Note that the same method can also be followed when you have a FAT32/non-SLiM login manager setup, however, you will then need to do the changes in the `syslinux.cfg` file (located at `/home/boot/syslinux`), see How to make Slitaz USB persistent[147], and not `extlinux.conf`. To change the `extlinux.conf` or `syslinux.cfg` (which one you need to alter will depend on the filesystem you use), you need to pick one of the SliTaz entries you're currently not using at the boot screen (i.e. *base*, *justx*, … entry), and then modify it. Alternatively, you can make an entirely new label as well (see How to make changes persistent on a usb key live installation?[148]. Make it look like this:

```
LABEL my slitaz
MENU LABEL My slitaz or "whatever you like"
KERNEL /boot/bzImage
APPEND initrd=/boot/rootfs4.gz,/boot/rootfs3.gz,/boot/rootfs2.gz,/boot/
↪rootfs1.gz rw root=/dev/null vga=normal autologin home=UUID lang=en_GB␣
↪kmap=uk
```

Note that the UUID will be a range of numbers which you can copy from another entry, or which you find by looking at the properties of the (removable USB) drive you want to write to.

Note that you can also make both a root and a user account (each with or without persistence, so 4 accounts in total). This can be done by adding the `home=UUID` or by leaving it out (no persistence).

2.2.15.3 Method 3

The last method is done by simply typing in some commands at the terminal (see Slitaz on USB / persistent[149]. So, after booting into SliTaz, just bring up the terminal, log in as root, and then type:

```
# blkid
```

Edit your boot code so `home=UUID`

```
# blkid
/dev/sda1: UUID="2c55c420-760a-4fa3-871b-64191dcc338a" TYPE="ext2"
root@slitaz:~# cat /proc/cmdline
initrd=/boot/rootfs4.gz,/boot/rootfs3.gz,/boot/rootfs2.gz,/boot/rootfs1.gz␣
↪rw root=/dev/null
vga=normal autologin home=2c55c420-760a-4fa3-871b-64191dcc338a BOOT_IMAGE=/
↪boot/bzImage
```

Important: Having read the *Modify the isolinux configuration* (page 356) page, I seem to understand that what was the "main boot loader" was actually the **isolinux** boot loader, which in turn starts **syslinux** (with FAT32 USB sticks). With ext-formatted drives, **syslinux** itself then also starts **extlinux** before that boot loader can load SliTaz.

[146] http://forum.slitaz.org/topic/how-to-make-changes-persistent-on-a-usb-key-live-installation
[147] https://superuser.com/questions/123399/how-to-make-slitaz-usb-persistent
[148] http://forum.slitaz.org/topic/how-to-make-changes-persistent-on-a-usb-key-live-installation
[149] http://forum.slitaz.org/topic/slitaz-on-usb-persistent

Is this correct? If so, update the text accordingly above. One really need to now this to know what files to edit. Also, is it possible to leave out booting the **syslinux** boot loader completely when booting a SliTaz LiveUSB stick formatted in ext3? If possible, I suppose one can just alter the **isolinux** files to accomplish this, and then make 2 versions of the customised slitaz distro if one wants to; knowlingly

- one for FAT32-formatted sticks (**isolinux** booting **syslinux** which then boots SliTaz)
- one for ext-formatted sticks (**isolinux** booting **extlinux** which then boots SliTaz)

Important: I looked at the ext3-based SliTaz 5 version I made a few days ago to check whether there is indeed an **extlinux** as well as an **syslinux** folder at `/boot/` (and see whether there are any **isolinux** files in `/`). I didn't find a `syslinux` folder though (which would imply it automatically loads **extlinux** after loading the main boot loader **isolinux**). The only folder I found was an `extlinux` folder at `/boot/`. Other files at `/boot/` were `bzImage`, `rootfs.gz`, `rootfs2.gz`, `rootfs3.gz`, `rootfs4.gz`. So besides not finding a **syslinux** boot loader, I also didn't find any **isolinux** files at `/`, but I did found an **isolinux** file at `/boot/extlinux`, confirming that this **isolinux** boot loader is indeed the first boot loader that is started.

In the `/boot/extlinux` folder, I found both the `isolinux.cfg` file mentioned above, as well as `extlinux.conf`. I added in the files in annex as texts file. I'm not sure but this would hence seem to imply that there is also no `/boot/syslinux/` folder made at all (for any SliTaz version, fat32 or ext3). I'm not sure on this however.

2.2.15.4 Rolling back

If anything should go wrong when writing your filesystem, you can simply rollback to your previous filesystem by typing `previous` at the `boot:` prompt. Older backups are named `rootfs.gz.`*`unixtimestamp`* and can be safely deleted from the `/home` folder to save disk space using **tazusb clean**.

2.2.15.5 See also

- *Modifying isolinux* (page 356) (**isolinux** appearantly being the "main boot loader" which starts **syslinux**, **syslinux** itself starting **extlinux** if you have an ext-formatted USB drive with SliTaz on it.

2.2.15.6 Additional links/references

- http://forum.slitaz.org/topic/slitaz-40-liveusb-writefs
- http://forum.slitaz.org/topic/a-couple-of-newbie-questions

2.3 Networking / LAN

2.3.1 Wi-Fi configuration, easy method

author linea, pankso, jozee

2.3.1.1 Introduction

To install Wi-Fi under Linux, two methods are possible:

- Use the kernel module specific to your Wi-Fi card
- Use the **`Ndiswrapper`** module to install a Windows driver

Here, we will discuss the usage of a kernel module specific to your card (**`Ndiswrapper`** will one day have another page on the wiki).

To use a specific kernel module, you'll need to:

1. Know your Wi-Fi hardware
2. Know which module your card needs
3. Install this module, and possibly firmware too
4. Configure your connection to the access point
5. Get connected and surf

`Netbox` is a tool which can do that just by clicking a few buttons. But it's also possible to configure your Wi-Fi from the command line.

2.3.1.2 `Wifibox`/`Netbox` Graphical Utility

With **`netbox`** or **`wifibox`** you can configure your Wi-Fi with only a few mouse clicks. As you already know which module you need, it's easy to use **`wifibox`**:

```
$ subox wifibox
```

`Wifibox` will install all the software needed and then launch the Wi-Fi. You won't have to manually install your Wi-Fi.

Here are the necessary steps for **`Wifibox`**:

- First, go to the *Drivers* tab, and install the module you need. SliTaz will load the firmware (if needed), the kernel module, configure your Wi-Fi, and connect to the access point.
- Then, go to *Configuration*, and start to configure. If you don't know what to add, have a look at the sample `/etc/network.conf` — **`wifibox`** uses the same parameters. Click on *Start*
- If all goes well, you may start surfing!

2.3.1.3 Manually Configure your Wi-Fi

If you want to understand how Wi-Fi works in SliTaz, then you can try configuring your Wi-Fi manually.

But if you want to know how Wi-Fi generally works under Linux, or if your kernel module is not on this list, you should look at the *Wi-Fi, step by step* (page 154).

Here is a quick summary of the steps needed:

- Know your Wi-Fi card
- Check and install if your Wi-Fi card needs any firmware, e.g., `b43`
- Load the kernel module specific to your Wi-Fi card

- Check that your Wi-Fi card is detected and your module is loaded
- Configure `/etc/network.conf`
- Load up your `WIFI` interface
- Start `/etc/init.d/network.sh`

The following commands do all of the above steps. This is also the easiest way:

```
# tazhw detect-pci --get-firmware
# /etc/init.d/network.sh restart
```

Now for more detailed instructions:

2.3.1.4 Detailed Instructions

2.3.1.4.1 Identifying your hardware (Which Wi-Fi card do I have?)

You can list your hardware using the terminal. It's useful to know which Wi-Fi card you have.

If you have an integrated Wi-Fi card:

```
$ lspci | grep -i network
```

If your card is an USB one, you'll need **`lsusb`** which is available in the package **`usbutils`**:

```
# tazpkg get-install usbutils
$ lsusb
```

In either case, you'll see something like this:

```
02:02.0 Network controller: Intel Corporation PRO/Wireless LAN 2100 3B␣
↪Mini PCI Adapter (rev 04)
```

This tells us the following:

- `Intel Corporation` made the card
- The card is a `Pro/Wireless Lan 2100 3B`
- The chipset is a `IPW 2100 (Intel Pro Wireless 2100)`
- The interface is a `Mini PCI`

2.3.1.4.2 Which module do I need, and do I need firmware?

The SliTaz Linux kernel is made to be light. Some software modules, especially those needed by Wi-Fi cards are not installed by default, but have to be loaded by the user.

Many Wi-Fi cards will work if you load the correct module. The easiest way is to let SliTaz auto-detect your hardware.

For integrated Wi-Fi cards:

```
# tazhw detect-pci
```

For USB cards:

```
# tazhw detect-usb
```

Sometimes, however, the module alone is not enough. Some types of card (Intel for example) also need firmware. Such firmware is not free software, so we can't distribute it as part of SliTaz. You may need to obtain the firmware from the website of your card manufacturer and download it to `/lib/firmware`. But you'll see that SliTaz can sometimes do this for you!

2.3.1.4.3 For `b43`, `b43legacy`, `ipw2100` or `ipw2200` kernel modules

If the kernel module you need is either `b43`, `b43legacy`, `ipw2100` or `ipw2200`, you also need to install the package: **`get-wifi-firmware`**. You can install it like this:

```
# tazpkg get-install get-wifi-firmware
```

2.3.1.4.4 Launch Wi-Fi

`get-wifi-firmware` will install some commands named `get-`*`a module`*`-firmware`. To see the list:

```
# ls /usr/bin/get-*-firmware
```

Launch the software that corresponds to your module (`ipw2100` in the earlier example):

```
# get-my_module-firmware
```

For example, say your module is `ipw2200`, you can type:

```
# get-ipw2200-firmware
```

This command will:

1. Get the needed firmware for "*my_module*", make the package *`my_module`*`-firmware`, and install it.
2. Get useful software for Wi-Fi support (**`iwconfig`**, **`wpa_supplicant`** if needed. . .)
3. Load the module "*my_module*" into the Linux kernel.
4. Launch **`/etc/init.d/network.sh restart`**, which starts Wi-Fi.

If `/etc/network.conf` is correctly configured, you can surf!

The easiest way is to do this is to download the firmware graphically using **`Wifibox`** (SliTaz Wireless Manager). Yes, SliTaz graphical wireless manager (**`wifibox`**) can do these steps on the *Driver* tab (after selecting the correct module, say `ipw2200` and pressing *install*).

2.3.1.5 Troubleshoot your Wi-Fi

Here is a quick summary of the steps needed (Repeated again for understanding):

- Know your Wi-Fi card

- Check and install if your Wi-Fi card needs any firmware, e.g., `b43`
- Load the kernel module specific to your Wi-Fi card
- Check that your Wi-Fi card is detected and your module is loaded
- Configure `/etc/network.conf`
- Load up your `WIFI` interface
- Start `/etc/init.d/network.sh`

The following commands do all of the above steps. This is also the best way to troubleshoot. To get maximum help on the forums, please post the output of each of these commands.

```
lspci | grep -i network
modprobe your_module
dmesg | tail
lsmod
nano /etc/network.conf
ifconfig eth1 up
ifconfig -a
iwconfig
/etc/init.d/network.sh restart
ifconfig -a
```

2.3.1.6 Configure `/etc/network.conf`

SliTaz launches Wi-Fi using the script `/etc/init.d/network.sh`. This script uses the config file `/etc/network.conf`. You should first edit your config file using the instructions in the sample `/etc/network.conf`. You'll find more information about `/etc/init.d/network.sh` in *Secrets of /etc/init.d/network.sh* (page 157)

Now your module must be loaded at each boot. To automate, add "*my_module*" to the line `LOAD_MODULES` in `/etc/rcS.conf`:

```
# geany /etc/rcS.conf
```

```
LOAD_MODULES="nls_utf8 my_module"
```

2.3.1.7 Sample `/etc/network.conf` file with comments

```
# Start Wi-Fi on boot: "yes" or "no".
WIFI="yes"

# Wi-Fi interface (usually "wlan0" or "eth0").
WIFI_INTERFACE="wlan0"


# ESSID of access point: "my_essid" or "any".
# If "any" is chosen, SliTaz will try to connect to the first access point.
# Be careful:
# In some states (in France for example), you are not allowed to connect
# to a private access point if it is not secured.
# WIFI_ESSID="any"
```

(continues on next page)

(continued from previous page)

```
# Type of connection:
# You can chose between:
# * ad-hoc    : Connect to one cell without access point.
# * managed   : One or more cell, with access point.
# * master    : Your card is a master card.
# * repeater  : Your card acts as a repeater.
#               Useful for long distances.
# * secondary : Your card is a backup for master or repeater
# * monitor   : Your card only receives messages.
# For most of the time, you'll use "managed".
# (Home, cyber-coffe, work, university...)
WIFI_MODE="managed"

# Key
WIFI_KEY="ma8clef8de8chiffrement8difficilement8déchiffrable"

# Key type "wep" or "wpa" or "any" or "none"
# If you're using WPA-EAP (at work for example), try "any".
WIFI_KEY_TYPE="wpa"

# Driver needed by wpa_supplicant.
# It depends on your kernel module.
# The possible drivers are:
# * wext    : Linux wireless extensions (in most cases, use this one.)
# * hostap : Host AP driver (Intersil Prism2/2.5/3)
# * atmel  : ATMEL AT76C5XXx (USB, PCMCIA)
# * wired  : wpa_supplicant wired Ethernet driver
WPA_DRIVER="wext"

# Wi-Fi channel. Leave it blank, if you don't know what it is.
WIFI_CHANNEL=""

# More args to pass to iwconfig.
# Look at iwconfig man page for more information.
WIFI_IWCONFIG_ARGS=""
```

2.3.1.8 Extra Information

Now for more detailed instructions:

First do **`lspci`** to know which card you have. Once we know which card you have, we can surf the internet to find which module you require, and if we need any special firmware. To help you, here are some useful links:

- Linux wireless LAN support[150] (The column on the right, tells you where to get the firmware, if applicable)

[150] http://linux-wless.passys.nl/

- Ubuntu documentation[151] (Good Ubuntu website, Wi-Fi page)
- Google[152] with "*the name of your Wi-Fi card*" + "*modprobe*" or "*linux*"

If these links are not useful, you can ask at the forum[153]

So we learn that the `Intel Corporation PRO/Wireless LAN 2100 3B Mini PCI Adapter` works with the `IPW2100` module and the firmware is available at http://ipw2100.sourceforge.net/.

2.3.2 Wi-Fi, step by step

author linea, jozee

2.3.2.1 Introduction

If you want to use your Wi-Fi right away, this page is not for you. You should look at the *Wi-Fi configuration, easy method* (page 148) tutorial which explains how to use the tools given by SliTaz. But if you want to know how to use Wi-Fi (under Linux), this page will explain how, and help you to configure it *from scratch*. Before continuing:

- You should know the kernel module needed by your Wi-Fi card.
- If you need any firmware; you should know where to download it.

We are going to:

1. Install any useful software and firmware.
2. Load the kernel module.
3. Configure the Wi-Fi connection.
4. Configure WPA.
5. Get connected and surf.
6. Shutdown Wi-Fi.

Note: On SliTaz, the script `/usr/bin/get-wifi-firmware` takes care of steps 1 and 2, and the script `/etc/init.d/network.sh`; steps 3 to 6.

2.3.2.2 Install needed software

You'll need the Wi-Fi kernel modules and some software to manage Wi-Fi. If you are using a WPA key, you'll also need **`wpa_supplicant`**:

```
# tazpkg get-install linux-wireless
# tazpkg get-install wireless_tools
# tazpkg get-install wpa_supplicant
```

[151] http://help.ubuntu.com/
[152] http://www.google.com/
[153] http://forum.slitaz.org/

If you need firmware:

```
# cd /lib/firmware
# wget http://www.address/of/my/firmware
```

Untar to install:

```
# tar -xvf my_firmware.tar*
# rm my_firmware.tar*
```

2.3.2.3 Load the kernel module

```
# modprobe -v my_module
```

If you've got errors, verify that your firmware is where it should be and look at **`dmesg`**:

```
$ ls -l /lib/firmware
$ dmesg
```

If you don't have any errors, you can continue.

2.3.2.4 Configure Wi-Fi interface

Before configuring a new interface, you should de-configure the old one. If your ethernet interface is configured, you should:

```
# ifconfig eth0 down
```

`iwconfig` allows you to configure your Wi-Fi card, so that it can connect to your access point. You need to know the name of your Wi-Fi interface (usually `wlan0` or `eth1`). If you don't know its name, just run **`iwconfig`**:

```
# iwconfig
```

Now we can configure your Wi-Fi interface and start it:

```
# ifconfig WIFI_INTERFACE up
# iwconfig WIFI_INTERFACE txpower on
```

Let's test that the card works:

```
# iwlist scan
```

If you've got a list of access points you can now tell your Wi-Fi interface which ESSID to connect to:

```
# iwconfig WIFI_INTERFACE essid MY_ESSID
```

`iwconfig` can also accept others args, look at its man page to know more.

2.3.2.5 Configure a WEP or WPA key

You can easily configure a WEP key with **`iwconfig`**:

```
# iwconfig WIFI_INTERFACE key my_wep_key
```

But you should *always* use a WPA key, because WEP keys can be easily cracked with **aircrack**, as noted here[154], **wpa_supplicant** allows you to use a WPA key (some cards may use WPA without **wpa_supplicant**). It needs a config file. Usually, `/etc/wpa_supplicant.conf`. If you are using `wpa_psk` (normally, you are), add this to the file:

```
ap_scan=1
network={
     ssid="my_essid"
     scan_ssid=1
     proto=WPA
     key_mgmt=WPA-PSK
     psk="my_clear_key"
     priority=5
}
```

Or try:

```
ap_scan=1
network={
     ssid="my_essid"
     scan_ssid=1
     key_mgmt=WPA-EAP WPA-PSK IEEE8021X NONE
     group=CCMP TKIP WEP104 WEP40
     pairwise=CCMP TKIP
     psk="my_clear_key"
     priority=5
}
```

It's now possible to launch **wpa_supplicant**:

```
# wpa_supplicant -B -w -c/etc/wpa_supplicant.conf -DWPA_DRIVER -iWIFI_
↪INTERFACE
```

`WPA_DRIVER` is the name of the driver used by **wpa_supplicant**. Usually, it's `wext`, but sometimes, another is needed. Here is a list of possible drivers:

`wext`	Linux wireless extensions (generic, should work in most cases)
`hostap`	Host AP driver (Intersil Prism2/2.5/3)
`atmel`	ATMEL AT76C5XXx (USB, PCMCIA)
`wired`	**wpa_supplicant** wired Ethernet driver

The option `-B` launches **wpa_supplicant** as a daemon. If you want to kill it:

```
# killall wpa_supplicant
```

2.3.2.6 Get connected

If you want to connect in DHCP, just run:

[154] http://www.tuto-fr.com/tutoriaux/tutorial-crack-wep-aircrack.php

```
# /sbin/udhcpc -b -i WIFI_INTERFACE -p /var/run/udhcpc.WIFI_INTERFACE.pid
```

Normally, you should be surfing!

2.3.2.7 Turn off Wi-Fi

To stop Wi-Fi, you should shutdown your Wi-Fi card, and stop the **wpa_supplicant** and UDHCPC daemons:

```
# iwconfig WIFI_INTERFACE txpower off
# kill `cat /var/run/udhcpc.WIFI_INTERFACE.pid`
# killall wpa_supplicant
```

You can also unload the kernel module:

```
# rmmod my_module
```

2.3.3 Secrets of /etc/init.d/network.sh

author pankso, linea, mojo

2.3.3.1 Introduction

SliTaz launches the /etc/init.d/network.sh at startup to initialize the network. It configures the hostname, loopback interface, and internet connection.

It's also possible to call the script when SliTaz is started, and use it to open or close internet connections. For example netbox and any software linking **get-wifi-firmware** (**get-ipw2100-firmware**, **get-b43-firmware**…) use it.

2.3.3.2 Usage

By default, /etc/init.d/network.sh uses /etc/network.conf as the conf file. Parameters written in that file are made for the default network connection.

To start the default connection, as when booting:

```
# /etc/init.d/network.sh start
```

The start arg should be used only at boot. To stop the connection:

```
# /etc/init.d/network.sh stop
```

To stop and restart:

```
# /etc/init.d/network.sh restart
```

But, most interestingly, is that /etc/init.d/network.sh may also use another config file. It's useful if you're using a laptop, as you can configure multiple connections for multiple access points.

For example, we can create a directory /etc/network, containing some config files, named:

- Home, for home, using an ethernet connection and a static IP.
- Desktop, for the desktop, with a WEP encryption, and a static IP.
- Univ, Wi-Fi without encryption, and with DHCP.

Now to get connected at Home, later at a Desktop and finally at Univ, before stopping connection, we only have to:

```
# /etc/init.d/network.sh restart /etc/network/Home
# /etc/init.d/network.sh restart /etc/network/Desktop
# /etc/init.d/network.sh restart /etc/network/Univ
# /etc/init.d/network.sh stop
```

2.3.3.3 sudo

Since /etc/init.d/network.sh can only be used by root, if you want a normal user to use it, you should install **sudo**:

```
# tazpkg get-install sudo
```

And then configure it:

```
# visudo
```

For user tux, which must use network.sh from every host; without a password you should add:

```
tux  ALL=NOPASSWD: /etc/init.d/network.sh,
```

For user tortux, which may only get connected from localhost, and which should use a password each time, you should add:

```
tortux my_hostname=PASSWD: /etc/init.d/network.sh,
```

If you forgot your hostname, just run:

```
$ cat /etc/hostname
```

Here is some help to use **visudo**:

- i insertion mode (to write).
- Escape exit insertion mode.
- :wq record and quit.
- :q! quit without recording.

2.3.3.4 Openbox

All this is not that really user friendly…

That's why I'll give you a perfect treat: A way to integrate all this in an Openbox menu! Create a script /usr/lib/openbox/network-menu.sh, and add this to it:

```
#!/bin/sh
#
# Openbox pipe menu to start network connections
# (This script is only useful if sudo is installed, and correctly␣
↪configured)

echo '<openbox_pipe_menu>'

# For default file:
echo '<item label="Load Default network.conf">'
echo -n '<action name="Execute"><execute>'
echo -n "sudo /etc/init.d/network.sh restart"
echo '</execute></action>'
echo '</item>'

# For other configuration files (you may state a different directory here
# depending on your setup):
# e.g nice for frugal installs:
# ls /home/tux/network/ | while read; do
ls /etc/network/ | while read; do
echo '<item label="'"${REPLY}"'">'
echo -n '<action name="Execute"><execute>'
echo -n "sudo /etc/init.d/network.sh restart '/etc/network/${REPLY}'"
echo '</execute></action>'
echo '</item>'
done

# To stop connections:
echo '<item label="stop Connection">'
echo -n '<action name="Execute"><execute>'
echo -n "sudo /etc/init.d/network.sh stop"
echo '</execute></action>'
echo '</item>'

echo '</openbox_pipe_menu>'
```

Make it executable:

```
# chmod +x /usr/lib/openbox/network-menu.sh
```

And now you only have to add these lines in ~/.config/openbox/menu.xml:

```
<menu id="network-menu" label="Network"
     execute="/usr/lib/openbox/network-menu.sh" />
```

Then reconfigure Openbox:

```
$ openbox --reconfigure
```

Enjoy!

2.3.4 Virtual Private Network

author jozee, linea, bellard

A VPN binds isolated networks with tunnels over the Internet network. A tunnel encrypts and encapsulates network frames.

- frames are hidden (data and header, including routing info)
- encapsulation sets tunnel ends as source and destination
- only traffic volume is visible. But you can send dummy frames

2.3.4.1 Tunnel using SSH and PPP

SliTaz can build such tunnels out of the box (without additional packages) with menu *System* → *Netbox*. You need on remote server:

- a user account
- a SSH access
- PPPD launch rights (group rights or *setuid* bit)
- remote stations know how to route frames to the tunnel (peer is the default router or is running routing protcols such as rip, ospf, . . .)

2.3.4.2 Tunnel setup

- launch *System* → *Netbox*
- select *VPN* tab
- fill *Peer* and update *Route(s)* fields. Routes are the local area networks on peer side you want to reach. You only need to update *Local IP* and *Remote IP* when you create multiple tunnels. They are tunnel end IP points
- click *Send Key* and enter password
- click *Connect*

2.3.4.3 Limitations

- This tunnel should not be used for real time traffic like VoIP. Based on TCP, the tunnel tries to avoid data loss instead of respecting timing. VoIP should use UDP based tunnels
- Only the SliTaz box can see the remote network. Other stations on the LAN can't, until you add your LAN routes to peer (assuming SliTaz is the default router or is running routing protcols)

2.3.4.4 OpenSSH

The **openssh** package is available on the mirror and provides TCP based tunnels:

```
# yes y | tazpkg get-install openssh
```

See http://man.openbsd.org/ssh#SSH_BASED_VIRTUAL_PRIVATE_NETWORKS

2.3.4.5 OpenVPN

The **openvpn** package is available on the mirror:

```
# yes y | tazpkg get-install openvpn
```

See documentation at http://openvpn.net/

2.3.4.6 Cisco EasyVPN

The **vpnc** package is available on the mirror:

```
# yes y | tazpkg get-install vpnc
```

See documentation at http://www.unix-ag.uni-kl.de/~massar/vpnc/

2.3.4.7 Others VPN

SliTaz packages search tool shows every VPN supported by SliTaz: http://pkgs.slitaz.org/search.sh?tags=vpn

2.3.5 3G-UMTS

author kultex, linea, jozee

The # sign means that you do this command as root on your console:

2.3.5.1 Automatic Installation

The new **hwsetup** in SliTaz 3.0 now adds automatic installation for 3G (still in trial version).

```
# tazhw setup 3g-modem
```

Now open the **wvdial** box by typing `y`, enter your data (as described in `wvdial.conf`), save the configuration (if you don't save you risk breaking your *pin*), dial the *pin* (you have to do this only once you plug in your modem), then start dialing and you should be connected.

2.3.5.2 Manual Installation

```
# tazpkg get-install wvdial
# tazpkg get-install linux-dialup
```

You have to bring down your network, otherwise you will have a route problem:

```
# ifconfig eth0 down
# ifconfig eth1 down
```

You have to switch on the modem with:

```
# modprobe option  # (Huawei E220, E160g, Venus-VT12)
```

Please help update which modem or phone is using which module using the forum post: http://labs.slitaz.org/issues/show/149. You will find if your modem is supported by **`linux-dialup`**.

There is an extra guide for the *Cricket-a600* (page 164)

For automatization load the module at startup with your SliTaz Control Box.

2.3.5.3 `wvdial.conf`

Check the modem if it is `/dev/ttyUSB*` or `/dev/ttyACM*` (e.g. with **`# dmesg`**) and edit your `wvdial.conf` (here as an example):

```
# leafpad /etc/wvdial.conf
```

```
[Dialer pin]
Modem = /dev/ttyUSB0
Init1 = AT+CPIN=1234                     # here's your pin

[Dialer umts]
Modem = /dev/ttyUSB0
ISDN = off
Modem Type = USB Modem
Baud = 460800
Init = ATZ
Init2 = ATQ0 V1 E1 S0=0 &C1 &D2 +FCLASS=0
Dial Prefix =
Dial Attempts = 1
Dial Command = ATM1L3DT
Ask Password = off
Auto Reconnect = off
Abort on Busy = off
Carrier Check = on
Check Def Route = on
Abort on No Dialtone = on
Stupid Mode = off
Idle Seconds = 0
Init3 = AT+CGDCONT=1,"IP","drei.at"      # here's your string
Username = drei.at                       # here's your Username
Password = drei.at                       # here's your Password
Phone = *99#                             # here's your Phone
```

All the needed data can be found here:

- Austria, Germany, Switzerland[155]
- World[156]

But you can search the net with the name of your provider and `wvdial.conf` and I think you will need to.

To connect do:

[155] http://linux.frankenberger.at/Huawei_E220_Daten.html
[156] http://www.flexispy.com/Mobile%20APN%20Setting%20to%20use%20GPRS.htm

```
# wvdial pin
# wvdial umts
```

You can put the *pin* also in `Dialer umts`, but I use my modem most of the time on the train, and you would get an error when you reconnect (when the connection breaks) — so use **`wvdial`** *pin* only the first time you plug in the modem.

The only thing thats left is to put the DNS in `resolv.conf` (you will find the DNS in the two lists with all the data).

```
# leafpad /etc/resolv.conf
```

```
nameserver 213.94.78.16                   # here's your DNS
```

If you want to dial out as user — add yourself to the group "dialout" (**`cat /etc/group`**) and set the permissions for `wvdial` and `wvdial.conf`.

For automatization you can put a script in `/usr/local/bin` called `umts-connect` (executable)! From here it's not proved, because I connect from the console, but it should work like this!

```
#! /bin/bash
ifconfig eth0 down
ifconfig eth1 down
wvdial pin
wvdial umts
```

But don't reconnect with this.

2.3.5.4 Desktop entry

You can also put a desktop entry in `/usr/share/applications` with the name `umts.desktop`

```
[Desktop Entry]
Encoding=UTF8
Name=UMTS connection
Name[de]=UMTS-Verbindung
Comment=UMTS-Verbindung
Type=Application
Exec=/usr/local/bin/umts-connect
Icon=/usr/share/icons/...          # whatever you want
Categories=Application;Network;
```

Please check `Exec=/usr/local/bin/umts-connect` for permissions.

Here is the forum post[157] about this.

2.3.6 Hardware Guides: Modems

author jozee

Setting Up The Cricket A600 Broadband Modem (page 164) — How to install a Cricket a600 Broadband Modem

[157] http://forum.slitaz.org/index.php/discussion/comment/440/#Comment_440

2.3.7 Setting Up The Cricket A600 Broadband Modem

author jozee, linea

2.3.7.1 Introduction

First thing you need to do is download **wvdial** and all it's dependencies. It will do that automatically. From terminal as root:

```
# tazpkg get-install wvdial
```

Download this: Jozee's USB_Modeswitch[158]. Then from terminal as root install it:

```
# tazpkg install /path/to/usb_modeswitch-1.0.5.tazpkg
```

`/path/to` is where you downloaded it to. So you will need to adjust the above.

Make a script called `flipflop.sh`. You can place it anywhere but I am going to use where I have it as an example: `/home/`*`tux`*`/scripts/`.

If you use another name besides "tux" then adjust it accordingly. Now open up `flipflop.sh` with your prefered editor. Put the following in it and save the file:

```
/usr/sbin/usb_modeswitch
sleep 15
usb_modeswitch -v 0x1f28 -p 0x0020 -R 1
```

From terminal we need to **chmod** it so you can use the script:

```
# chmod a+x /home/tux/scripts/flipflop.sh
```

Again if you have it saved somewhere else then adjust accordingly. Sorry to repeat that LoL.

2.3.7.2 wvdial.conf

We have now installed all that we need to install. There are just a few more steps involved. We need to edit `wvdial.conf`. As root from terminal enter this:

```
# leafpad /etc/wvdial.conf
```

In `wvdial.conf` we need to make it look like this:

```
[Dialer Defaults]
Modem = /dev/ttyACM0
Baud = 460800
Stupid Mode = 1
Auto DNS = 1
Init1 = ATZ
Init2 = ATQ0 V1 E1 S0=0 &C1 &D2 +FCLASS=0
ISDN = 0
Modem Type = USB Modem
Phone = #777
```

(continues on next page)

[158] http://people.slitaz.org/~jozee/packages/3G_modem/usb_modeswitch-1.0.5.tazpkg

(continued from previous page)

```
FlowControl = Hardware (CRTSCTS)
Dial Command = ATDT
Username = Your Cricket Number Goes Here.
Password = Cricket
```

Where it says "Your Cricket Number Goes Here" replace that with your cricket number. Your cricket number should be on your monthly bill if you don't know what it is. Once it is done save the file.

Now again as root in terminal issue this command:

```
# /home/tux/scripts/flipflop.sh
```

You should get something like this below:

```
Looking for target devices ...
 No devices in target mode or class found
Looking for default devices ...
 Found default devices (1)
 Found a default device NOT in target class mode
Accessing device 003 on bus 002 ...
Using endpoints 0x08 (out) and 0x87 (in)
Inquiring device details; driver will be detached ...
Looking for active driver ...
 OK, driver found ("dummy")
 OK, driver "dummy" detached

Received inquiry data (detailed identification)
-------------------------
  Vendor String: Cricket
   Model String: T-Flash Disk
Revision String: 2.31
-------------------------

Device description data (identification)
-------------------------
Manufacturer: Cal-comp E&CC Limited
     Product: USB Micro SD Storage
  Serial No.: XXXXXXXXXXXXXXX
-------------------------
Setting up communication with interface 0 ...
Trying to send the message to endpoint 0x08 ...
 OK, message successfully sent
 Device is gone, skipping any further commands
-> Run lsusb to note any changes. Bye.


Looking for target devices ...
 No devices in target mode or class found
Looking for default devices ...
 No default device found. Is it connected? Bye.

tux@slitaz:~$
```

It looks like it totally failed but it didn't. It just doesn't know what the driver is. Again from root execute this command:

```
# modprobe cdc_acm
```

That should be the last time you have to enter that command. We will now add it to to the LOAD_MODULES line in /etc/rcS.conf to load at boot.

2.3.7.3 rcS.conf

From terminal as root:

```
# leafpad /etc/rcS.conf
```

Go down to the following line:

```
LOAD_MODULES="snd_intel8x0 yenta_socket rtc agpgart intel-agp"
```

Add this to the line: “cdc_acm”. It should now look like this:

```
LOAD_MODULES="snd_intel8x0 yenta_socket rtc cdc_acm agpgart intel-agp"
```

Save it and exit.

We are almost done. From terminal as root again give this command:

```
# wvdial
```

It should now dial and connect. Do not close the terminal or it will disconnect. Everytime you poweroff or unplug the modem you have to do the flipflop.sh like I showed you above. If you just reboot your computer you don’t have to do the flipflop.sh. Good Luck!

2.3.8 Bluetooth

author ernia, linea, jozee

2.3.8.1 Introduction

This guide will help you to configure bluetooth, e.g. to use your phone as a modem. Do following steps as root. First of all you must install Slitaz packages related to bluetooth:

```
# tazpkg get-install bluez
# tazpkg get-install linux-bluetooth
```

The first package contains the Official Linux Bluetooth protocol stack www.bluez.org and the second contains the kernel modules related to bluetooth. Now you can launch the bluetooth daemon:

```
# bluetoothd
```

And load the module required by your local bluetooth device, this is very likely btusb:

```
# modprobe btusb
```

If btusb is the module you need the command **hcitool dev** to show you the local device address:

```
root@slitaz:/home/tux# hcitool dev
Devices:
        hci0    00:02:72:xx:xx:xx
```

If the command **hcitool dev** does not show you any `hci*` device you must detect which module you need and repeat the previous step until you have a `hci*` device. Now that you have `hci*` you can run the **hcitool scan** command to show you if there is any discoverable device nearby. So if you are searching, e.g. your phone, remember to put it in discoverable mode:

```
root@slitaz:/home/tux# hcitool scan
Scanning ...
        00:18:C5:xx:xx:xx       Nokia 6151
root@slitaz:/home/tux#
```

Now you can pair your locale device with your remote device. You use **bluez-simple-agent** to do that.

If you want to start the pairing from the remote device you need to put your local device in discoverable mode:

```
# hciconfig hci0 piscan
```

Now launch **bluez-simple-agent**, start the pairing from the remote device and wait for **bluez-simple-agent** to ask you for the pin code needed to pair. This code will be the same one you typed on the phone. When you are done you need to kill **bluez-simple-agent** with `Ctrl+C`:

```
root@slitaz:/home/tux# hciconfig hci0 piscan
root@slitaz:/home/tux# bluez-simple-agent
Agent registered
RequestPinCode (/org/bluez/2944/hci0/dev_00_18_C5_xx_xx_xx)
Enter PIN Code: 1234
^CTraceback (most recent call last):
  File "/usr/bin/bluez-simple-agent", line 113, in <module>
    mainloop.run()
KeyboardInterrupt
root@slitaz:/home/tux#
```

Put your local device back in undiscoverable mode:

```
# hciconfig hci0 pscan
```

If you want to start the pairing from your local device you can do it using **bluez-simple-agent** with the `hci*` of your local device and the address of the device you discovered with **hcitool scan**. In this case you will choose the pin in **bluez-simple-agent** and type the same pin on the remote device:

```
root@slitaz:/home/tux# bluez-simple-agent hci0 00:18:C5:xx:xx:xx
RequestPinCode (/org/bluez/3078/hci0/dev_00_18_C5_xx_xx_xx)
Enter PIN Code: 1234
Release
New device (/org/bluez/3078/hci0/dev_00_18_C5_xx_xx_xx)
root@slitaz:/home/tux#
```

Be fast because **bluez-simple-agent** has a timeout which I don't know how to set (suggestions?)

2.3.8.2 Use a phone as a modem

Now you have your two devices paired, i will go on with the phone as a modem example, I don't have other experiences with bluetooth devices. To use your phone as a modem you need the rfcomm module. At the moment I am writing this guide the rfcomm module of Slitaz is compiled without tty support. You can check if this is the case with the following commands:

```
root@slitaz:/home/tux# zcat /proc/config.gz | grep RFCOMM
CONFIG_BT_RFCOMM=m
# CONFIG_BT_RFCOMM_TTY is not set
root@slitaz:/home/tux#
```

If `CONFIG_BT_RFCOMM_TTY` is not `=y` you must recompile the module with tty support. Install **`linux-source`**:

```
# tazpkg get-install linux-source
```

`cd` in `/usr/src/linux-`uname -r`/` and give this command:

```
# make CONFIG_BT_RFCOMM=m CONFIG_BT_RFCOMM_TTY=y M=net/bluetooth/rfcomm
```

This will build only the `rfcomm` module with tty support. You will find it in `/usr/src/linux-`uname -r`/net/bluetooth/rfcomm/rfcomm.ko`. Now you can delete the old `rfcomm` module and replace it with your module:

```
# rm /lib/modules/`uname -r`/kernel/net/bluetooth/rfcomm/rfcomm.ko.gz
# cp /usr/src/linux-`uname -r`/net/bluetooth/rfcomm/rfcomm.ko /lib/modules/
↪`uname -r`/kernel/net/bluetooth/rfcomm/
# depmod -a
```

Now that we have a working rfcomm module we must detect the channel where the phone dialup service is listening:

```
# sdptool browse 00:18:C5:xx:xx:xx
```

Where 00:18:C5:xx:xx:xx is the address of your phone, it will give a list with the services available on your phone. The one you are looking for is Dial-up networking or DUN:

```
Service Name: Dial-up networking
Service RecHandle: 0x10000
Service Class ID List:
  "Dialup Networking" (0x1103)
  "Generic Networking" (0x1201)
Protocol Descriptor List:
  "L2CAP" (0x0100)
  "RFCOMM" (0x0003)
    Channel: 1
Language Base Attr List:
  code_ISO639: 0x656e
  encoding:    0x6a
  base_offset: 0x100
Profile Descriptor List:
  "Dialup Networking" (0x1103)
    Version: 0x0100
```

2.3.8.3 `rfcomm.conf`

As you can see it is listening on channel 1, so you edit `/etc/bluetooth/rfcomm.conf` this way:

```
#
# RFCOMM configuration file.
#
rfcomm0 {
#        # Automatically bind the device at startup
         bind yes;
#
#        # Bluetooth address of the device
         device 00:18:C5:xx:xx:xx;
#
#        # RFCOMM channel for the connection
         channel 1;
#
#        # Description of the connection
#        comment "Example Bluetooth device";
}
```

Now with:

```
# modprobe rfcomm
# rfcomm bind all
```

You can use `/dev/rfcomm0` as a modem in a **`wvdial`** script. You may want to check this guide *3G-UMTS* (page 161)

2.3.9 Samba

author emgi, linea

2.3.9.1 What is Samba?

The Samba or SMB suite is a collection of software to enable Windows-style networking on Linux computers. The name Samba is a variation on SMB, short for Server Message Block, which is the file sharing protocol used by Windows machines. The first version of Samba dates from 1992 and a lot of development has been invested over the years. This has resulted in very stable and reliable code. The Samba project also has its own website, www.samba.org where you can do more reading on all aspects of the program.

2.3.9.2 What can I do with it?

With Samba, your SliTaz machine is able to access file shares which are on Windows servers. Samba is a collection of programs which enable file sharing between Windows and Linux. It is also possible to host a Windows share on your Linux machine, allowing Windows clients to access the files as if they were on a Windows server.

2.3.9.3 How does it work?

The Samba Suite contains two components. A server implementation (next section) and a client implementation. For example if you have a storage device at home, it will most likely support SMB/CIFS. This means you can use the Samba client to access it from Linux. This is more convenient than FTP or HTTP, especially when you want to use the share for more than just an incidental file copy. With Samba you can access the SMB share as if it were a local disk meaning you can read and write to it, both from the shell, from Perl or with the Graphic File Manager from the SliTaz distribution.

As usual, we use **tazpkg** to install the Samba client:

```
# tazpkg -gi smbclient
# tazpkg -gi smbfs
```

The **tazpkg** manager will take care of the dependencies so probably you will see a lot more packages being installed. Once this is completed, you can view the shares on your Windows server or storage system with:

```
# smbclient -L nas
Enter User's password: ****
```

This will provide a list of the shares available on `\nas`. The result should look similar to the example below:

```
Domain=[WORKGROUP] OS=[Unix] Server=[Samba 3.6.3-31a.osstech]

     Sharename       Type      Comment
     ---------       ----      -------
     IPC$            IPC       IPC Service ("My NAS")
     webaxs          Disk
     share           Disk      LinkStation folder
     info            Disk      LinkStation Utilities
Domain=[WORKGROUP] OS=[Unix] Server=[Samba 3.6.3-31a.osstech]

     Server               Comment
     ---------            -------
     NAS                  My NAS

     Workgroup            Master
     ---------            -------
     WORKGROUP            NAS
#
```

From the above example, we are interested in access to Sharename 'share'. Although not strictly required, it is recommended to add the server name to `/etc/hosts` for name resolution when it has a fixed IP address. Because we are running on Linux we need to 'escape' the backslashes in the UNC path, effectively needing twice the number of them. Accessing `\share` on `\\nas` goes like this:

```
# smbclient \\\\nas\\share
Enter User's password: ****
smb: \>
```

At the prompt you can issue FTP-style commands to browse the file system and read/write files. Typing `help` gives you a summary of the options:

```
smb: \> help
?              allinfo        altname        archive        blocksize
cancel         case_sensitive cd             chmod          chown
close          del            dir            du             echo
exit           get            getfacl        hardlink       help
history        iosize         lcd            link           lock
lowercase      ls             l              mask           md
mget           mkdir          more           mput           newer
open           posix          posix_encrypt  posix_open     posix_mkdir
posix_rmdir    posix_unlink   print          prompt         put
pwd            q              queue          quit           readlink
rd             recurse        reget          rename         reput
rm             rmdir          showacls       setmode        stat
symlink        tar            tarmode        translate      unlock
volume         vuid           wdel           logon          listconnect
showconnect    ..             !
smb: \> quit
#
```

Fortunately you can also access a Windows share by mounting it on your box like this:

First create the mount point on your SliTaz box:

```
# mkdir /nas
#
```

Then issue the **mount** command:

```
# mount.cifs \\\\nas\\share /nas -o user=admin
Password: ******
#
```

An alternative method is like this:

```
# mount //nas/share /nas
Password: ******
#
```

If there is no access-denied-message in reply to the password, it means the share is mounted and can be accessed from your box. That's basically it!

Tip: The password must be the admin password for your storage device. This may not work when you have enabled user-based security. In some cases the mount may be read-only.

Many of the problems with Samba are caused by a lack of understanding the Windows security mechanism. In its basic version, which is still used most of the time, this is user-based security so you will need a **valid windows user account** to access the file share. When in trouble, verify the access is working from a windows machine and use the same credentials under Linux.

2.3.9.4 Running a Samba Server on SliTaz

If you need to set up a Windows file server but you don't want to invest in over-the-top hardware and Windows licenses, Samba is the way to go. Of course you can use SliTaz as the Linux platform to run the server. With this solution you will also benefit from the superior stability of the Linux OS. One of the problems of a Samba administrator is that he sometimes forgets the details of his installation because everything simply keeps on running once it has been set up properly. This is one of the reasons why Linux people need documentation anyway. ;-)

To install the server component of Samba, install the following package:

```
# tazpkg -gi samba
```

In order to make the user experience even more seamless, it is recommended to keep the accounts synchronized. This means the users should have the same username & password on their windows box as they are using on the Samba server. Explanation: When Windows connects to a share, it always attempts to authenticate using the credentials of the user that requests the access. If there is no match, the user is prompted for a username/password but when they are the same, the share is accessed without any further prompting.

The procedure to add a user to your samba server:

First add Linux user using the **adduser** command.

```
# adduser smbuser01
Changing password for smbuser01
New password:
Retype password:
Password for smbuser01 changed by root
```

Then add the same user name to Samba using **smbpasswd -a**

```
# smbpasswd -a smbuser01
New SMB password:
Retype new SMB password:
Added user smbuser01.
#
```

Use **smbpasswd -h** to view the options.

The file shares and printers you want to allow your Windows clients access to can be defined in the Samba configuration file which is `/etc/samba/smb.conf`

There are countless options to configure here like login scripts, home folders etcetera. Elaborating on all of them would be way beyond the scope of this guide.

A good place to start is http://www.tldp.org/HOWTO/SMB-HOWTO-2.html which provides a comprehensive overview of Samba but there are many other sites on the web with virtually the same content. As Samba has been around for over two decades, almost all things have already been tried, done and extensively documented in the process. Just search the web. You may encounter some negative comments as well but these are generally posted by people who made some basic mistake or have been the victim of some undocumented change in Windows. Don't let yourself be discouraged by this. Samba is Alive and Kicking and when you know how to search for them, you can be sure to find detailed answers to all issues you may encounter while setting up your own server.

2.3.10 Remote Desktop

author seawolf

2.3.10.1 NoMachine (NX)

Important: This section is under construction and has not been verified by others. It has been created from one particular (working) configuration.

NoMachine[159] (NX) is a propietary remote desktop system, supporting multiple hosts and keypair-based log-ins.

Note: This guide assumes you have a working SSH configuration for remote access. NoMachine/NX channels authentication and encrypted communications through SSH. This can be via password authentication or an RSA key pair; NoMachine/NX uses a DSA key pair to authenticate.

Note: My (seawolf[160]) personal SSH configuration uses a non-standard port and key pair-based, password-less authentication. The differences in this guide to a standard, password-protected SSH configuration using port 22 are irrelevant as the default is given initially.

There are three parts to a NX system:

1. the **server** is the machine to which you connect;
2. the **node** is one in a group of resources that hosts your session — this can be the server;
3. the **client** that connects to a server (and in turn, the node).

2.3.10.2 Download the Software

There are three packages available to download[161] from NoMachine. All clients must have installed the client package, whereas all three packages must be installed on servers and nodes. This is because parts of the client package is used by the node, parts of which are used by the server.

- Decompress the three NX packages (client, node, server) on the server into `/usr` to create `/usr/NX` directory.
- Decompress the client package on the client(s).

2.3.10.3 Server Configuration

The automated commands are not compatible with BusyBox and the SliTaz configuration, so the installation commands need to be edited:

- Create the symlinks `/etc/rc.d/rc#.d/` all pointing to `/etc/init.d/`. This can be easily achieved with the following command performed as the *root* user:

```
# for NUM in 0 1 2 3 4 5 6 ; do cp /etc/rc.d/init.d /etc/rc.d/rc$NUM.
↪d; done
```

[159] http://www.nomachine.com
[160] http://forum.slitaz.org/index.php/profile/20/seawolf
[161] http://www.nomachine.com/download-package.php?Prod_Id=2071

- Modify `/usr/NX/scripts/setup/nxserver` by:
  - fixing the user add/del commands in lines 924, 963, 984.
  - comment out the command in line 956 by placing a hash immediately after the opening quotation mark (command not needed)
- Install the server:

```
$ sudo /usr/NX/scripts/setup/nxserver --install fedora
```

  - the OS doesn't really matter but it's the closest match for the numerous `/etc/init.d` commands
- Change the server name & SSH port (if necessary) in `/usr/NX/etc/server.cfg` (line 31, 36 & 236)
- Allow administrative logins in line 87 of `/usr/NX/etc/server.cfg`

2.3.10.4 Node Configuration

The automated commands are not compatible with BusyBox and the SliTaz configuration, so the installation commands need to be edited:

- Modify `/usr/NX/scripts/setup/nxnode` by:
  - specify SSH port in line 43 (if necessary)
  - change to `local.sh` on line 1305
- Install the node:

```
$ sudo /usr/NX/scripts/setup/nxnode --install fedora
```

  - ignoring warning about CUPS detection if you don't have a printer
- Change server name & SSH port (if necessary) in `/usr/NX/etc/node.cfg` (line 32, 342)

2.3.10.5 User Configuration

- Enable the nx user account by using:

```
$ sudo passwd -u nx
```

- Confirm this with:

```
$ sudo /usr/NX/bin/nxserver --usercheck <username>

NX> 900 Verifying public key authentication for NX user: <username>.
NX> 900 Adding public key for user: <username> to the authorized keys␣
↪file.
NX> 716 Public key added to: /home/ <username> /.ssh/authorized_keys2.
NX> 900 Verifying public key authentication for NX user: <username>.
NX> 900 Public key authentication succeeded.
NX> 999 Bye.
```

You should now be able to log-in to the server/node using a normal username and password.

2.3.10.6 Recreating Keys

When the default key pair authenticates correctly, they should be regenerated for security reasons. Issue the following command to create a new key pair:

```
$ sudo /usr/NX/bin/nxserver --keygen
```

New keys should be created. Distribute the secret key `/usr/NX/share/keys/default.id_dsa.key` to clients and import it in the client GUI (*Configure* → *General* → *Server* → *Key...* → *Import*).

Restart the server (**`sudo /usr/NX/bin/nxserver --restart`**) to complete the changes.

2.3.10.7 Tips

- If you want to use key pair and password-less authentication, ammend `/etc/ssh/sshd_config` with:

  ```
  PasswordAuthentication no
  AllowUsers nx //other usernames//
  ```

- Open ports 5000-5200 for an unencrypted connection. This is because after a successful authentication has taken place the client reconnects to a display in the range starting at 'DisplayBase' up to the value ('DisplayBase' + 'DisplayLimit'). These parameters default to the values "1,000" and "200" respectively and TCP port numbers are obtained by adding the value "4,000" to the display numbers, thus giving 5000 and 5200. If encrypted display is enabled, all traffic is piped through SSH.
- If the client fails to connect to the server with the following messages:

  ```
  NX> 203 NXSSH running with pid: <PID>
  NX> 285 Enabling check on switch command
  NX> 285 Enabling skip of SSH config files
  NX> 285 Setting the preferred NX options
  NX> 200 Connected to address: <IP address> on port: <SSH port>
  NX> 202 Authenticating user: nx
  NX> 208 Using auth method: publickey
  NX> 204 Authentication failed.
  ```

then the `/usr/nx/home/nx/.ssh/authorized_keys2` file is likely at fault.

2.4 Software

2.4.1 How-To install Adobe Flash Player

author pankso, seawolf, linea, jozee, kultex

Open a terminal. Become super-user:

```
$ su -
```

The default password for the root account is `root`.

Type the following as one command:

```
# tazpkg get-install get-flash-plugin && get-flash-plugin
```

The package installs a **get-flash-plugin** command which creates a pseudo-package that downloads the source code from Adobe. It packages the build source code and pulls in any necessary dependencies.

By using this method the Flash plugin can be managed through **tazpkg**; it will update itself as the repo package is updated.

When you install Flash to SliTaz 3.0 and **Midori** is not recognizing the plugin — just reinstall **Midori** with

```
# tazpkg get-install midori --forced
```

Thats it.

Note: This was successfully tested in **Midori**, **Firefox/Shiretoko** & **Opera**, on Cooking 2010-03-14 (the v3 RC).

2.4.2 Boot time

author jozee, linea

SliTaz boots fast! But most likely you haven't turned super fast boot on yet. To do this, just edit your `/etc/rcS.conf`:

```
FAST_BOOT_X="yes"
```

Remove `dbus`, `hald` and `slim` from the `RUN_DAEMONS` array. That's all. Note, on some machines, `FAST_BOOT_X` may not boot properly. However, if Xvesa/Xorg works after you enable this feature, it's safe to leave it on.

Now, how do you measure the boot time? SliTaz logs time that rcS scripts take to run in the `/var/log/boot-time` file. In my case, it shows 5 secs.

But, how do you graph your full boot process (including the kernel-boot time)? Its quite easy on SliTaz. Just do and install the following:

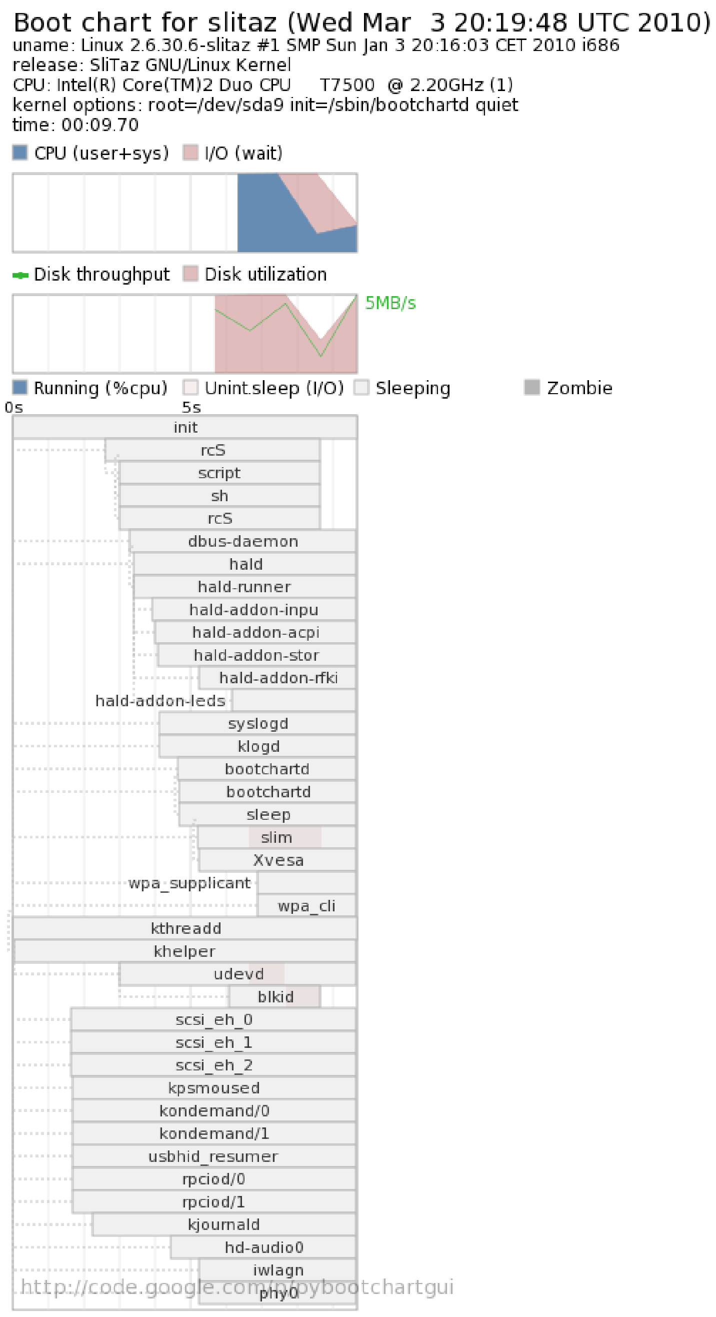
```
# tazpkg get-install pybootchartgui
```

Pybootchartgui will then create a chart automatically in `/var/log`.

Now, just edit your `menu.lst` file for grub and add `init=/sbin/bootchartd`. In my case, it's like this:

```
title slitaz with bootchart
root (hd0,8)
kernel /boot/vmlinuz-2.6.30.6-slitaz root=/dev/sda9 init=/sbin/bootchartd␣
↪quiet
```

Here's my boot chart with WPA2 Wi-Fi (under 10 secs boot time on a laptop with hard disk). This is out-of-the-box!!!



```
Boot chart for slitaz (Wed Mar  3 20:19:48 UTC 2010)
uname: Linux 2.6.30.6-slitaz #1 SMP Sun Jan 3 20:16:03 CET 2010 i686
release: SliTaz GNU/Linux Kernel
CPU: Intel(R) Core(TM)2 Duo CPU     T7500  @ 2.20GHz (1)
kernel options: root=/dev/sda9 init=/sbin/bootchartd quiet
time: 00:09.70
```

Fig. 2: My bootchart

2.4.3 Midori Tips & Tweaks

author rupp, jozee, linea, godane

I am writing this guide since **`Midori`** is the default browser in SliTaz 3. Your first thought may be to install **`Firefox`** and not giving **`Midori`** a try because it lacks features. I thought it lacked features or Addon's also until I dug around some. If you want to know the history of **`Midori`** visit it's Wiki page: Midori Wiki[162].

Things that I will cover:

[162] http://en.wikipedia.org/wiki/Midori_(web_browser)

- Ad Blocking
- Custom Search Engines
- Flash Blocking
- UserScripts
- Shortcuts

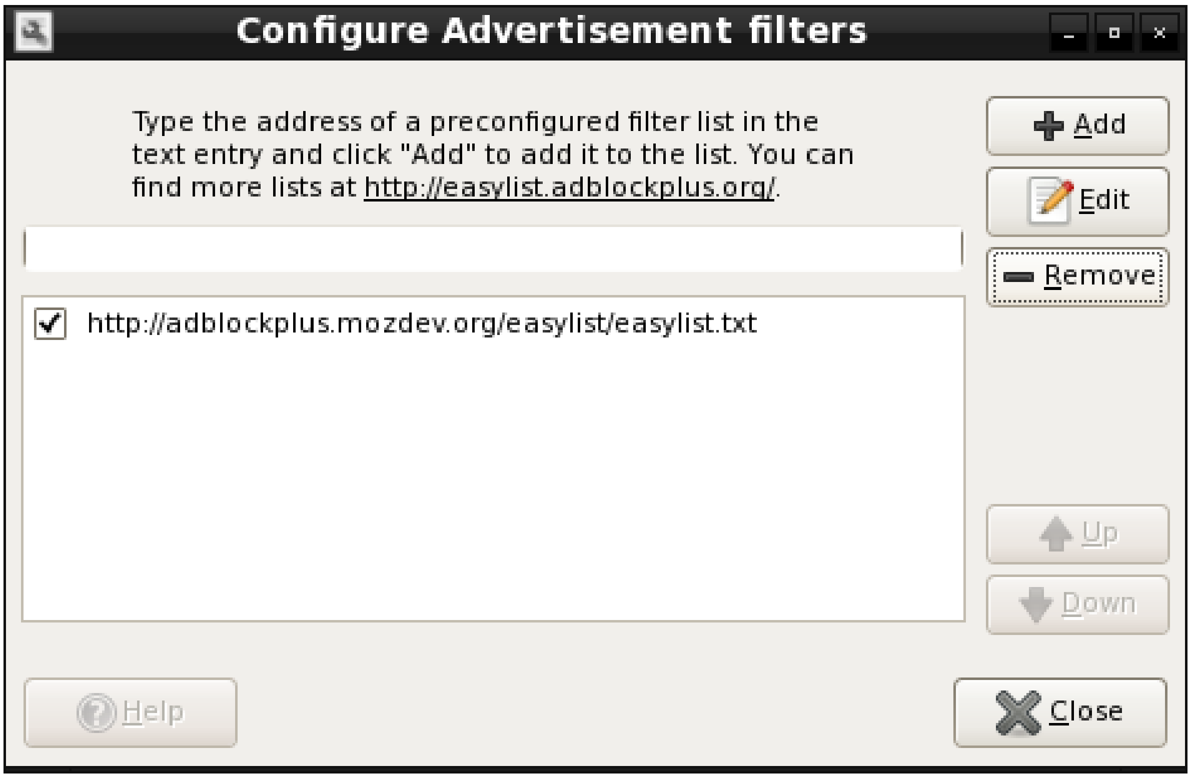
2.4.3.1 Enabling Extensions

Upon opening **Midori** click on the *Tools Menu*, Go down to *Extensions* and click on it. This should open up a “panel” that displays boxes that you can check to enable different Extensions or Options. Check only the boxes you wish to enable. I would enable *Statusbar Features* and *Toolbar Editor* right now. Everything involving those two should be self explantory. So no need for a “HowTo”. I would tell you what other things to enable right now, but if someone adds to this wiki then this paragraph would need to be edited. So if anything needs to be enabled it will be mention it as it go along.

2.4.3.2 Ad Blocking

In extensions enable: Advertisement blocker

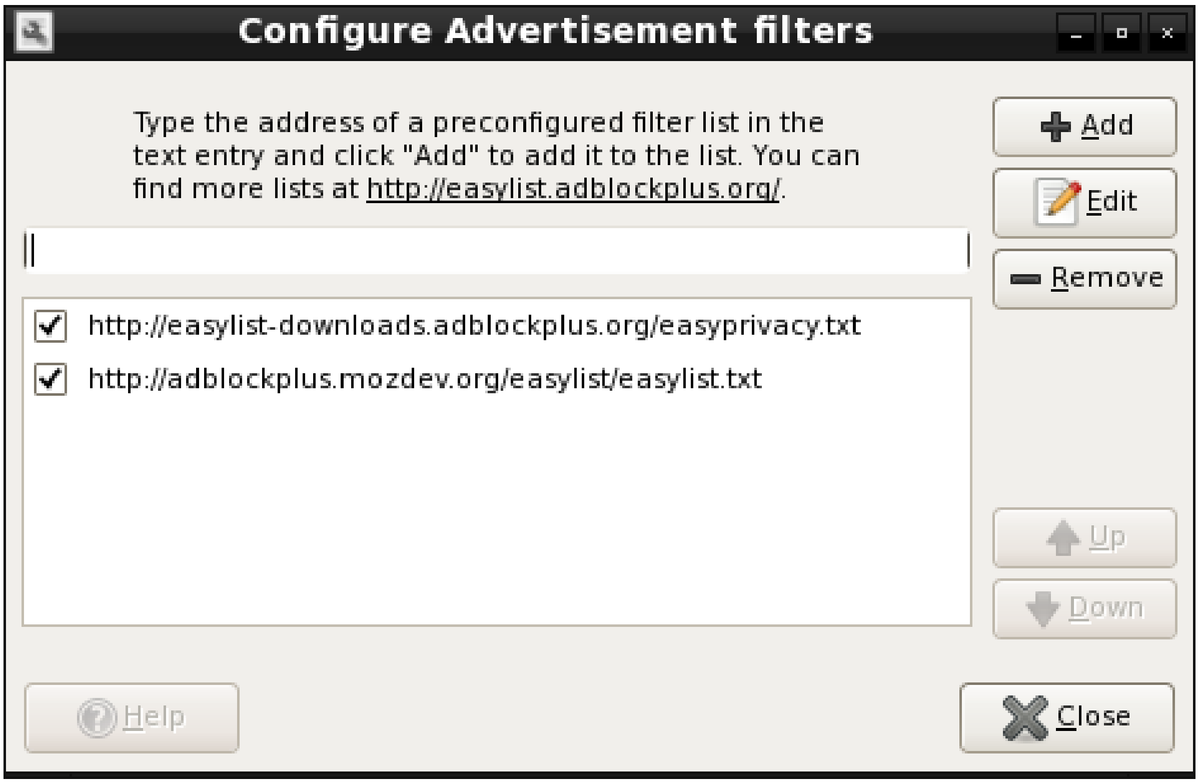
Most people like ad blocking that **FireFox** has through an extension called Adblock Plus (ABP). Well **Midori** has an adblocker that uses the well known “easylist” from ABP. Now that you have it enabled we need to edit it. Click on the *Tools Menu*, now click on *Configure Advertisement filters*. This should now open a box looking like this:


Configure Advertisement filters

Type the address of a preconfigured filter list in the text entry and click "Add" to add it to the list. You can find more lists at http://easylist.adblockplus.org/.

☑ http://adblockplus.mozdev.org/easylist/easylist.txt

From the picture you can see that adding a ruleset is pretty self explanatory. As an example click on *Add*, then insert this URL: http://easylist-downloads.adblockplus.org/easyprivacy.txt



Type the address of a preconfigured filter list in the text entry and click "Add" to add it to the list. You can find more lists at http://easylist.adblockplus.org/.

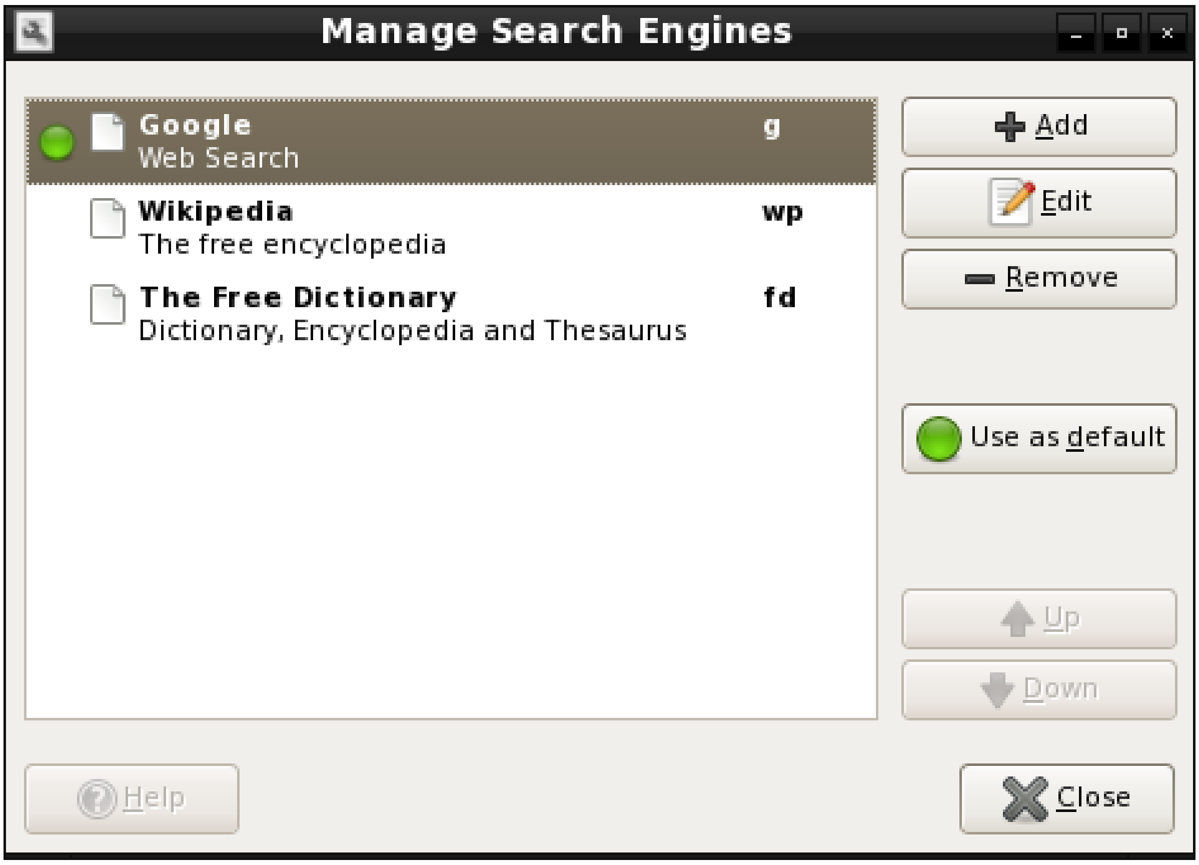
- [✓] http://easylist-downloads.adblockplus.org/easyprivacy.txt
- [✓] http://adblockplus.mozdev.org/easylist/easylist.txt

That is what it should now look like. Now to block a single image just right click the image on the page you want to block. Go down to *Block Image* and click *Add*. Now that image is blocked.

2.4.3.3 Managing Search Engines

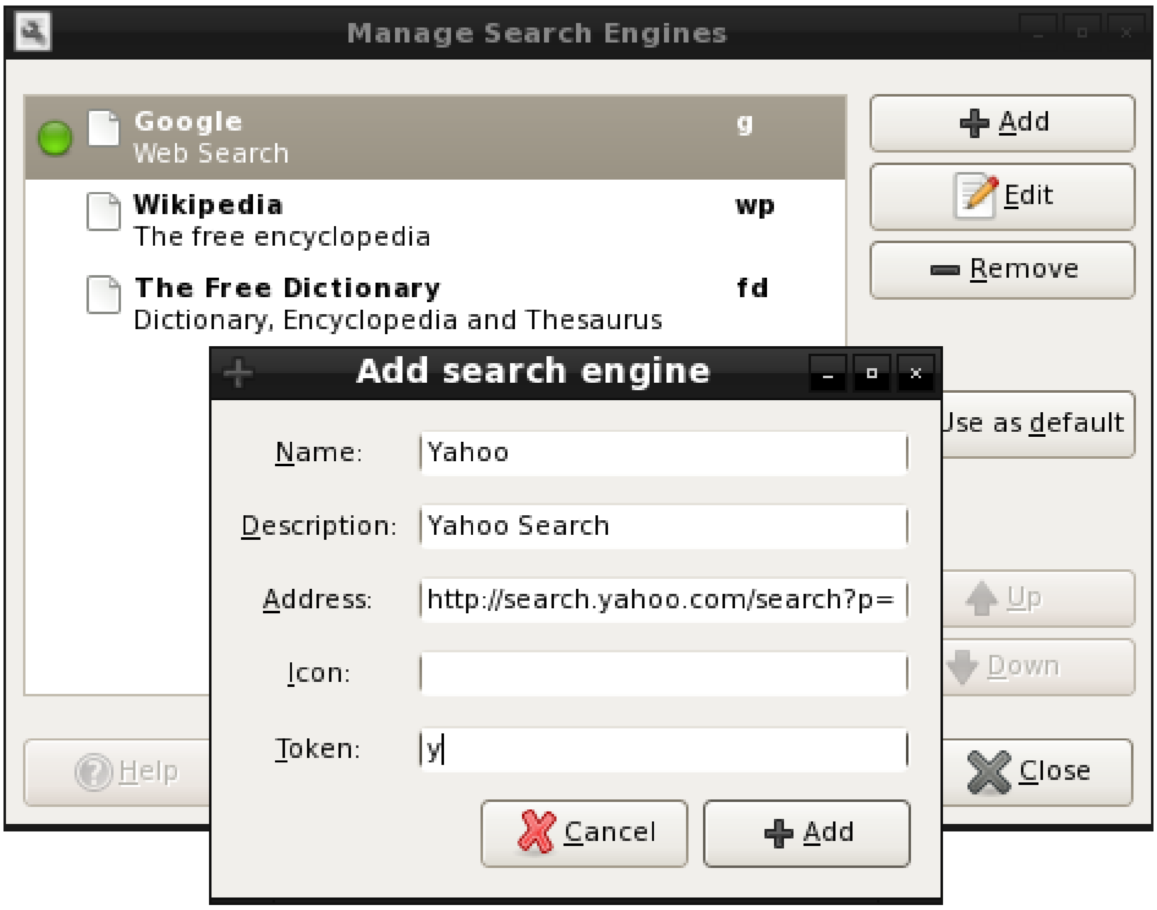
No extension needs to be enabled for this feature.

This is one of my favorite things in any browser. I use this for other things then just search engines. I will share that later. Click on the “Tools Menu”, then click on “Manage Search Engines”. Just like Ad blocking you should get a box to pop up that looks like this:



Google Web Search	g
Wikipedia The free encyclopedia	wp
The Free Dictionary Dictionary, Encyclopedia and Thesaurus	fd

Since I don't really use anything else but google I am just going to add Yahoo! as an example of how to add it and use it. Click add and fill it in to look like this:


Manage Search Engines

Google Web Search	g
Wikipedia The free encyclopedia	wp
The Free Dictionary Dictionary, Encyclopedia and Thesaurus	fd

Add search engine

Name: Yahoo

Description: Yahoo Search

Address: http://search.yahoo.com/search?p=

Icon:

Token: y

Once that is done go up to the URL bar in midori and type: “y slitaz linux” without the quotes.

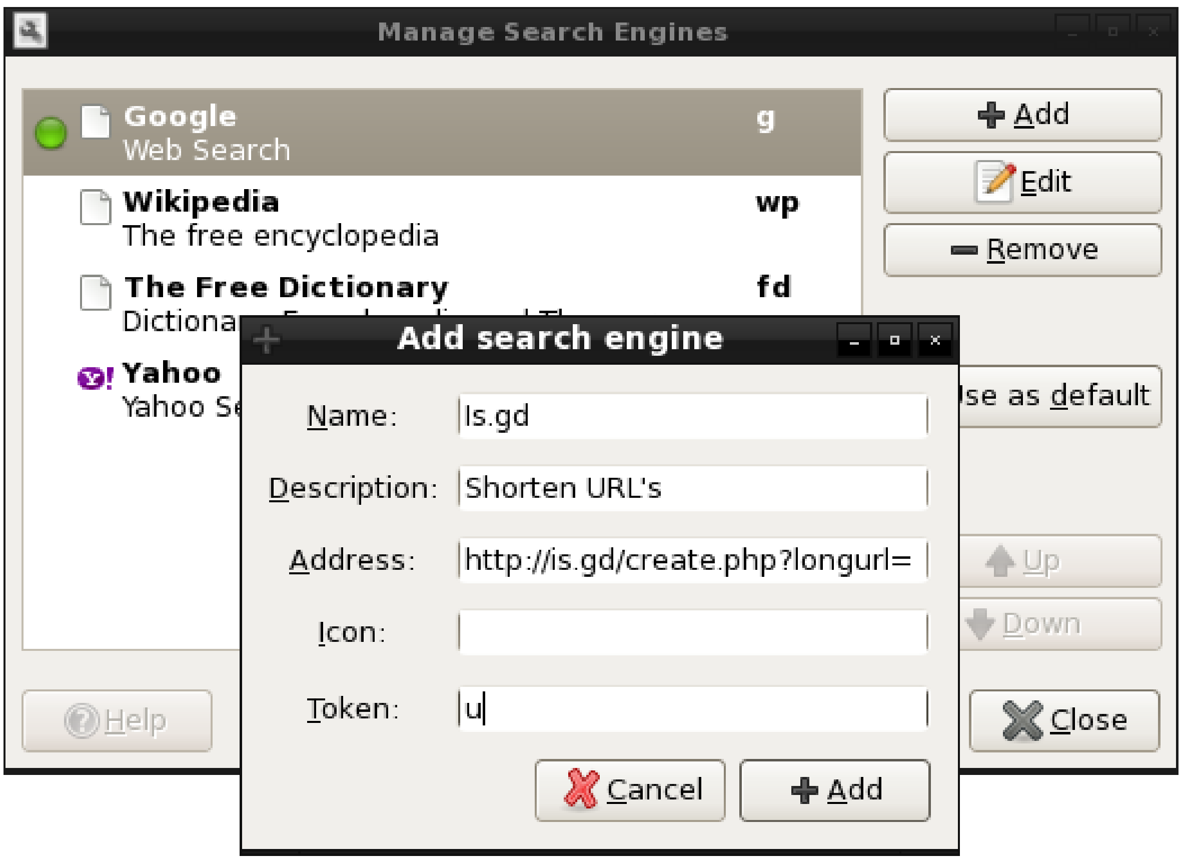
After pressing `Enter` you should now see the Yahoo search results for *slitaz*.

Typing “g slitaz” again without the quotes would give you the google results for SliTaz.

Here is what I use it for besides search engines. In Opera when you bookmark something you can make a nickname for the site. I have the letters: `fs` for http://forum.slitaz.org. So I just type “fs” without the quotes in the URLbar to go to SliTaz’s forum. Now I don’t see this feature in Midori so I just make a custom search engine like I did for Yahoo!. I am not really adding a search just using it as a shortcut. I think for the pictures I added that it is self explanatory how to add any site you want to make a shortcut for.

For all you Twitter users or people that like shortened URLs. Add this to custom searches:

```
http://is.gd/create.php?longurl=
```



Name: Is.gd

Description: Shorten URL's

Address: http://is.gd/create.php?longurl=

Icon:

Token: u

I give mine the Token letter "u". When I have a big URL I want to shorten, I go to the URL bar, go to the left of the "http" in the URL, then I type "u". Then press `Space` then `Enter` it takes me to that site and shortens the URL. Example:

Result:

For those who hate using the mouse like me and want to shorten URL's you can do the same thing above like this:

`Ctrl+L`, ←, type u, `Space`, then `Enter`

If you have javascript enabled then the shortened URL should already be highlighted. So a quick `Ctrl+C` should copy it for you.

2.4.3.4 Flash Blocking

To flashblock you need to use a userscript. UserScripts are just custom javascripts to do a range of things. I will cover other UserScripts later in this wiki but flashblock is more then likely gonna be the only one anyone will use.

First we need to make a folder if you don't already have it: `/home/`*`tux`*`/.local/share/midori/scripts`. "tux" is just used as an example. If you have a different user name then change accordingly.

These pictures are kind of big so I am just linking to them. Example before FlashBlock:

Now go to: Flashblock Wannabe[163]. Download `FlashBlock.user.js.txt` (remove the `.txt` extension) to the folder we just created. Go to the *Tools Menu*, and click on *UserScripts*. You should

[163] http://rightfootin.blogspot.com/2009/04/flashblock-wannabe.html

now see "FlashBlock Wannabe". Make sure there is a checkmark next to it. Sometimes the UserScripts show up right away and other times they don't so I just restart **`Midori`** to view the UserScripts. So if it isn't showing then just restart *Midori*. Now you should have flash blocked unless you click on it like the flashblocker addon for **`FireFox`**. It should look like this:

2.4.3.5 User Scripts

If you already did the Flashblocking part then this should be as easy as downloading them and putting them in the correct folder. No big HowTo here I will just show you some examples of the ones I use and what they do.

Linkify[164]: If you have ever been to a site or forum and someone posted a link that is just text and not clickable then this is for you. It makes all "http" links clickable. It saves you from highlighting and copying and pasting links to a URL bar.

Google Image Redirector[165]: If you ever google search for images and click on an image you will notice that it take you to the page but the tiny image is in a frame. This userscript takes you directly to the actual image. Saves you a couple clicks of the mouse.

Google Show Options[166]: Last year google added advanced search options in results. You could click on the "Show Options" and it would list them. This bypasses clicking on it and it shows it right away without have to click it.

Just Show Images[167]: If you have ever clicked on a link from a forum or anything to certain image hosting sites you would notice all the ads. This UserScript just shows the picture you want and little else.

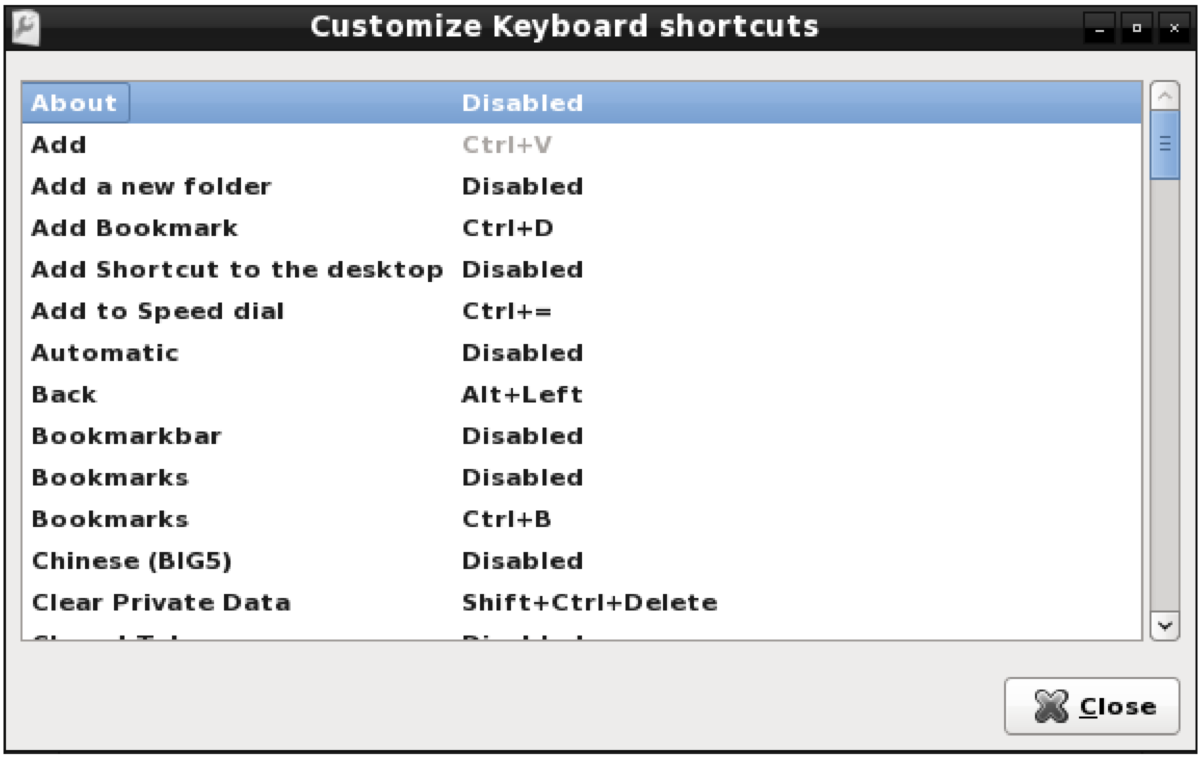
Easy YouTube Downloader[168]: If you would like to download the video from YouTube without going to a seperate site, you can use this script. It adds a download option for multiple video formats on the screen.

2.4.3.6 Shortcuts

In extensions enable: shortcuts

I try to use the mouse as little as possible. So shortcuts come in handy. Here is how to view and/or edit them to your taste. Open up the *Tools Menu* go down to and click on *Customize Shortcuts*. You will get a popup that looks like this:

[164] http://userscripts.org/scripts/show/1352
[165] http://userscripts.org/scripts/show/5059
[166] http://userscripts.org/scripts/show/72270
[167] http://userscripts.org/scripts/show/54108
[168] http://userscripts.org/scripts/show/54790



About	Disabled
Add	Ctrl+V
Add a new folder	Disabled
Add Bookmark	Ctrl+D
Add Shortcut to the desktop	Disabled
Add to Speed dial	Ctrl+=
Automatic	Disabled
Back	Alt+Left
Bookmarkbar	Disabled
Bookmarks	Disabled
Bookmarks	Ctrl+B
Chinese (BIG5)	Disabled
Clear Private Data	Shift+Ctrl+Delete

In the picture you will see an action along with an assigned "shortcut". Some show an action but are disabled. I will show you how to change a "shortcut" or to enabled one that is disabled. To assign a "shortcut" click on what you would like to change to highlight. Now go to the right and click the mouse where the "shortcut" or word "disabled" is and click. It should change the text to *New accelerator. . .* Now all you have to do is press a combination of keys you would like to make up your shortcut. Popular choices to start your shortcut with are any of these:

`Ctrl`, `Alt`, or either `Win` keys

Winkeys will be either "Super L" or "Super R" depending on which you press. Not every keyboard has a "Winkey" or only has just one. I prefer to use the `F1` to `F12` keys. Here is an example of some of my shortcuts:

Show/Hide Sidepanel	`F2`
Show/Hide Bookmarkbar	`F3`
Show/Hide Menubar	`F4`
Refresh	`F5`
Preferences	`F7`
Focus Current Tab	`F9`
Full Screen	`F11`
Bookmarks	`Ctrl+B`
Add Bookmarks	`Ctrl+D`
Homepage	`Ctrl+H`
Highlight URLbar	`Ctrl+L`
Close Tab	`Ctrl+W`
Next Tab	`Ctrl+PgDn`
Previous Tab	`Ctrl+PgUp`

Just wanted to add that if you want to "disable" a shortcut. Pressing `Backspace` in *New accelarator. . .*

will disable it.

2.4.4 Sexy desktop apps and config

author jozee, ionreflex, linea

2.4.4.1 Introduction

This page provides information about creating a beautiful desktop, available applications via the SliTaz package manager and configuration file examples or tweaks. Wbar and Tint2 desktop

First install the used packages:

- **`wbar`**
- **`tint2`**
- **`nitrogen`**
- **`cairo-clock`**

We have to create, modify or configure 3 files in your personal home directory (click on the link to download files or check the bottom of the page):

- `~/.wbar`
- `~/.config/tint2/tint2rc`
- `~/.config/openbox/autostart.sh`

`Wbar` and **`Tint2`** configuration files don't exist if you've never started the applications before, but **`Openbox`** autostart scripts should exist since it is the default Window Manager on SliTaz. Save the attached config files and put them in the correct directory.

The *Cairo analog desktop clock* needs the "Composite" extension enabled and a *composite manager* running (both activated by default on SliTaz). If you use Xorg you must adjust your `xorg.conf` file. The *composite manager* we use in SliTaz is called **`xcompmgr`**, it is light and has some command line options for shadows, etc.

To automatically execute all applications when your X session starts you must edit the **`Openbox`** autostart script with your favorite editor or use the GUI **`$(desktopbox autostart)`** to add the following lines:

```
# Start the Freedesktop standard menu panel.
#lxpanel &

# Tint2 - Simple and clean panel.
tint2 &

# Desktop Wallpaper with Nitrogen.
nitrogen --restore &

# Desktop effects composer (xcompmgr -c -r 10 &).
xcompmgr &

# Wbar icons panel.
(sleep 4 && wbar -above-desk -bpress -pos top center -isize 24 -jumpf 0 -
↪zoomf 2.0 -balfa 0) &
```

(continues on next page)

(continued from previous page)

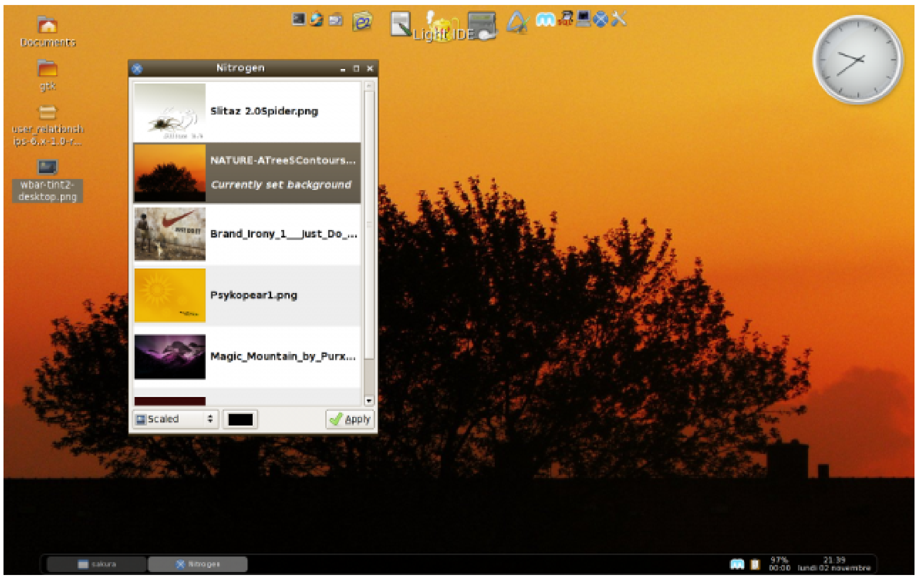
```
# Nice clock for the desktop.
(sleep 2 && cairo-clock) &
```

Make sure to comment out `lxpanel` and modify **`PCManFM`** preferences to let **`Nitrogen`** handle the desktop background. To choose your image:

```
$ nitrogen /usr/share/images
```

Now logout and login again into your customized desktop. Or kill the current process and then restart the applications:

```
$ killall lxpanel
$ tint2 &
$ wbar -above-desk -bpress -pos top center -isize 24 -jumpf 0 -zoomf 2.0 -
↪balfa 0 &
```



- Link to download .wbar file[169]
- Link to download tint2rc file[170]

2.4.5 Conky

author genesis, linea

2.4.5.1 Introduction

`Conky` is a highly customizable, lightweight system monitor that sits on the desktop. It is able to show loads of information about your computer, such as CPU usage, RAM, disk usage, CPU temperature, and also the weather or the number of new messages in your email. It allows color graphical formatting of its indicators, offering a stylish look to the desktop.

[169] http://savedonthe.net/download/185/dot.html
[170] http://savedonthe.net/download/186/tint2rc.html

2.4.5.2 Installation

Installation is made through **TazPkg**:

```
# tazpkg -gi conky
```

2.4.5.3 Basic Configuration

To set up your **Conky**, first copy the file `/etc/conky/conky.conf` to your folder `home` and rename it to `.conkyrc`. Now, you can modify the `.conkyrc` file as you like; if you want to return to the original settings, simply repeat this procedure.

To start **Conky**, open a terminal and type:

```
$ conky
```

Note that the **Conky** window appears at the top-right of the screen, with a black background.

Conky should start automatically when you turn on/restart your computer. If it doesn't, go to the *Applications menu* → *Preferences* → *Desktop Session Settings* — on the *Automatically Started Applications* tab, select *Conky* box to boot with Linux.

2.4.5.4 Appearance and meters settings

The appearance of **Conky** and the items it monitors are highly configurable. For a complete list of configuration options, access the official Conky site[171], "Documentation" link.

Tip: There are several websites that offer custom settings ready for download. Just pick one and copy it to your `~/.conkyrc`.

For example, to get your **Conky** window transparent, allowing you to see the wallpaper behind it instead of the default black background, there are two ways:

1. Change the following line of your file `.conkyrc`:

   ```
   own_window_transparent no
   ```

   to:

   ```
   own_window_transparent yes
   ```

   This creates a simple transparency effect, forcing **Conky** to copy the wallpaper as its background image.

2. For a more sophisticated effect, in which you can control the level of transparency of the **Conky** window, change your `.conkyrc` as follows:

   After this line:

   ```
   own_window_transparent no
   ```

[171] http://conky.sourceforge.net

add these two:

```
own_window_argb_visual yes
own_window_argb_value 150
```

The added two lines make **Conky** use the Linux composite (by default **Xcompmgr**) to create the transparency effect.

Note that you must activate the composite at Linux boot to get this effect working. To do this, go to the *Applications menu* → *Preferences* → *Desktop Session Settings* — on the *Automatically Started Applications* tab, select *Desktop effects with Xcompmgr* box to boot with Linux. Now right-click in an empty area of your desktop, select *Desktop effects* → *Activate composite*. You should be able to see the semi-transparent **Conky** window.

To control the level of transparency, change the end of the line:

```
own_window_argb_value 150
```

entering any value between 0 and 255, according to your preference.

2.4.5.5 References

official Conky site[172]

2.4.6 The Tor Installation Guide

author linea, jozee, mojo

This guide allows you to use the anonymous TOR network with most web browsers and command line tools such as **wget**. Begin by installing TOR:

```
# tazpkg get-install tor
# tazpkg get-install privoxy
# echo 'forward-socks5t /        127.0.0.1:9050   .' >> /etc/privoxy/config
# sed -r '/RUN_DAEMONS/s/(" *)$/ privoxy\1/' -i /etc/rcS.conf
# /etc/init.d/privoxy start
```

You can use **Vidalia** to start and configure **tor**, the proxy server is launched automatically when the SliTaz system boots. Once installed, you will find **Vidalia** in the *Menu* → *Internet*. To configure preferences via the proxy server with the following values:

Proxy address 127.0.0.1

Port 8118

and the path to the configuration file: `/home/USER/.config/torrc` (changing `USER` by User name).

```
# tazpkg get-install vidalia
# exit
$ cp /etc/tor/torrc ~/.config
```

Use Bridges to get tor working if the above options are not working.

[172] http://conky.sourceforge.net

```
# echo '
UseBridges 1
UpdateBridgesFromAuthority 1
bridge 66.160.141.98:6085 ' >> ~/.config/torrc
```

2.4.6.1 Verification and Advice

For more added security; do not launch **TOR** via the 'root' user, **Vidalia** is perfect for this (or the command line). To check your current browser with TOR: https://check.torproject.org/

2.4.7 Conspy: tiny screen or VNC

author jozee, linea, bellard, t0n1, hgt

SliTaz core provides the 10Kb **conspy** to get remote control of Linux virtual consoles. See http://conspy.sourceforge.net/

SliTaz opens 6 virtual consoles which you can access with `Ctrl+Alt+F1` to `Ctrl+Alt+F6`. You can connect to console 1 with **conspy 1** and console n with **conspy n** or the current active console with **conspy** (root user only).

To exit from **conspy** (and the virtual console) press the `Esc` key three times in quick succession.

2.4.7.1 Conspy as screen (session manager)

Linux supports up to 63 virtual consoles. You can have up to 62 (63 — X11 on console 7) sessions. Six sessions are already opened by SliTaz. You can open a new console / new session (say console 28) with **openvt -c 28 /bin/login** or **openvt -c 28 /bin/ash**. You can free this virtual console with **deallocvt 28**.

Example:

```
home$ ssh tux@slitazbox
box$ su
box# openvt -c 28 /bin/ash
box# conspy 28
# some commands
...
# <ESC><ESC><ESC>
box# exit
box$ exit
```

Tip: With a recent **busybox** (see below) you can skip the openvt step:

```
home$ ssh tux@slitazbox
box$ su
box# conspy -cs 28
# some commands
...
# <ESC><ESC><ESC>
box# exit
box$ exit
```

Later:

```
home$ ssh tux@slitazbox
box$ su
box# conspy 28
# more commands
...
# <ESC><ESC><ESC>
box# exit
box$ exit
```

To close the session:

```
home$ ssh tux@slitazbox
box$ su
box# conspy 28
# exit
<ESC><ESC><ESC>
box# deallocvt 28
box# exit
box$ exit
```

Tip: With a recent **busybox** (see below) you can skip the deallocvt step:

```
home$ ssh tux@slitazbox
box$ su
box# conspy 28
# clear; exit
<ESC><ESC><ESC>
box# exit
box$ exit
```

If you prefer to use **screen**, see http://www.gnu.org/software/screen:

```
# tazpkg get-install screen
$ screen -S MySession
```

2.4.7.2 Conspy as VNC (shared console)

You can share a virtual console between two or more users. Say RemoteUser wants to show some commands to SlitazUser using SlitazBox. RemoteUser selects SlitazUser's console with **chvt**:

```
home$ ssh SlitazBox
SlitazBox$ su
SlitazBox# chvt 1
SlitazBox# conspy 1
```

Now both users show the same terminal. A third user can do **conspy 1** too.

If you prefer to share the X11 display, install **x11vnc** (VNC server) and **x11vnc-extra** (java VNC client) see http://www.karlrunge.com/x11vnc/:

```
SlitazBox# tazpkg get-install x11vnc
SlitazBox# tazpkg get-install x11vnc-extra
SlitazBox# /etc/init.d/x11vnc start
```

```
home$ su
home# get-java-jre
home# exit
home$ firefox http://SlitazBox:5800/ultrasigned.vnc
```

2.4.7.3 X11VNC Autostart

If you'd like to start X11VNC automatically you must edit `/home/$USER/.xinitrc` with:

```
# nano /home/tux/.xinitrc
```

```
# ~/.xinitrc: Executed by slim login manager to startx X session.
# You can use F1 with Slim to change your window manager or configure
# it permanently with your personal applications.conf file.
#
. $HOME/.config/slitaz/applications.conf

##############################################################
## ATTENTION!!! INSERT THE FOLLOWING LINE AFTER .conf FILE CALL
/etc/init.d/x11vnc start &

case $1 in
     e17|enlightenment*)
...
```

DO NOT USE the *Autostart Programs* option under **`OpenBox`** or Daemon tricks, it will start a **`X11VNCserver`** before a **`X11server`** and crash the VNC after the first client connection. So you'll only connect once to the server (and it's not desirable in support environments).

2.4.7.4 Conspy and slow connections

Launch a very verbose command into a **`conspy`** and the output is displayed at full speed in the virtual console whatever your connection speed to the remote box is (even if your connection is broken).

2.4.7.5 Conspy and Busybox

`Conspy` is a busybox applet since busybox 1.17.0. It adds 2.5Kb to busybox, has better terminal support and supports some new options:

- `-c` to create missing devices (`/dev/vcsaXX` and `/dev/ttyXX`)
- `-d` for screen shot

  ```
  # conspy -nd 28 > screen28.txt
  ```

- `-s` to launch a shell

```
# conspy -cs 28
```

(no more openvt/deallocvt)

- `-x` COL `-y` LINE upper left corner position
- `-f` follow cursor with automatic scrollings

The conspy applet is enabled in the busybox package. The conspy package is no longer more useful than the recent busybox and will be removed from the packages database.

2.4.8 Wicd

author kultex, linea

Pankso's Openbox treat (page 157) is very close to **Wicd**, but if you want to set up SliTaz for not very experienced users it's recommended that you use **Wicd** for Wireless Internet Control.

Do as root:

```
# tazpkg get-install wicd
# leafpad /etc/rcS.conf
```

In `/etc/rcS.conf` remove from the section `RUN_SCRIPTS`: `network.sh` and add to the section `RUN_DAEMONS`: `wicd`

Then as user:

```
$ leafpad .config/openbox/autostart.sh
```

And add to the end:

```
# wicd Network Configuration
wicd-client &
```

Then remove the network-plugin from the LX Panel by right clicking on the network-plugin.

Reboot — don't test before rebooting, because `network.sh` is interfering with **Wicd** (even if you stop it by typing **`/etc/int.d/network.sh stop`**).

2.4.9 Alsaequal

author linea

2.4.9.1 Installation

First download alsaequal:

```
# tazpkg get-install alsaequal
```

Then create a `/home/tux/.asoundrc` file:

```
ctl.equal {
  type equal;
}

pcm.plugequal {
  type equal;
  # Modify the line below if you don't
  # want to use sound card 0.
  # slave.pcm "plughw:0,0";
  # or if you want to use with multiple applications output to dmix
  slave.pcm "plug:dmix"
}

pcm.equal {
  # Or if you want the equalizer to be your
  # default soundcard uncomment the following
  # line and comment the above line.
# pcm.!default {
  type plug;
  slave.pcm plugequal;
}
```

2.4.9.2 mpg123

Change (**cd**) into your music directory and then run:

```
$ mpg123 -a equal *
```

Or

```
$ mpg123 -a equal track1
```

Now you should be able to open up a separate terminal and use:

```
$ alsamixer -D equal
```

2.4.9.3 mpd

Just edit the `audio_output` section of your `/etc/mpd.conf`:

```
audio_output {
  type              "alsa"
  name              "equal"
  device            "plug:plugequal"
  ## format         "44100:16:2"     # optional
  ## mixer_device   "default"        # optional
  ## mixer_control  "PCM"            # optional
  ## mixer_index    "0"              # optional
}
```

And start/restart **mpd** and you should be able to use **alsamixer -D equal**

2.4.9.4 moc

Copy the `config.example` file in `/usr/share/doc/moc` to your `~/.moc` folder:

```
$ cp /usr/share/doc/moc/config.example ~/.moc/config
```

Then change the **`alsa`** output device line to:

```
# ALSA output device
AlsaDevice     = equal
```

And then start/restart **`moc`**.

2.4.9.5 References

http://www.thedigitalmachine.net/alsaequal.html

2.4.10 Xbindkeys

author sidini, linea, hgt

2.4.10.1 Introduction

`Xbindkeys` is a program that allows you to launch shell commands with your keyboard or your mouse under X Windows. It links commands to keys or mouse buttons using a configuration file. It's independent of the window manager and can capture all keyboard keys (ex: `Power`, `Wake...`).

It is handy to make "dead keys" of a multimedia keyboard work (ex: `play`, `pause`, `browser homepage...`)

2.4.10.2 Installing

Use **`tazpkg`** to quickly install **`xbindkeys`** package. Open a terminal as *root* and type:

```
# tazpkg get-install xbindkeys
```

2.4.10.3 Configuration and detecting key codes

`Xbindkeys` uses a configuration file to link a command to a key on your keyboard. Usually this file is `$HOME/.xbindkeysrc`

You can have a default one created by using:

```
$ xbindkeys --defaults > $HOME/.xbindkeysrc
```

Start the program using:

```
$ xbindkeys
```

To add a custom keyboard shortcut, first you have to detect the key code. To do this, use:

```
$ xbindkeys -k
```

When a small white window shows up on screen, just press the desired key. Terminal will show you the 3-line code of the pressed key. Example:

```
"(Scheme function)"
m:0x10 + c:180
Mod2 + XF86HomePage
```

Now, close **xbindkeys** application to make changes on configuration file:

```
$ killall xbindkeys
```

Open `.xbindkeysrc` file on **Leafpad** (or on your favorite text editor) and copy the 3-line code above to the end of the file. We need to change the first code line to link the key code to a desired function. For example, if we want to open **Midori** when `XF86Homepage` key is pressed, we must change the first line to:

```
"midori"
m:0x10 + c:180
Mod2 + XF86HomePage
```

Save and close the `.xbindkeysrc` file. Restart **xbindkeys** program and your new shortcut is already working! To add other shortcuts, repeat the procedure above.

If you want to detect a multi-key shortcut code like `Ctrl+F`, use:

```
$ xbindkeys -mk
```

2.4.10.3.1 Summary

Summary of commands:

```
# tazpkg get-install xbindkeys
$ xbindkeys --defaults > $HOME/.xbindkeysrc
$ xbindkeys
$ xbindkeys -k
$ killall xbindkeys
$ leafpad HOME/.xbindkeysrc
$ xbindkeys
```

2.4.10.3.2 Adding xbindkeys in autostarted applications

In the **PCManFM** in your `home` folder search for hidden folder `.config`, open it, next open `openbox` folder and open `autostart.sh` in your text editor. Add these lines at the end of your file:

```
# Fn Keys
xbindkeys &
```

Save and quit text editor. All custom shortcuts will stay after you reboot or switch off/on your machine.

2.4.10.4 Examples and tips

This section shows you many command codes ready to use. Key codes (last two of 3-line key code) may vary from keyboard to keyboard, so it's up to you to catch them using **`xbindkeys`**.

2.4.10.4.1 Volume control (Alsa mixer)

```
#Muter/UnMute
"amixer set "Master" toggle"
m:0x00 + c:121
XF86AudioMute

#Volume up
"amixer set "Master" 5%+"
m:0x0 + c:123
XF86AudioRaiseVolume

#Volume down
"amixer set "Master" 5%-"
m:0x0 + c:122
XF86AudioLowerVolume
```

2.4.10.4.2 Alsaplayer

```
#stop alsaplayer
"alsaplayer --pause"
m:0x0 + c:172
XF86AudioPlay

#next alsaplayer
"alsaplayer --next"
m:0x0 + c:171
XF86AudioNext

#previous alsaplayer"
"alsaplayer --prev"
m:0x0 + c:173
XF86AudioPrev
```

2.4.10.4.3 Midori Browser

```
#Open Midori at Homepage
"midori --execute Homepage"
m:0x10 + c:180
Mod2 + XF86HomePage

#Back for previous page
"midori --execute Back"
m:0x10 + c:166
Mod2 + XF86Back
```

(continues on next page)

(continued from previous page)

```
#Forward to next page
"midori --execute Forward"
m:0x10 + c:167
Mod2 + XF86Forward

#Stop loading current page
"midori --execute Stop"
m:0x10 + c:136
Mod2 + Cancel

#Reload/Refresh current page
"midori --execute Reload"
m:0x10 + c:181
Mod2 + XF86Reload
```

Note: If you use **`Firefox`** or Google **`Chrome`**, there's no need to modify `xbindkeysrc` file: these browsers automatically recognize the multimedia keys. Also, if you change from **`Midori`** to **`Firefox`** or **`Chrome`**, you must comment (#) or delete the shortcuts above. If you don't do this, it will open a **`Midori`** window when you press a navigation button.

2.4.10.4.4 Power management

```
#suspend to ram
"sudo pm-suspend"
m:0x0 + c:150
XF86Sleep
```

Note: This one (suspend to RAM) works if you install **`pm-utils`**.

```
#power off button
"poweroff"
m:0x0 + c:124
XF86PowerOff
```

2.4.10.4.5 Screenshots

```
#screenshot
"mtpaint -s"
m:0x0 + c:107
Print
```

2.4.10.4.6 Applications Menu (Start Menu)

```
#Show Start Menu with left windows-key
"lxpanelctl menu"
m:0x50 + c:133
Mod2+Mod4 + Super_L

#Show Start Menu with right windows-key
"lxpanelctl menu"
m:0x50 + c:134
Mod2+Mod4 + Super_R
```

2.4.10.5 References

Xbindkeys homepage: http://www.nongnu.org/xbindkeys/xbindkeys.html

Forum topics:

- http://forum.slitaz.org/topic/make-fn-keys-work
- http://forum.slitaz.org/topic/slitaz-40-how-to-made-screen-shots-captures-d-ecran
- http://forum.slitaz.org/topic/keyboard-shortcutshooks
- http://forum.slitaz.org/topic/keyboard-shortcuts-how-create-them
- http://forum.slitaz.org/topic/magic-sysrq-keys-power-button
- http://forum.slitaz.org/topic/power-button-shutdown
- http://forum.slitaz.org/topic/fn-key

2.4.11 TazWiKiss

author hgt, linea

2.4.11.1 Introduction

`TazWikiss` is a port of the wiki software WiKiss[173]. As it is a CGI application, it needs a web server supporting CGI and a browser — on SliTaz these are by default **`httpd`** (in busybox) and **`tazweb`**. The CGI scripts are shell scripts, so no other language interpreter is needed.

The software is installed in `/var/www/wiki` and the wiki pages are (by definition in the config file: `/var/www/wiki/config.sh`) stored in `/var/www/wiki/pages`.

The wiki pages are plain text files without any subdirectory hierarchy.

2.4.11.2 Requirements

A running web server supporting CGI and a web browser.

[173] http://wikiss.tuxfamily.org/

2.4.11.3 Start TazWiKiss

2.4.11.3.1 via a desktop menu

Accessories → Wiki documents

2.4.11.3.2 via a command

```
$ tazweb http://localhost.wiki.index.sh
```

When logged in on the same system where the web server is running, else localhost is to be replaced by a name or IP address of the system where the wiki resides.

2.4.11.4 Installation

When **TazWiKiss** is not incorporated in the flavor you installed, you can add it by installing the package **slitaz-dev-tools**:

```
# tazpkg -gi slitaz-dev-tools
```

2.4.11.5 Hints

2.4.11.5.1 Help on wiki syntax

Help on the syntax is available from the welcome page of the delivered wiki by hyperlink `help` or `helptable`.

2.4.11.5.2 Hide wiki pages

To hide pages, the user can set a password on the page by inserting

```
{PASSWORD=read-password}
```

2.4.11.5.3 Protection of wiki

To prevent unauthorised users from modifying the wiki, a password can be set in `/var/www/wiki/config.sh` (or a language specific configuration file) by replacing `PASSWORD=""` with `PASSWORD="modification-password"`.

2.5 Hardware

2.5.1 Xorg & XVesa

author kultex, jozee, linea, seawolf

By default, SliTaz v3 uses Xorg 7.4 with the **xorg-xf86-video-vesa** v2.0.0 driver. This basic driver can cause problems with certain hardware combinations and can sometimes result in a very low resolution. Occasionally, X does not start at all and instead falls back to the log-in screen or to a text-based prompt. There are some simple solutions that can yield major improvements:

- check that you have enough RAM to run the default ISO. If not, use the Low RAM ISO[174] instead.
- try the XVesa ISO[175] (a.k.a. TinyX)
- install a more specific driver for your graphics card
- customise your Xorg configuration file (`xorg.conf`)

Good introductions to Xorg Configuration can be found at The FreeBSD Handbook[176] and X Configuration from Ubuntu[177].

Tip: Up to v7.3, the `Ctrl+Alt+Backspace` key combination could be used to quit the X server. To enable it in version 7.4 and later, type the following command from any X terminal emulator:

```
# setxkbmap -option terminate:ctrl_alt_bksp
```

Important: Throughout this page, commands preceeded with a hash sign (#) should be executed as the **root** user. This is best under a terminal window. Otherwise, the dollar symbol ($) denotes a regular (tux) user.

2.5.1.1 Using Xorg & Vendor-Specific Drivers

When you are using SliTaz as your main system, it's recommended to use Xorg over XVesa. You will get a much better display and performance than when using Xvesa tinyX.

2.5.1.1.1 AGP Cards

All AGP video cards need extra kernel modules to function under Xorg. Check if you have an AGP video card with **lspci**; if so, install the necessary modules in the **linux-agp** package before using Xorg:

```
# tazpkg get-install linux-agp
```

Load the modules using the SliTaz hardware detection tool:

```
# tazhw detect-pci
```

You can now use the SliTaz X configuration tool to detect your settings:

```
# tazx
```

Select the appropriate driver for your video card from the list.

[174] http://mirror.slitaz.org/iso/3.0/flavors/
[175] http://mirror.slitaz.org/iso/3.0/flavors/slitaz-3.0-xvesa.iso
[176] http://www.freebsd.org/doc/en/books/handbook/x-config.html
[177] https://wiki.ubuntu.com/X/Config

2.5.1.1.2 DRI / DRM Problem

Sometimes, the auto-detection is not enough. Cards that require DRI / DRM[178] are supported under SliTaz, but v3 has a couple of bugs! The file `/dev/dri` should be a directory and not a file and, to get DRI working correctly, we have to add `tux` to the group `video` or modify permissions in the Xorg configuration file:

```
# tazpkg get-install linux-drm
# tazpkg get-install mesa-demos   # for glxinfo and glxgears
# rm /dev/dri
# mkdir /dev/dri/
# addgroup tux video
```

The `drm` module is not loaded by **`tazhw`** so you have to do it manually:

```
# modprobe drm
```

Tip: All modules, which are loaded by **`tazhw`** and yourself to make your changes permanent, have to be added to the **`SliTaz Control Box`** under *Initialization* in *Load Modules*.

2.5.1.1.3 Intel cards

The **`xorg-xf86-video-vesa`** 2.0.0 driver has a lot of trouble with Intel chips. For example, on a 82945GM chipset it does not display 1280×1024 and 1024×768, but 1600×1200 is not a problem.

Use **`tazx`** to select the `intel` driver, then **`tazhw detect-pci`**, solve the `dri` problem and load `drm` and restart X — normally that's it (perhaps you must change your `xorg.conf` too).

Some users may need to add the `intel_agp` module to the `xorg.conf` file:

```
Section "Module"
  # ...
  Load  "intel_agp"
  # ...
EndSection
```

for the driver to work.

Depending on the hardware, the *mode-setting* feature must be turned on or off. To turn it off append one of the following to the `kernel` line in the GRUB boot-loader configuration:

- `nomodeset`
- `i810.modeset=0`
- `i915.modeset=0`

If mode-setting is off by default and should instead be turned on, append one of the following:

- `modeset`
- `i810.modeset=1`
- `i915.modeset=1`

[178] http://www.bitwiz.org.uk/s/how-dri-and-drm-work.html

2.5.1.1.4 Trident cards

Use **tazx** to select the `trident` driver, install **mesa-dri-trident**

```
# tazpkg get-install mesa-dri-trident
```

and restart X

2.5.1.1.5 nVidia cards

SliTaz provides automatic configuration for nVidia cards. There are two drivers available, the Xorg-provided `nv` and the nVidia-provided, non-free `nvidia`. The `nv` driver should be tried first as this has been compiled for SliTaz, whereas the nVidia-supplied driver is a *binary blob* that tries to fit each and every Linux distribution.

Free Driver (nv)

To set-up the free nVidia drivers, use the SliTaz Hardware Configuration tool:

```
# tazhw setup nvidia
```

Alternatively, you can do this process manually:

- Download the following packages: **mesa**, **mesa-demos**, **linux-agp**, **xorg-xf86-video-nv**

  ```
  # tazpkg get-install xorg-xf86-video-nv
  # tazpkg get-install mesa
  # tazpkg get-install mesa-demos
  # tazpkg get-install linux-agp</code>
  ```

- Load the kernel modules

  ```
  # tazhw detect-pci
  ```

- Replace the standard `vesa` driver with `nv`

  ```
  # sed -i 's/vesa/nv/' /etc/X11/xorg.conf
  ```

Non-Free Driver (nvidia)

To set-up the non-free nVidia drivers, use the SliTaz Hardware Configuration tool with the `--non-free` switch:

```
# tazhw setup nvidia --non-free
```

Alternatively, you can do this process manually:

- Download the following packages: **mesa**, **mesa-demos**, **linux-agp**, **nvidia**

```
# tazpkg get-install nvidia
# tazpkg get-install mesa
# tazpkg get-install mesa-demos
# tazpkg get-install linux-agp
```

- Load the kernel modules

```
# tazhw detect-pci
```

- Attempt to configure the card with nVidia's tool:

```
# nvidia-xconfig
```

- To test if rendering is working,

```
# glxinfo | grep render
```

- To change nVidia's settings, use nVidia's Settings tool:

```
# nvidia-settings
```

Hiding the Logo

To hide the nVidia logo when the system boots, add the following to `/etc/X11/xorg.conf` at the end of the `Device` section:

```
Option "NoLogo" "True"
```

2.5.1.1.6 ATI cards

SliTaz provides automatic configuration for ATI cards. There are two drivers available, the Xorg-provided `radeon` and the ATI-provided, non-free `catalyst`. The `ati` driver should be tried first as this has been compiled for SliTaz, whereas the ATI-supplied driver is a *binary blob* that tries to fit each and every Linux distribution.

Free Driver (radeon)

To set-up the free ATI drivers, install the AGP & DRM modules, then use the SliTaz Hardware Configuration tool:

```
# tazpkg get-install linux-agp
# tazpkg get-install linux-drm
# tazhw setup ati</code>
```

— but perhaps this is not enough — check `/var/log/Xorg.0.log` to see if other modules must be loaded: See *DRI / DRM Problem* (page 201)

Alternatively, you can do this process manually:

- Download the following packages: **`xorg-xf86-video-ati`**, **`mesa-dri-ati`**, **`mesa-demos`**, **`linux-agp`**

```
# tazpkg get-install xorg-xf86-video-ati
# tazpkg get-install mesa-dri-ati
# tazpkg get-install mesa-demos
# tazpkg get-install linux-agp
# tazpkg get-install linux-drm
```

- Load the kernel modules

```
# tazhw detect-pci
```

- To install the free ATI driver radeon, replace `vesa` with `radeon` in the `/etc/X11/xorg.conf` file:

```
# sed -i 's/vesa/radeon/' /etc/X11/xorg.conf
```

Non-Free Driver (catalyst)

To set-up the non-free ATI drivers, use the SliTaz Hardware Configuration tool with the `--non-free` switch:

```
# tazhw setup ati --non-free
```

2.5.1.1.7 Modifying the Xorg Configuration

Normally Xorg will start successfully, but perhaps not with the correct resolution. See the *Adding Resolutions* (page 204) tips at the end of this page or sometimes search the Internet for the `xorg.conf` for your card and your monitor — or use another LiveCD and copy the working configuration file to SliTaz.

2.5.1.2 Configuring X

While X will do it's bets to auto-configure itself for your graphics card set-up, sometimes it needs a tweak. Its configuration file, `/etc/X11/xorg.conf`, is the place to customise the configuration. The best example of this is switching drivers or adding resolutions it doesn't detect.

2.5.1.2.1 Adding Resolutions

1. Include `HorizSync` and `VertRefresh` (refresh timings) in the `Monitor` section:

```
Section "Monitor"
  Identifier    "Monitor0"
  VendorName    "Monitor Vendor"
  ModelName     "Monitor Model"
  HorizSync     28-64
  VertRefresh   60
EndSection
```

2. Include a `DefaultDepth` in the `Screen` section:

```
Section "Screen"
  Identifier     "Screen0"
  Device         "Card0"
  Monitor        "Monitor0"
  DefaultDepth   24
EndSection
```

3. Add an extra `Modes` to the line in the `Display` sub-section:

```
SubSection "Display"
  Viewport   0 0
  Depth      24
  Modes      "1024x768" "800x600"
EndSubSection
```

4. Also, add the default Font paths in the `Files` section:

```
Section "Files"
  ModulePath   "/usr/lib/X11/modules"
  FontPath     "/usr/share/fonts/X11/misc/"
  FontPath     "/usr/share/fonts/X11/TTF/"
  FontPath     "/usr/share/fonts/X11/OTF"
  FontPath     "/usr/share/fonts/X11/Type1/"
  FontPath     "/usr/share/fonts/X11/100dpi/"
  FontPath     "/usr/share/fonts/X11/75dpi/"
  FontPath     "/usr/share/fonts/truetype/ttf-dejavu"
EndSection
```

- If you have DRI/DRM enabled, it may be easier to change its permission so all users can use it, rather than adding each to the `video` group. Append the following section:

```
Section "DRI"
  Mode 0666
EndSection
```

Tip: You can use **`xrandr`** to identify your monitor(s). This utility is in the **`xorg-xrandr`** package:

```
# tazpkg get-install xorg-xrandr
```

See the Debian RandR 1.2 Wiki[179] for more information.

2.5.1.2.2 Restarting Xorg

For changes to take effect, you need to restart Xorg. This can be done by logging out of your session and back in again. Choose *Logout* from the Menu and select the *Logout X session* button. If you see the SLiM log-in manager, Xorg has restarted successfully!

[179] http://wiki.debian.org/XStrikeForce/HowToRandR12

2.5.1.3 Using XVesa and Generic Drivers (TinyX)

The SliTaz-3.0-xvesa.iso[180] uses the XVesa system instead of Xorg, which offers a more generic driver at the cost of performance. It boots on nearly all computers and laptops, but can only display a 4:3 resolution.

To find out which resolutions are possible with your card, type:

```
# Xvesa -listmodes
```

Here's a selection of resolutions for an example Intel Atom Board:

```
VBE version 3.0 (Intel(r) 82945GM Chipset Family Graphics Chip Accelerated␣
↪VGA BIOS)
DAC is switchable, controller is VGA compatible, RAMDAC causes snow
Total memory: 7872 kilobytes
0x015A: 1600x1200x24 TrueColor [8:8:8:8]
0x011B: 1280x1024x24 TrueColor [8:8:8:8]
0x0118: 1024x768x24 TrueColor [8:8:8:8]
0x0112: 640x480x24 TrueColor [8:8:8:8]
0x0115: 800x600x24 TrueColor [8:8:8:8]
```

This output shows widescreen resolutions cannot be displayed with XVesa.

If X does not start with the default SliTaz ISO, you could use the XVesa ISO to install your default Xorg driver. When you are asked for your resolution at boot-time, scroll down the window and find the option to install your Xorg driver *before* XVesa is started. However, don't use **`tazx`** and **`tazhw setup ati or nv`** on the XVesa ISO to install your Xorg driver because you will end up with a blank and confused screen!

2.5.2 Install the latest non-free Nvidia driver

author jozee, linea, domcox

2.5.2.1 Required Reading

- Handbook — Xorg Installation[181]
- Quickstart Guide — *Build your own custom Linux Kernel for SliTaz* (page 237)

2.5.2.2 Prepare

- Download the installer from the Nvidia website[182]. Latest version 190.53
- Install the Xorg server

  ```
  # tazpkg get-install xorg-server
  ```

- Install the **`linux-source`** package and development tools. See *Build your own custom Linux Kernel for SliTaz* (page 237)

[180] http://mirror.slitaz.org/iso/3.0/flavors/slitaz-3.0-xvesa.iso
[181] http://doc.slitaz.org/en:handbook:xwindow#xorg
[182] http://www.nvidia.com/object/unix.html

```
# tazpkg get-install linux-source
```

2.5.2.3 Install the driver

For this part you're going to need a pencil and paper as we now have to work in text mode without a X-server running.

When you're ready press `Alt+Ctrl+Del`, you should now see a command line:

- Configure the X-server and copy the (generated) `xorg.conf`

```
# Xorg -configure
# cp /root/xorg.conf.new /etc/X11/xorg.conf
```

- Prepare the Kernel

```
# cd /usr/src/linux

# make oldconfig && make prepare

# make menuconfig  # (not required - but if you've come this far, you␣
↪can take a peek)
# make bzImage
# make modules
# make modules_install
```

- Now make executable and install the Nvidia driver, change to directory you installed to

```
# chmod +x NVIDIA-Linux-x86-177.80.pkg1.run
# ./NVIDIA-Linux-x86-177.80.pkg1.run --kernel-source-path=/usr/src/
↪linux
```

- Copy the Kernel Image to `/boot`

```
# cd /usr/src/linux
# cp arch/x86/boot/bzImage /boot
```

2.5.2.4 Restart

- Reboot into text mode

```
# reboot                          # Hard drive users
# tazusb writefs gzip && reboot   # USB users
```

  Don't forget to pass the `screen=text` option at startup, (it may well boot into text mode anyway, if not just press `Alt+Ctrl+Del` again)

- Load the `nvidia` module

```
# modprobe -v nvidia
```

- Edit SLiM configuration file to load Xorg server

```
# vi /etc/slim.conf
```

```
default_xserver     /usr/bin/Xorg
#default_xserver     /usr/bin/Xvesa
#xserver_arguments   -ac -shadow dpms +extension Composite -screen␣
↪1024x768x24
```

- Restart SLiM

```
# /etc/init.d/slim start
```

Err. . . that's it

(If you want the `nvidia` module to persist, just add it to the `LOAD_MODULES` variable in `/etc/rcS.conf`)

2.5.3 Poulsbo (aka GMA500)

author o, llev, mojo

This is a short guide on getting the Poulsbo graphics working nicely with SliTaz 4.0 or new rolling-core.

Boot SliTaz and establish a working network connection. After a successful installation you will need to do the following steps to get the graphics working:

```
# nano /etc/rcS.conf
```

Look for the row called `LOAD_MODULES` and remove `poulsbo` from the list, also make sure that `psb_gfx` is present.

Then remove VESA and add the `psb_gfx` module (which is inside the **`linux-staging`** package):

```
# tazpkg remove xorg-xf86-video-vesa
# tazpkg recharge
# tazpkg get-install linux-staging
```

The fbdev-driver needs to be installed/updated:

```
# tazpkg -gi xorg-xf86-video-fbdev
```

Simply reboot and the graphics should work nicely with the correct resolution.

2.5.4 Webcam on SliTaz

author pankso, linea

SliTaz provides all the necessary tools to take pictures, record and stream webcam output.

2.5.4.1 Installation

By default your webcam may not be detected due to missing kernel drivers and this is normal since SliTaz doesn't provide webcam support on the LiveCD. To install and load the drivers:

```
# tazpkg -gi linux-media
# modprobe uvcvideo
```

You can check if the kernel correctly loaded the module with:

```
$ dmesg | tail
```

The webcam may be used with **Skype** and other visual tools. To try the webcam with **Mplayer**:

```
$ mplayer tv:// -tv driver=v4l2:device=/dev/video0 -vo x11
```

2.5.4.2 Pictures and recording

To take pictures or record using the webcam you can install **Mplayer**:

```
# tazpkg -gi mplayer
```

Take a picture:

```
mplayer tv:// -tv driver=v4l2:width=640:height=480:device=/dev/video0 \
-fps 15 -vo png -frames 1
```

Record a video to a file named `myvideo.avi`:

```
mencoder tv:// -tv \
driver=v4l2:width=640:height=480:device=/dev/video0:forceaudio:adevice=/
↪dev/dsp \
-ovc lavc -oac mp3lame -lameopts cbr:br=64:mode=3 \
-o myvideo.avi
```

2.5.5 Printing

author papagoose, jozee, linea, seawolf

The CUPS (Common Unix Printing System) software manages printers connected to the local computer or over a network. The local printers are attached via a loop-back connection (IP 127.0.0.1), a network connection pointing to the same system.

2.5.5.1 Installing a Network Printer

Install CUPS and add the user(s) to the `lp` group to gain permissions to access the devices. This can be done as root in a terminal:

```
# tazpkg get-install cups
# addgroup tux lp
```

Again as root, customise the default configuration in `/etc/cupsd.conf`:

```
# Administrator user group...
SystemGroup lp

```

(continues on next page)

(continued from previous page)

```
# Restrict access to the admin pages...

  Order allow,deny
  Allow 127.0.0.1


# Restrict access to configuration files...

  AuthType Default
  Require user @SYSTEM
  Order allow,deny
  Deny From None
  Allow From 127.0.0.1
```

Set the administrative password for CUPS and add an administrative user:

```
# lppasswd -g lp -a tux
```

You should now be able to have access to the CUPS administration through your browser by entering the address http://localhost:631/

Install the printer drivers with the following packages:

- **hplip**: for HP printers
- **gutenprint**: for Canon, Epson, Lexmark, Sony, Olympus
- **foomatic-db**, **foomatic-db-engine**, **foomatic-db-nonfree** and **foomatic-filters**: several free software printer drivers
- **ufr2**: for Canon printers (extra drivers)
- **splix**: for Samsung
- **cups-pdf**: for "printing" to PDF files

For example, install the HP printer drivers with the following *root* command

```
# tazpkg get-install hplip
```

Now you add a new printer in the browser interface and choose: *LPD/LPR Host or Printer (Unknown)*, and add the address socket//192.168.2.1:9100

If you get the error, "*Returning IPP client-error-document-format-not-supported for Print-Job in /var/log/cups/error.log*" then add two files to the `/etc/cups` directory. The easiest method is to start a text editor such as **Leafpad** from a root terminal; start the Terminal and switch to the root user.

- `mime.convs` (`/etc/cups/mime.convs`):

```
application/pdf          application/postscript          33  pdftops
application/postscript   application/vnd.cups-postscript 66  pstops
application/vnd.hp-HPGL  application/postscript          66  hpgltops
application/x-cshell     application/postscript          33  texttops
application/x-csource    application/postscript          33  texttops
application/x-perl       application/postscript          33  texttops
application/x-shell      application/postscript          33  texttops
text/plain               application/postscript          33  texttops
text/html                application/postscript          33  texttops
```

(continues on next page)

(continued from previous page)

```
image/gif                 application/vnd.cups-postscript 66  imagetops
image/png                 application/vnd.cups-postscript 66  imagetops
image/jpeg                application/vnd.cups-postscript 66  imagetops
image/tiff                application/vnd.cups-postscript 66  imagetops
image/x-bitmap            application/vnd.cups-postscript 66  imagetops
image/x-photocd           application/vnd.cups-postscript 66  imagetops
image/x-portable-anymap   application/vnd.cups-postscript 66  imagetops
image/x-portable-bitmap   application/vnd.cups-postscript 66  imagetops
image/x-portable-graymap  application/vnd.cups-postscript 66  imagetops
image/x-portable-pixmap   application/vnd.cups-postscript 66  imagetops
image/x-sgi-rgb           application/vnd.cups-postscript 66  imagetops
image/x-xbitmap           application/vnd.cups-postscript 66  imagetops
image/x-xpixmap           application/vnd.cups-postscript 66  imagetops
image/x-sun-raster        application/vnd.cups-postscript 66  imagetops


image/gif                 application/vnd.cups-raster 100 imagetoraster
image/png                 application/vnd.cups-raster 100 imagetoraster
image/jpeg                application/vnd.cups-raster 100 imagetoraster
image/tiff                application/vnd.cups-raster 100 imagetoraster
image/x-bitmap            application/vnd.cups-raster 100 imagetoraster
image/x-photocd           application/vnd.cups-raster 100 imagetoraster
image/x-portable-anymap   application/vnd.cups-raster 100 imagetoraster
image/x-portable-bitmap   application/vnd.cups-raster 100 imagetoraster
image/x-portable-graymap  application/vnd.cups-raster 100 imagetoraster
image/x-portable-pixmap   application/vnd.cups-raster 100 imagetoraster
image/x-sgi-rgb           application/vnd.cups-raster 100 imagetoraster
image/x-xbitmap           application/vnd.cups-raster 100 imagetoraster
image/x-xpixmap           application/vnd.cups-raster 100 imagetoraster
image/x-sun-raster        application/vnd.cups-raster 100 imagetoraster

application/vnd.cups-postscript application/vnd.cups-raster 100␣
↪pstoraster
```

- `mime.types` (`/etc/cups/mime.types`):

```
application/pdf             pdf string(0,%PDF)
application/postscript      ai eps ps string(0,%!) string(0,<04>%!) \
    contains(0,128,<1B>%-12345X) + \
    (contains(0,4096,"LANGUAGE=POSTSCRIPT") \
     contains(0,4096,"LANGUAGE = Postscript") \
     contains(0,4096,"LANGUAGE = PostScript") \
     contains(0,4096,"LANGUAGE = POSTSCRIPT") \
     (contains(0,4096,<0a>%!) + \
      !contains(0,4096,"ENTER LANGUAGE")))
application/vnd.hp-HPGL             hpgl \
    string(0,<1B>E<1B>%0B) \
    string(0,<1B>%-1B) string(0,<201B>)\
    string(0,BP;) string(0,IN;) string(0,DF;) \
    string(0,BPINPS;) \
    (contains(0,128,<1B>%-12345X) + \
     (contains(0,4096,"LANGUAGE=HPGL") \
      contains(0,4096,"LANGUAGE = HPGL")))

######################################################################
#
```

(continues on next page)

(continued from previous page)

```
# Image files...
#

image/gif                 gif string(0,GIF87a) string(0,GIF89a)
image/png                 png string(0,<89>PNG)
image/jpeg                jpeg jpg jpe string(0,) &&\
    (char(3,0xe0) char(3,0xe1) char(3,0xe2) char(3,0xe3)\
     char(3,0xe4) char(3,0xe5) char(3,0xe6) char(3,0xe7)\
     char(3,0xe8) char(3,0xe9) char(3,0xea) char(3,0xeb)\
     char(3,0xec) char(3,0xed) char(3,0xee) char(3,0xef))
image/tiff              tiff tif string(0,MM<002A>) string(0,II<2A00>)
image/x-photocd           pcd string(2048,PCD_IPI)
image/x-portable-anymap   pnm
image/x-portable-bitmap   pbm string(0,P1) string(0,P4)
image/x-portable-graymap  pgm string(0,P2) string(0,P5)
image/x-portable-pixmap   ppm string(0,P3) string(0,P6)
image/x-sgi-rgb           rgb sgi bw icon short(0,474)
image/x-xbitmap           xbm
image/x-xpixmap           xpm ascii(0,1024) + string(3,"XPM")
#image/x-xwindowdump      xwd string(4,<00000007>)
image/x-sun-raster        ras string(0,<59a66a95>)

#image/fpx                fpx
image/x-alias             pix short(8,8) short(8,24)
image/x-bitmap            bmp string(0,BM) && !printable(2,14)
image/x-icon              ico

######################################################################
#
# Text files...
#

application/x-cshell      csh printable(0,1024) + string(0,#!) +\
    (contains(2,80,/csh) contains(2,80,/tcsh))
application/x-perl        pl printable(0,1024) + string(0,#!) +\
    contains(2,80,/perl)
application/x-shell       sh printable(0,1024) + string(0,#!) +\
    (contains(2,80,/bash) contains(2,80,/ksh)\
     contains(2,80,/sh) contains(2,80,/zsh))
application/x-csource     c cxx cpp cc C h hpp \
    printable(0,1024) + \
    (string(0,/*) string(0,//)
     string(0,#include) contains(0,1024,<0a>#include) \
     string(0,#define) contains(0,1024,<0a>#define))
text/html                 html htm printable(0,1024) +\
    (istring(0,"") istring(0,"))
text/plain                txt printable(0,1024)
text/css                  css


######################################################################
#
# RSS feed type...
#

application/rss+xml       rss
```

(continues on next page)

(continued from previous page)

```
########################################################################
#
# CUPS-specific types...
#

application/vnd.cups-command     string(0,'#CUPS-COMMAND')
application/vnd.cups-form        string(0,"")
application/vnd.cups-pdf
application/vnd.cups-postscript
application/vnd.cups-ppd         ppd string(0,"*PPD-Adobe:")
application/vnd.cups-raster      string(0,"RaSt") string(0,"tSaR")
application/vnd.cups-raw      (string(0,<1B>E) + !string(2,<1B>%0B)) \
    string(0,<1B>@) \
    (contains(0,128,<1B>%-12345X) + \
     (contains(0,4096,"LANGUAGE=PCL") \
      contains(0,4096,"LANGUAGE = PCL")))

########################################################################
#
# Raw print file support...
#
# Comment the following type to prevent raw file printing.
#

application/octet-stream
```

Tip: If at the end of the procedures the printer is not working, restarting the service or computer may help.

2.5.5.2 Installing an USB Brother HL 2030 Printer

To install **cups**, **hal-cups-utils**, **usbutils**. As root, type:

```
root@slitaz:# tazpkg get-install cups
root@slitaz:# tazpkg get-install hal-cups-utils
root@slitaz:# tazpkg get-install usbutils
```

Cups is used to manage the printer, **hal-cups-utils** allows CUPS to use HAL for printer connections and **usbutils** gives us the **lsusb** utility which lets us know how the printer is connected.

Now we can add `tux` to the `lp` (printer) group. As root, we do:

```
root@slitaz:# addgroup tux lp
```

For the web interface of CUPS to be properly activated, we still need to change a few things in the `/etc/cupsd.conf` file:

```
root@slitaz:# leafpad /etc/cups/cupsd.conf
```

```
# Administrator user group...
SystemGroup lp

# Restrict access to the admin pages...

Order allow,deny
Allow 127.0.0.1


# Restrict access to configuration files...

AuthType Default
Require user @SYSTEM
Order allow,deny
Deny From None
Allow From 127.0.0.1
```

To modify permissions on the printer you need to know the details of the bus and device. For this, we do as root:

```
root@slitaz:# lsusb
```

And get the following output:

```
Bus 001 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 001 Device 002: ID 04f9:0027 Brother Industries, Ltd HL-2030 Laser␣
↪Printer
```

You can now change the permissions on the associated file:

```
root@slitaz:# chmod 666 /dev/bus/usb/001/002
```

And then start CUPS:

```
root@slitaz:# /etc/init.d/cupsd start
```

We can now configure the printer and fetch the openprinting ppd file for that printer model: http://www.openprinting.org/printer/Brother/Brother-HL-2030 and also add the following packages: **foomatic-filters**, **foomatic-db**, **foomatic-db-engine**:

```
root@slitaz:# tazpkg get-install foomatic-filters
root@slitaz:# tazpkg get-install foomatic-db
root@slitaz:# tazpkg get-install foomatic-db-engine
```

Now we can restart CUPS and point midori to http://localhost:631

CUPS then asks for:

1. The login "root" (default tux)
2. The corresponding password.

Then navigate to *Printers* → *Add Printer*, click on the printer name that you recognize and do not forget to indicate the path in the connection box: `/dev/bus/usb/001/002` (in my case).

Then go to the web interface of CUPS ppd and install the new printer using the file `Brother HL-2030-hl1250.ppd` located in the user's account. Now you can automatically restart **cupsd** each time you start the system by using menu *System Tools* → *Control box* → *Initialization* and allow the **cupsd** daemon to run by adding it to the *Run daemons* section:

```
dbus hald firewall slim cupsd
```

2.5.5.3 Installing a HP Printer

Note: This is for a clean installation of SliTaz GNU/Linux cooking-20100314

1. With the printer plugged in and powered on, run

   ```
   # su root
   # tazpkg recharge
   # tazhw setup printer
   ```

2. Install only **hplip** and **hal-cups-utils**
3. The printer should show up in the list of printers displayed as part of this command. It will then attempt to open the CUPS admin pages in Midori (at http://localhost:631)
4. On my system, the printer does not show up in the browser under *Find printers* and I get a 'Forbidden' error when trying to add a printer via the browser
5. Instead, run

   ```
   # hp-setup
   ```

   and follow the instructions. The test page should print correctly. The printer should also show up in the list of printers in applications (like the text editor) and will also appear in the list of printers on the CUPS browser pages, although in my case it is still not possible to make any changes to it.

2.5.5.4 Installing a HP All-In-One Printer/Scanner

2.5.5.4.1 Preparation & Packages

Power on the scanner before installing the following packages:

- **xsane**
- **sane-backends**
- **libusb**
- **libusb-compat**
- **usbtools**
- **usbutils**

Tip: You can automate the process with the following BASH command as *root* user:

```
for PKG in xsane sane-backends-libusb libusb-compat usbtools usbutils; do
    tazpkg get-install $PKG
done
```

2.5.5.4.2 Detection

With those packages installed, use the *Hardware Detection Tool* (*Menu* → *System Tools* → *Hardware Detection And Drivers*). Click on *Scanner* button.

If your scanner does not show up immediately, enter for none in my set-up; now the scanner device should be listed in purple text, e.g.:

```
/dev/bus/usb/004/002
```

2.5.5.4.3 Verify Permissions

Your scanner must be in scanner group with 666 permissions

```
crw-rw-rw-    1 root     scanner  189, 385 Jun 22 19:44 /dev/bus/usb/0
```

Reply `y` to start scanner. Warning pops up about running scanner as root. Click *continue at your own risk* button Agree to license.

`Xsane` should open and be working!

The *tux* user is automatically added to `scanner` group, so *tux* may scan, but if you run under another user name that user can't scan until you add them to `scanner` group. Do this with the command (as *root*):

```
# addgroup <username> scanner
```

2.5.5.5 Links

- http://www.openprinting.org/printers

2.5.5.6 TODO

- Parallel Printer

2.5.6 ACPI (Advanced Configuration and Power Interface)

author linea, kultex

To install the **`acpi`** daemon and modules:

```
# tazpkg get-install acpid
# tazpkg get-install linux-acpi
```

The **acpi** daemon needs to be started before **dbus** and **hal**, so edit your `/etc/rcS.conf` to look like this:

```
RUN_DAEMONS="acpid, dbus, hald, etc..."
```

Or, to start manually:

```
# /etc/init.d/hald stop
# /etc/init.d/dbus stop
# /etc/init.d/acpid start
# /etc/init.d/dbus start
# /etc/init.d/hald start
```

To check it's working, just:

```
$ cat /proc/acpi/battery/{BAT0}/info
```

You can find some example scripts in `/etc/acpi`.

2.5.7 CPU Frequency Scaling

author kultex, linea

2.5.7.1 Introduction

To reduce power consumption is not only essential for portable computers: it is also essential to use it on desktop machines to reduce unnecessary CO_2 emissions. You can save power by turning off not needed hardware components like WiFi, Bluetooth, etc., switching off or reducing the monitor backlight, spinning down HDDs and controlling the CPU frequency.

CPU frequency scaling is built into the 2.6 kernel and available; but because Slitaz is very small and light, you have to install some additional tools and set it up by yourself.

2.5.7.2 Installation

Install **linux-cpufreq**, **linux-acpi**, **cpufrequtils** and optional **powertop** (for easy controlling) — **cpufrequtils** and **powertop** we have to take from cooking, because **cpufrequtils** does not exist in 3.0 and **powertop** is buggy.

```
# tazpkg get-install linux-cpufreq
# tazpkg get-install linux-acpi,
# wget http://mirror.slitaz.org/packages/cooking/cpufrequtils-008.tazpkg
# wget http://mirror.slitaz.org/packages/cooking/powertop-1.13.tazpkg
# tazpkg install cpufrequtils-008.tazpkg
# tazpkg install powertop-1.13.tazpkg
```

2.5.7.3 CPU driver

You have to load the correct CPU kernel driver for your CPU — if you don't know, which CPU is in your PC you can get it from:

```
$ cat /proc/cpuinfo
```

Depending on the CPU, load one of the following modules — if you choose the wrong driver, you will get an error message, the module is not loaded and nothing goes wrong.

Generic ACPI P-States based driver:

```
# modprobe acpi-cpufreq
```

AMD mobile K6-2/3+ PowerNow!:

```
# modprobe powernow-k6
```

AMD mobile Athlon PowerNow!:

```
# powernow-k7
```

AMD Cool&Quiet PowerNow!(up to AMD “K10” CPU):

```
# modprobe powernow-k8
```

Intel SpeedStep using the SMI BIOS interface:

```
# modprobe speedstep-smi
```

Intel SpeedStep on ICH-based chipsets:

```
# speedstep-ich
```

Intel Enhanced SpeedStep (deprecated — use **`acpi-cpufreq`**):

```
# modprobe speedstep-centrino
```

Intel Pentium4/Xeon — The kernel documentation says

> This adds the CPUFreq driver for Intel Pentium 4 / XEON processors. When enabled it will lower CPU temperature by skipping clocks. This driver should be only used in exceptional circumstances when very low power is needed because it causes severe slowdowns and noticeable latencies. Normally Speedstep should be used instead.

— the p4-clockmod supports only `performance` and `powersave` governors (due to the long transition latency of the module itself):

```
# modprobe p4-clockmod
```

NatSemi Geode GX / Cyrix MediaGXm:

```
# modprobe gx-suspmod
```

Transmeta Crusoe / Efficeon LongRun:

```
# modprobe longrun
```

VIA Cyrix Longhaul:

```
# /sbin/modprobe longhaul
```

nForce2 FSB changing cpufreq driver:

```
# modprobe cpufreq-nforce2
```

Enhanced PowerSaver driver for VIA C7 CPUs:

```
# modprobe e_powersaver
```

2.5.7.4 Scaling governors

Governors can be thought of as pre-configured power schemes for the CPU.

Available governors:

`cpufreq_performance` (default in SliTaz) The `performance` governor is built into the kernel and runs the CPU(s) at maximum clock speed

`cpufreq_ondemand` (also built into the kernel) Dynamically increases/decreases the CPU(s) clock speed based on system load

`cpufreq_conservative` Similar to `ondemand`, but more conservative (clock speed changes are more graceful)

`cpufreq_powersave` Runs the CPU at minimum speed

`cpufreq_userspace` Manually configured clock speeds by user

The built in governors must not be loaded — the others must be loaded with **`modprobe`** — you may load as many governors as desired (only one will be active at any given time).

Which poses the question, which governor saves most energy? The `Powersave` governor will only save you power if you're playing 3D games (it does not save power because application process completion times are prolonged at the lower processor frequency and the system does not enter a deep C-state. The greatest power savings occur at idle in deeper C-states) and the `performance` governor will basically never give you extra performance. Don't use them — therefore I am not happy that the `performance` governor is used by default in SliTaz. So you might choose between `ondemand` and `conservative`, which is a matter of taste.

The `userspace` governor is necessary for third party applications — like cpupowerd[183] — unfortunately it does not work in SliTaz, because the kernel config for `CONFIG_X86_MSR` is not set.

2.5.7.5 Select the governor

At this point everything is prepared but no active governor selected — so no power saving yet.

Use the **`cpufreq-set`** command to activate one of the governors — for example:

```
cpufreq-set -g ondemand
```

Please note, if you have a dual-core or multiple-core CPU, you must explicitly specify the CPU. There are CPUs where each core can run with different settings! Example for a dual-core CPU:

[183] http://www.themaxer.com/index.php?option=com_content&view=article&id=86:undervolting-your-cpu&catid=41:nas&Itemid=81

```
# cpufreq-set -c 0 -c 1 -g conservative
```

2.5.8 SliTaz and EFI computers

author ceel, linea

2.5.8.1 HOWTO do a SliTaz LiveUSB for EFI computers

Create a “universal” LiveUSB of SliTaz Next that will start on every computer.

You can also to do it with another version of SliTaz. Although, keep in mind that recent hardware can’t be supported; in this case, SliTaz will boot in console mode.

HOWTO do a SliTaz LiveUSB for EFI computers (page 220)

2.5.8.2 HOWTO do a frugal installation on a computer in UEFI Boot Mode

Run SliTaz in RAM from your hard disk.

Realize a frugal installation on the Windows main partition.

HOWTO do a frugal installation on a computer in UEFI boot mode (page 226)

2.5.8.3 HOWTO do a full installation on a computer in UEFI boot mode

Install SliTaz in dual boot with Windows (In progress) ;-)

2.5.9 HOWTO do a SliTaz LiveUSB for EFI computers

author ceel, linea

Here starts my trilogy about UEFI & SliTaz ;-)

This first tutorial describes how to do an EFI LiveUSB of SliTaz Next from a SliTaz Rolling session.

It doesn’t matter that the operating system is 32bit or if your EFI computer is 64bit. The only imperative is a 64bit computer needs a 64bit EFI boot loader to start.

2.5.9.1 Preparing the USB stick

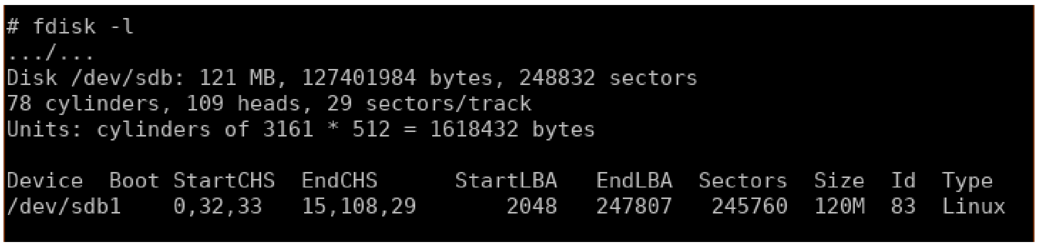
Insert a USB stick; if your file manager is set to mount removable media automatically when they are inserted, unmount the stick but don't eject it.

Open a terminal as the root user and change in `/root`:

```
tux@slitaz:~$ su
Password:
root@slitaz:/home/tux# cd /root
root@slitaz:~#
```

Identify the stick:

```
# fdisk -l
```



```
# fdisk -l
.../...
Disk /dev/sdb: 121 MB, 127401984 bytes, 248832 sectors
78 cylinders, 109 heads, 29 sectors/track
Units: cylinders of 3161 * 512 = 1618432 bytes

Device  Boot StartCHS   EndCHS      StartLBA   EndLBA  Sectors  Size  Id  Type
/dev/sdb1    0,32,33    15,108,29       2048   247807   245760  120M  83  Linux
```

Important: All the commands in the rest of this document will assume your stick is identified as `sdb1`.

If not, **replace all sdb1** occurrences with your device identification, ie: `sdc1`.

If the stick is not FAT32, format it (the packages **`dosfstools`** and **`mtools`** must be installed on your system):

```
# mkfs.vfat /dev/sdb1
```

Important: Don't forget to indicate the partition number.

All the data on the partition sdb1 will be destroyed.

Set the flags `boot` and `lba` to your stick:

```
# parted /dev/sdb set 1 boot on
# parted /dev/sdb set 1 lba on
```

Don't pay attention to the messages Information: You may need to update `/etc/fstab`.

2.5.9.2 Creating the LiveUSB

Download the SliTaz Next ISO[184].

```
# wget http://mirror1.slitaz.org/iso/next/slitaz-next-170930.iso
```

Here are two possibilities: use **tazusb** or copy the files manually.

2.5.9.2.1 Using tazusb

In the terminal, type:

```
# tazusb gen-iso2usb slitaz-next-170930.iso /dev/sdb1
```

When **tazusb** has finished the job, exit and mount the stick:

```
# exit
# mount /dev/sdb1 /mnt
```

Go to *Installing an EFI boot loader* (page 222) step.

2.5.9.2.2 Copying files manually

Mount the stick:

```
# exit
# mount /dev/sdb1 /mnt
```

Mount the ISO file:

```
# mount -o loop slitaz-next-170930.iso /media/cdrom
```

Copy the files to the stick:

```
# cp -r /media/cdrom/boot /mnt
```

Unmount the ISO:

```
# umount /media/cdrom
```

2.5.9.3 Installing an EFI boot loader

Create a `/efi/boot` directory on your stick:

```
# mkdir -p /mnt/efi/boot
```

We are going to install **GRUB** 2 but it exists on other boot loaders compatible with EFI.

[184] http://mirror1.slitaz.org/iso/next/slitaz-next-170930.iso

2.5.9.3.1 32bit EFI systems

Install the **grub2-efi** package:

```
# tazpkg -gi grub2-efi
```

Copy the boot loader in the `/efi/boot` directory you previously created on your stick:

```
# cp /boot/efi/boot/bootia32.efi /mnt/efi/boot/bootia32.efi
```

2.5.9.3.2 64bit EFI systems

Download the bootx64.efi[185] file from the Next64 project.

```
wget http://cook.slitaz.org/next64/grub2/browse/taz/grub2-efi-2.02/fs/boot/
↪efi/boot/bootx64.efi
```

Copy the boot loader in the `/efi/boot` directory you previously created on your stick:

```
# cp bootx64.efi /mnt/efi/boot/bootx64.efi
```

2.5.9.4 Configuration file for GRUB 2

Create a `/boot/grub` directory on your stick:

```
# mkdir -p /mnt/boot/grub
```

With your preferred text editor (ie **nano**), create a configuration file for **GRUB** 2:

```
# nano /mnt/boot/grub/grub.cfg
```

Example:

```
### CONFIGURATION FILE FOR GRUB 2 ###
#

# Comment the line if you want SliTaz to start automatically at boot or
# change the value to define the time (seconds) to wait before booting.
set TIMEOUT=-1

# Menu 0: boot SliTaz Next
menuentry "SliTaz Next (ISO 20170930 - kernel 4.9.30)" {
   set root=(hd0,1)
   linux /boot/bzImage ro root=/dev/null video=-32 autologin
   initrd /boot/rootfs.gz
}
```

Tip: Add your keyboard configuration in the linux line; ie for a french keyboard:

[185] http://cook.slitaz.org/next64/grub2/browse/taz/grub2-efi-2.02/fs/boot/efi/boot/bootx64.efi

```
linux /boot/bzImage ro root=/dev/null video=-32 kmap=fr-latin1 autologin
```

Unmount the stick:

```
# umount /mnt
```

Your Live is ready. But maybe the hardest remains to do.

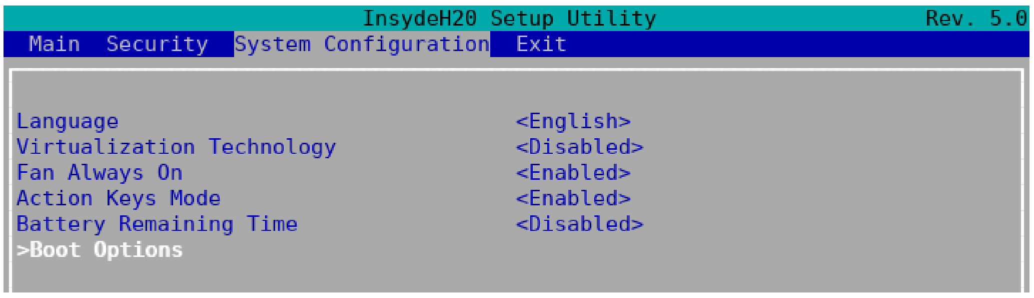
2.5.9.5 Configuring the computer

Computers that came with Windows 8 / Windows 10 preinstalled start in **Boot mode** = UEFI and have the **Secure boot** = Enabled. The **Secure boot** prevents the loading of drivers or OS loaders that are not signed with a digital signature ($old by Microsoft).

With SliTaz not having any digital signatures, you have to disable the secure boot. The following lines describe how to do it on a **hp** Laptop 17-bs032 but it can be different for you; there are almost as many ways as there are manufacturers. . . :-/

Even access to the *Setup* is different from one computer to another. If you don't know how to access the setup, have a look here[186] (sorry for French; I couldn't find an equivalent link in English).

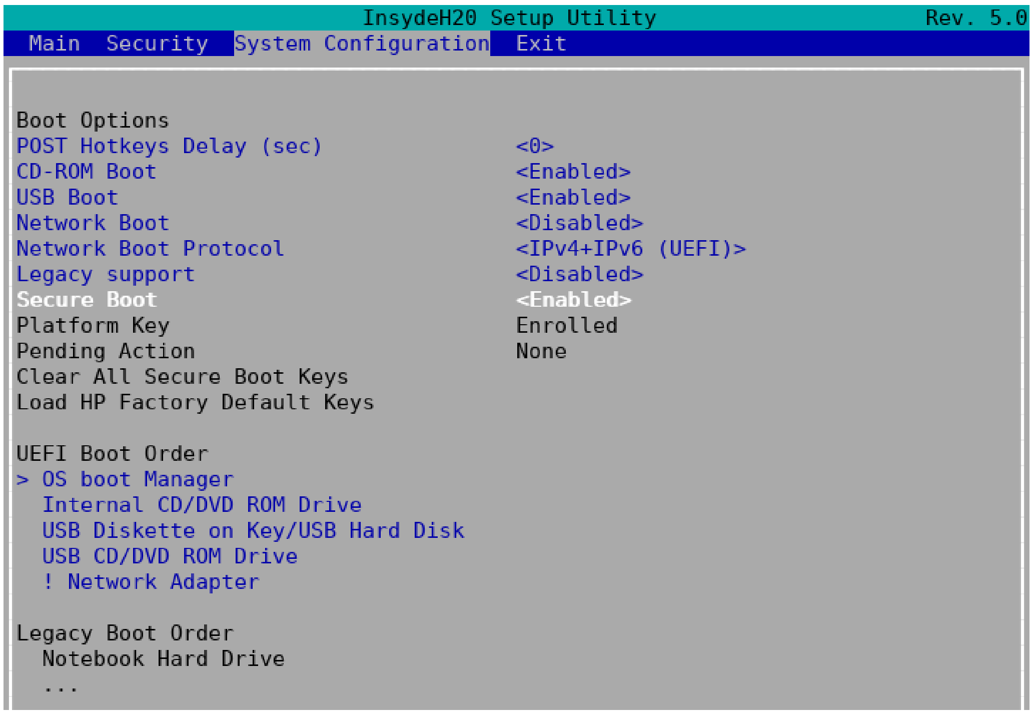
On the **hp** Laptop 17-bs032, press the `F10` key immediately after you've powered on the computer until the BIOS Setup Main page is displayed. Use the right arrow to display the *System Configuration* page.



```
                   InsydeH20 Setup Utility                     Rev. 5.0
 Main  Security  System Configuration  Exit

Language                                <English>
Virtualization Technology               <Disabled>
Fan Always On                           <Enabled>
Action Keys Mode                        <Enabled>
Battery Remaining Time                  <Disabled>
>Boot Options
```

Select >*Boot Options*

[186] https://doc.ubuntu-fr.org/tutoriel/modifier_ordre_amorcage_du_bios#liste_des_touches_pour_acceder_au_bios_et_au_boot_menu



```
                         InsydeH20 Setup Utility                     Rev. 5.0
  Main  Security  System Configuration  Exit

 Boot Options
 POST Hotkeys Delay (sec)                     <0>
 CD-ROM Boot                                  <Enabled>
 USB Boot                                     <Enabled>
 Network Boot                                 <Disabled>
 Network Boot Protocol                        <IPv4+IPv6 (UEFI)>
 Legacy support                               <Disabled>
 Secure Boot                                  <Enabled>
 Platform Key                                 Enrolled
 Pending Action                               None
 Clear All Secure Boot Keys
 Load HP Factory Default Keys

 UEFI Boot Order
 > OS boot Manager
   Internal CD/DVD ROM Drive
   USB Diskette on Key/USB Hard Disk
   USB CD/DVD ROM Drive
   ! Network Adapter

 Legacy Boot Order
   Notebook Hard Drive
   ...
```

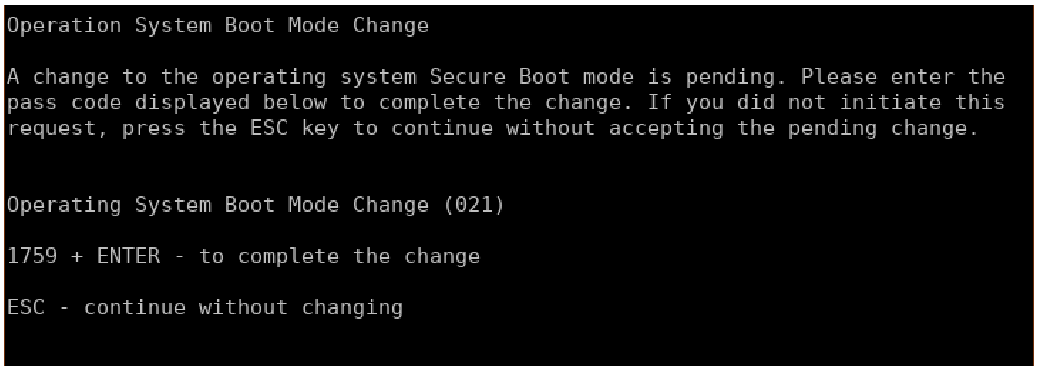
Warning: Don’t modify Setup parameters if you don’t understand what you are doing!

If you’re not sure or don’t remember what you’ve done/changed, Exit Setup without saving!

Select *Secure Boot* and set it to *Disabled*. Then, in *UEFI Boot Order*, place *USB Diskette on Key/USB Hard Disk* at the top of the list.

Tip: If you prefer not to change the UEFI Boot Order, you can use the Multiboot feature to boot your LiveUSB.

Exit Setup saving changes; the **hp** Laptop 17-bs032 reboots and prompts you to confirm your changes:



```
Operation System Boot Mode Change

A change to the operating system Secure Boot mode is pending. Please enter the
pass code displayed below to complete the change. If you did not initiate this
request, press the ESC key to continue without accepting the pending change.


Operating System Boot Mode Change (021)

1759 + ENTER - to complete the change

ESC - continue without changing
```

Well, it wasn’t so hard. But we’ll see when proceeding at the full installation on an acer Aspire v3-111p that it is quite different.

2.5.9.6 Run your Live

Warning: Windows 8 & 10 use both the **Fast Startup**. This feature consists to store the entire configuration in a file named `hiberfil.sys` when you shutdown the computer. At next boot Windows will load the file; this is faster than to load all the drivers.

When Linux finds an `hiberfil.sys` file, it refuses to mount the partition. If you still try to access to the partition and worst if you wrote on it, you can corrupt the file system and Windows won't start anymore. **You must disable the Fast Startup if you want to access your hard disk!** But this HOWTO is already long enough; we will see this in the second part of UEFI and SliTaz.

Insert your Live in a USB port and restart the computer.

If you didn't modify the *UEFI Boot Order*, access the Multiboot menu.

If you don't know how to access the Multiboot menu, have a look here[187]).

Tip: Want your 32bit Live to boot as well as on 32bit computers than 64bit computers?

Copy `bootia32.efi` **and** `bootx64.efi` in the `/efi/boot` directory of the Live!

Cherry on the cake!

Plug your Live in a non EFI computer and... yes it boots too! Well, at least on a Fujitsu E Series...

And on very old 32bit computers — *like Pentium 4* — if you've used **`tazusb`**, your Live will boot with **`Syslinux`** and though the stick is FAT32, it will be mounted at boot and `/home/tux` will be created giving you persistence. :)

Want to enjoy the memory beyond 4GB?

Create a LiveUSB with Rolling core64[188].

I never succeeded to boot it in graphics mode on the two UEFI computers I tested probably because some drivers are missing in Rolling. But alanyih did it[189]. :)

You are now ready for a frugal install. Click *HOWTO do a frugal installation on a computer in UEFI boot mode* (page 226)!

2.5.10 HOWTO do a frugal installation on a computer in UEFI boot mode

author ceel, linea

2.5.10.1 Introduction

Arun Prakash Jana has already described[190] how to install SliTaz in frugal mode on an UEFI computer. But he did install the kernel and the rootfs in the ESP partition.

[187] https://doc.ubuntu-fr.org/tutoriel/modifier_ordre_amorcage_du_bios#liste_des_touches_pour_acceder_au_bios_et_au_boot_menu
[188] http://mirror1.slitaz.org/iso/rolling/slitaz-rolling-core64.iso
[189] http://forum.slitaz.org/topic/slitaz-uefi#post-46187
[190] http://tuxdiary.com/2014/04/13/boot-slitaz-in-uefi-mode/

IMHO, the ESP partition should be reserved for boot loaders and no operating system or software should be installed in it. This document explains how to do a frugal install of SliTaz Next (32bit) on the Windows partition, C:

Note: All along this tutorial,

- the ESP partition will be noted **(ESP)**
- and the main partition of Windows **(C:)**.

2.5.10.2 Installing files on (C:)

1. Start Windows and open the file manager.
2. Select (C:) and create a new directory at root; say `slitaz`.
3. Open `slitaz` and create a directory, say `next` in it.
4. Plug your EFI LiveUSB and copy the files `bzImage` and `rootfs.gz` from the `\boot` directory of the Live into the `\slitaz\next` directory of (C:).
5. Copy the boot loader — `bootia32.efi` for a 32bit EFI system (or `bootx64.efi` for a 64bit EFI system) — from the `\efi\boot` directory of the Live into the `\slitaz\next` directory of (C:).

Note: If you didn't copy files from your EFI LiveUSB but extracted them from an ISO file, pay attention that Windows may truncate files names or/and extensions and will use uppercase.

It doesn't matter that the characters are uppercase or lowercase in NTFS or FAT partitions. But it'll be more consistent if you rename files with their exact name and extensions and respecting the case.

2.5.10.3 Installing the boot loader

The boot loader must be installed in (ESP); this partition is not mounted at boot and is hidden.

Mount (ESP): from Windows 8/10, right-click on the *Home button* in the bottom left corner of the screen. In the context menu that opens, select *Windows PowerShell (admin)*. The *User Account Control* will warn you that Windows PowerShell wants to modify the system; confirm by clicking *YES*.

Note: Note that Windows

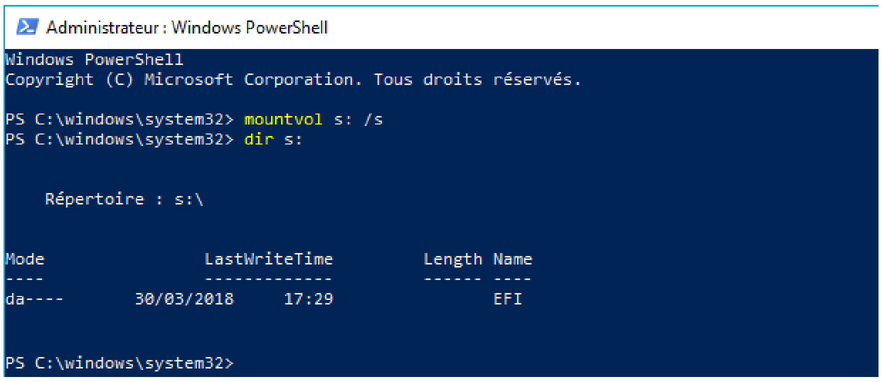
- uses `\` and `/` where Linux respectively uses `/` and `-`
- and doesn't do any differences between uppercase and lowercase.

In the admin terminal, type:

```
mountvol s: /s
```

You can check (ESP) is correctly mounted listing the directories and files it contains:

```
dir s:
```


Administrateur : Windows PowerShell

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Tous droits réservés.

PS C:\windows\system32> mountvol s: /s
PS C:\windows\system32> dir s:


    Répertoire : s:\


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
da----       30/03/2018     17:29                EFI


PS C:\windows\system32>
```

Create a directory, say `slitaz`, in the EFI directory of (ESP)

```
mkdir s:\efi\slitaz
```

Copy the boot loader into it.

- For a 32bit EFI computer, type:

  ```
  move \slitaz\next\bootia32.efi s:\efi\slitaz\grubia32.efi
  ```

- For a 64bit EFI computer, type:

  ```
  move \slitaz\next\bootx64.efi s:\efi\slitaz\grubx64.efi
  ```

Note we have renamed the file; this is not a necessity but it'll help to you remind that SliTaz is launched with GRUB 2.

2.5.10.4 Configuration file for GRUB 2

Create a `\boot\grub` directory in (ESP)

```
mkdir s:\boot\grub
```

Create the configuration file:

```
notepad s:\boot\grub\grub.cfg
```

Example:

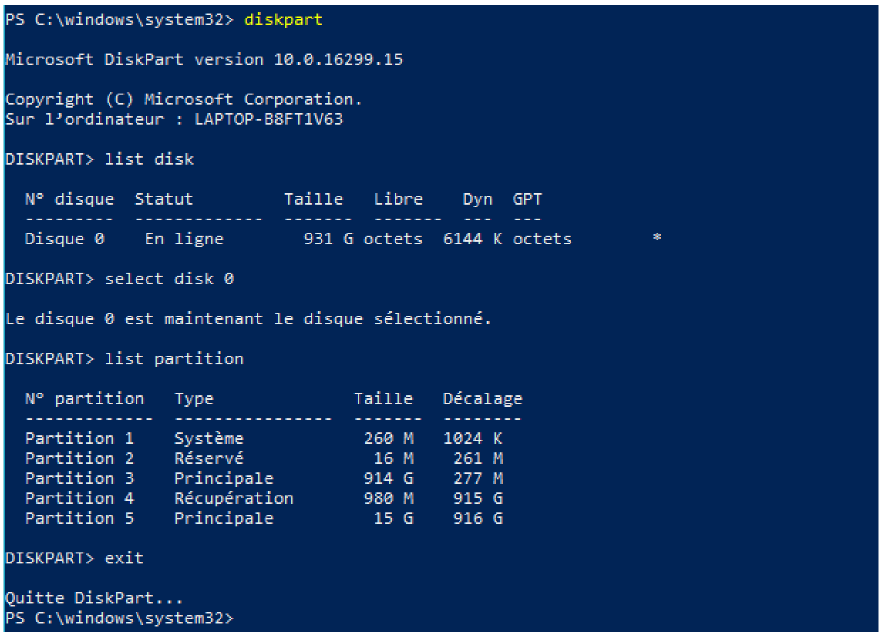
```
### CONFIGURATION FILE FOR GRUB 2 ###
#

# Comment the line if you want SliTaz to start immediately or
# change the value to define the time (seconds) to wait before booting.
set TIMEOUT=-1

# Menu 0: SliTaz Next (ISO 20170930 - kernel 4.9.30)
menuentry "SliTaz Next - frugal install" {
   set root=(hd0,3)
   linux /slitaz/next/bzImage ro root=/dev/null video=-32 autologin
```

(continues on next page)

(continued from previous page)

```
    initrd /slitaz/next/rootfs.gz
}
```

This config file considers (C:) is `(hd0,3)`: the third partition of the first hard disk, this is generally the case. To verify what the main partition of Windows is, run the **`Diskpart`** utility; ask for the `list` of the disks, `select` the Windows disk and `list` its partitions:



```
PS C:\windows\system32> diskpart

Microsoft DiskPart version 10.0.16299.15

Copyright (C) Microsoft Corporation.
Sur l'ordinateur : LAPTOP-B8FT1V63

DISKPART> list disk

  N° disque  Statut         Taille   Libre    Dyn  GPT
  ---------  -------------  -------  -------  ---  ---
  Disque 0    En ligne        931 G octets  6144 K octets        *

DISKPART> select disk 0

Le disque 0 est maintenant le disque sélectionné.

DISKPART> list partition

  N° partition   Type              Taille   Décalage
  -------------  ----------------  -------  --------
  Partition 1    Système            260 M   1024 K
  Partition 2    Réservé             16 M    261 M
  Partition 3    Principale         914 G    277 M
  Partition 4    Récupération       980 M    915 G
  Partition 5    Principale          15 G    916 G

DISKPART> exit

Quitte DiskPart...
PS C:\windows\system32>
```

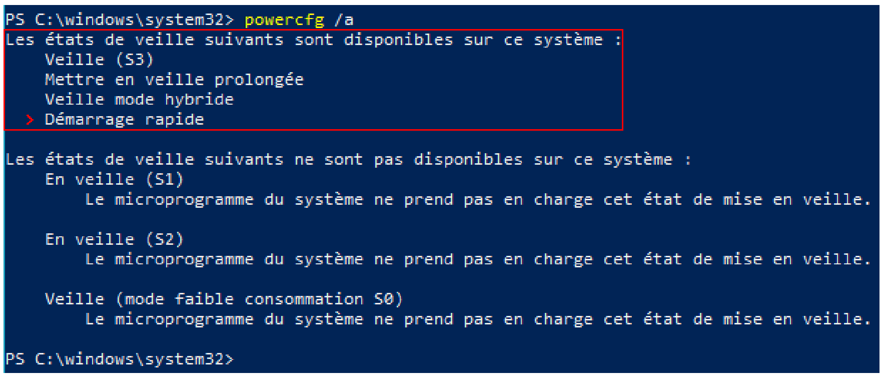
Unmount (ESP):

```
mountvol s: /d
```

2.5.10.5 Disabling the fast startup

Warning: Remember: you MUSN'T access (C:) as long as you haven't disabled the fast startup of Windows!

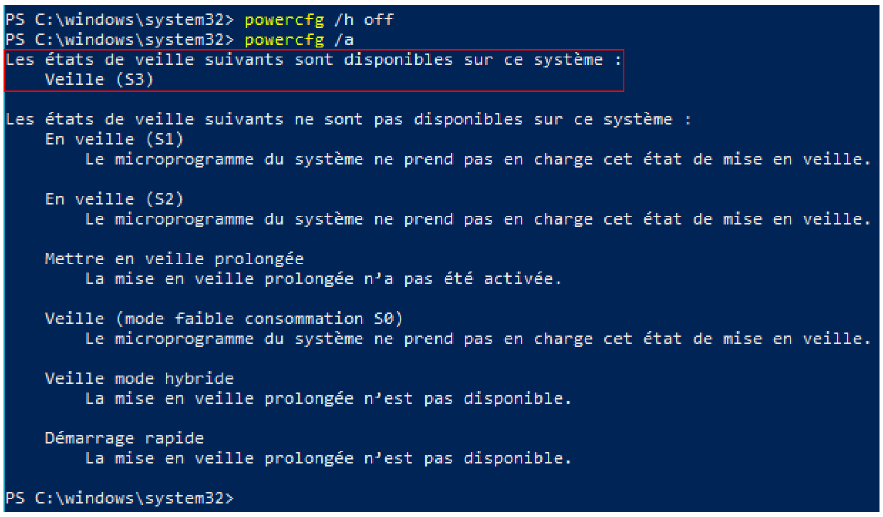
Check if the fast startup is enabled (default); in the admin terminal, type **`powercfg /a`**:



```
PS C:\windows\system32> powercfg /a
Les états de veille suivants sont disponibles sur ce système :
    Veille (S3)
    Mettre en veille prolongée
    Veille mode hybride
  > Démarrage rapide

Les états de veille suivants ne sont pas disponibles sur ce système :
    En veille (S1)
        Le microprogramme du système ne prend pas en charge cet état de mise en veille.

    En veille (S2)
        Le microprogramme du système ne prend pas en charge cet état de mise en veille.

    Veille (mode faible consommation S0)
        Le microprogramme du système ne prend pas en charge cet état de mise en veille.

PS C:\windows\system32>
```

Fig. 3: The fast startup is **enabled**.
(Sorry for french screens; feel free to replace them with english version. Thanks.)

To disable the fast startup, type **`powercfg /h off`**. Check the fast startup is really disabled by typing **`powercfg /a`** again:



```
PS C:\windows\system32> powercfg /h off
PS C:\windows\system32> powercfg /a
Les états de veille suivants sont disponibles sur ce système :
    Veille (S3)

Les états de veille suivants ne sont pas disponibles sur ce système :
    En veille (S1)
        Le microprogramme du système ne prend pas en charge cet état de mise en veille.

    En veille (S2)
        Le microprogramme du système ne prend pas en charge cet état de mise en veille.

    Mettre en veille prolongée
        La mise en veille prolongée n'a pas été activée.

    Veille (mode faible consommation S0)
        Le microprogramme du système ne prend pas en charge cet état de mise en veille.

    Veille mode hybride
        La mise en veille prolongée n'est pas disponible.

    Démarrage rapide
        La mise en veille prolongée n'est pas disponible.

PS C:\windows\system32>
```

Fig. 4: The fast startup is **disabled**.

You didn't think it was possible but now yes, Windows will boot even more slowly than before…

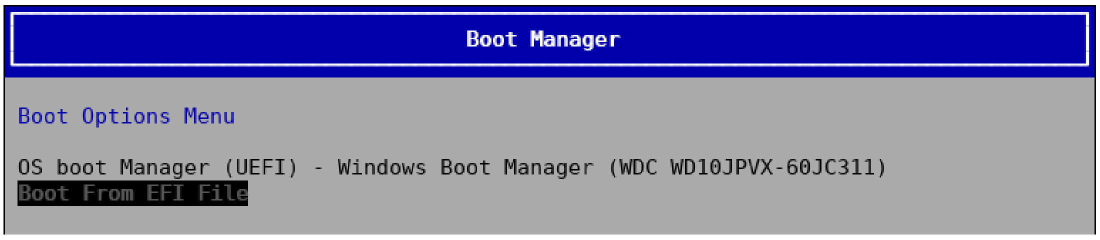
2.5.10.6 Configuring the computer (Setup)

You don't have anything else to do than to disable the *Secure Boot*. You've already did it when you created your EFI LIveUSB.

If you came directly to this tutorial without doing an EFI LiveUSB, see step *Configuring the computer* (page 224) of *HOWTO do a SliTaz LiveUSB for EFI computers* (page 220).

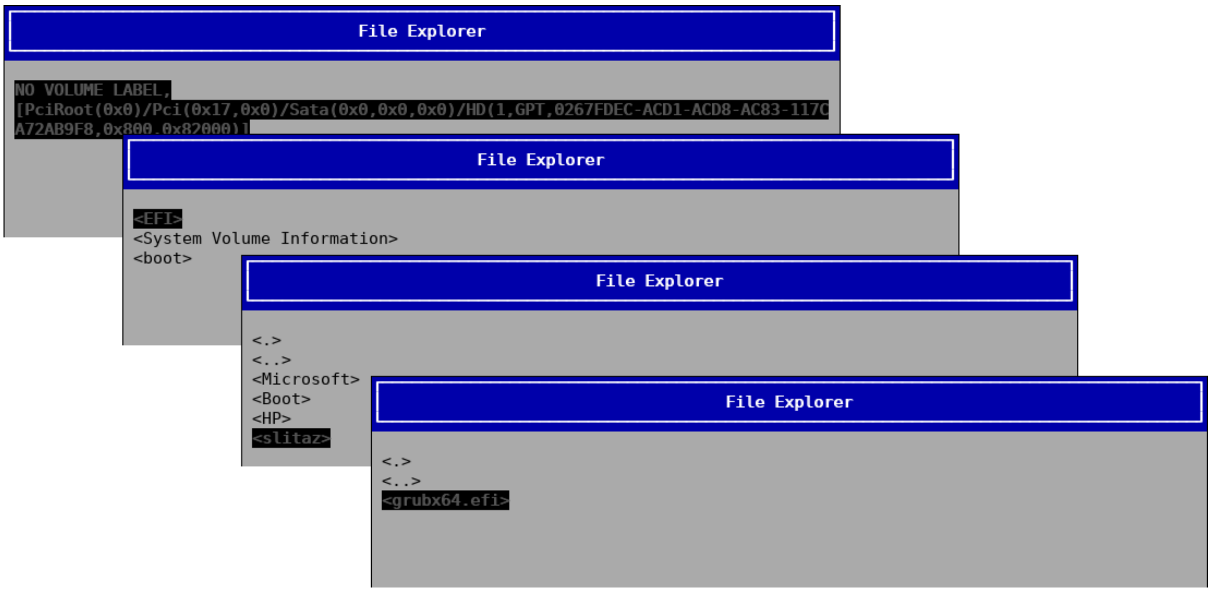
2.5.10.7 Run SliTaz

Reboot the computer and access the Multiboot mode. On the **hp** Laptop 17-bs032, press the `F9` key immediately after you've powered on the computer until the *Boot Manager* displays the list of the disks where it found a `bootx64.efi` file in a `\EFI\BOOT` of a FAT32 partition:



```
Boot Manager

Boot Options Menu

OS boot Manager (UEFI) - Windows Boot Manager (WDC WD10JPVX-60JC311)
Boot From EFI File
```

Select *Boot From EFI File*. The *EFI File Explorer* displays the list of the volumes found; select your hard disk then *EFI* → *slitaz* → *grubx64.efi* to run GRUB 2.



```
File Explorer

NO VOLUME LABEL,
[PciRoot(0x0)/Pci(0x17,0x0)/Sata(0x0,0x0,0x0)/HD(1,GPT,0267FDEC-ACD1-ACD8-AC83-117C
A72AB9F8,0x800,0x82000)]
```

```
File Explorer

<EFI>
<System Volume Information>
<boot>
```

```
File Explorer

<.>
<..>
<Microsoft>
<Boot>
<HP>
<slitaz>
```

```
File Explorer

<.>
<..>
<grubx64.efi>
```

Note: If you don't know how to access the Multiboot menu, have a look here[191]).

Congratulations and Welcome to SliTaz Next!

Tip: Want to test the latest ISOs of SliTaz without having to do changes to (ESP)?

Add a generic entry in your `grub.cfg`; ie:

```
# Menu 1: boot Slitaz from ISO file
menuentry "SliTaz - LiveISO" {
   loopback taziso (hd0,3)/slitaz/slitaz.iso
   linux (taziso)/boot/bzImage ro root=/dev/null video=-32 autologin
   initrd (taziso)/boot/rootfs.gz
}
```

Download the ISO to test in `c:\slitaz`, renaming it `slitaz.iso`.

We now know enough to proceed to a full installation of SliTaz in dual boot with Windows 8/10.

[191] https://doc.ubuntu-fr.org/tutoriel/modifier_ordre_amorcage_du_bios#liste_des_touches_pour_acceder_au_bios_et_au_boot_menu

2.5.11 Updating the processor microcode

author ceel, linea

Important: Updating the microcode of your processor can help to preserve your system against Spectre, a vulnerability that affects most of the modern processors.

For more information about Spectre (and Meltdown), read Important info about Meltdown and Spectre[192].

In the normal way, the microcode is loaded on boot from the BIOS. However it is possible to update it on Linux; this must be repeated on every boot. This document decribes how to proceed for Intel processors.

2.5.11.1 Install the `intel-ucode` package

First of all, update your system:

```
# tazpkg up
```

Then get-install the **`intel-ucode`** package:

```
# tazpkg -gi intel-ucode
```

The package manager installs microcodes for Intel processors in `/lib/firmware/intel-ucode`; files are stored with names in the format `XX-YY-ZZ`.

2.5.11.2 Which microcode for your processor

The command

```
# head -n7 /proc/cpuinfo
```

should return something like this:

```
# head -n7 /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 60
model name      : Intel(R) Core(TM) i3-4000M CPU @ 2.40GHz
stepping        : 3
microcode       : 0x1c
```

Convert the cpu `family-model-stepping` numbers in 3 pairs of hexadecimal values.

From the example above,

- cpu family= 6 → 06,
- model= 60 → 3c and
- stepping=3 → 03.

[192] http://forum.slitaz.org/topic/-important-info-about-meltdown-and-spectre-

The microcode for the processor is `06-3c-03`.

If you don't find your `XX-YY-ZZ` file in `/lib/firmware/intel-ucode`, this means there is no microcode available for your processor; if there is one, force its loading:

```
# echo 1 > /sys/devices/system/cpu/microcode/reload
```

Check if the revision of the microcode has changed:

```
# dmesg | grep microcode
```

```
microcode: CPU0 sig=0x306c3, pf=0x10, revision=0x1c
microcode: CPU0 sig=0x306c3, pf=0x10, revision=0x1c
microcode: CPU0 updated to revision 0x24, date = 2018-01-21
microcode: CPU1 sig=0x306c3, pf=0x10, revision=0x1c
microcode: CPU1 sig=0x306c3, pf=0x10, revision=0x1c
microcode: CPU1 updated to revision 0x24, date = 2018-01-21
microcode: CPU2 sig=0x306c3, pf=0x10, revision=0x1c
microcode: CPU2 sig=0x306c3, pf=0x10, revision=0x1c
microcode: CPU2 updated to revision 0x24, date = 2018-01-21
microcode: CPU3 sig=0x306c3, pf=0x10, revision=0x1c
microcode: CPU3 sig=0x306c3, pf=0x10, revision=0x1c
microcode: CPU3 updated to revision 0x24, date = 2018-01-21
microcode: Microcode Update Driver: v2.00 <tigran@aivazian.fsnet.co.uk>,␣
↪Peter Oruba
```

If the revision hasn't changed, then there is no update for your processor; if it did change, then continue with this procedure.

2.5.11.3 Loading the microcode at boot

The best thing consists of loading the microcode as soon as possible at boot, before the userspace has started. This is done by adding an initrd in the GRUB configuration file.

Create the appropriate environment (no matter where you do it):

```
# mkdir -p initrd/kernel/x86/microcode
```

Change in initrd:

```
# cd initrd
```

Copy your microcode (replace `XX-YY-ZZ` with your microcode file name):

```
# cp -v /lib/firmware/intel-ucode/XX-YY-ZZ kernel/x86/microcode/
↪GenuineIntel.bin
```

Create the initrd:

```
# find . | cpio -o -H newc > /boot/microcode.img
```

Add the following line in your `/boot/grub/menu.lst` (or `grub.cfg`):

```
initrd /boot/microcode.img
```

If you already use an initrd to boot your system, you must load the microcode first; below an example to start SliTaz Rolling in frugal mode:

```
#title SliTaz Rolling core64 - frugal install (kernel 3.16.55)
   root (hd0,2)
   kernel /slitaz/boot/vmlinuz-3.16.55-slitaz64 root=/dev/null autologin
  initrd /slitaz/boot/microcode.img /slitaz/boot/rootfs.gz
```

Reboot the computer and check the microcode has been loaded:

```
$ head -n7 /proc/cpuinfo
processor        : 0
vendor_id        : GenuineIntel
cpu family       : 6
model            : 60
model name       : Intel(R) Core(TM) i3-4000M CPU @ 2.40GHz
stepping         : 3
microcode        : 0x24
```

If you are using a 32bit kernel, it is possible that the CPUs aren't all updated; this is not a problem because the kernel doesn't use them. Below, an example with Rolling Core:

```
$ cat /var/log/messages | grep -i microcode
Mar 27 18:05:01 (none) user.warn kernel: [<c1020ea2>] ? save_microcode.
 ↪constprop.5+0x55/0xba
Mar 27 18:05:01 (none) user.warn kernel: [<c1748ae5>] ? save_microcode_in_
 ↪initrd_intel+0x25/0x50
Mar 27 18:05:01 (none) user.warn kernel: [<c1748920>] ? save_microcode_in_
 ↪initrd+0x1d/0x30
Mar 27 18:05:01 (none) user.err kernel: Cannot save microcode patches from␣
 ↪initrd.
Mar 27 18:05:01 (none) user.info kernel: microcode: CPU0 sig=0x306c3,␣
 ↪pf=0x10, revision=0x24
Mar 27 18:05:01 (none) user.info kernel: microcode: CPU1 sig=0x306c3,␣
 ↪pf=0x10, revision=0x24
Mar 27 18:05:01 (none) user.info kernel: microcode: CPU2 sig=0x306c3,␣
 ↪pf=0x10, revision=0x1c
Mar 27 18:05:01 (none) user.info kernel: microcode: CPU3 sig=0x306c3,␣
 ↪pf=0x10, revision=0x1c
Mar 27 18:05:01 (none) user.info kernel: microcode: Microcode Update␣
 ↪Driver: v2.00
```

The same computer with Rolling Core64:

```
$ cat /var/log/messages | grep -i microcode
Mar 27 18:05:01 (none) user.warn kernel: [<c1020ea2>] ? save_microcode.
 ↪constprop.5+0x55/0xba
Mar 27 18:05:01 (none) user.warn kernel: [<c1748ae5>] ? save_microcode_in_
 ↪initrd_intel+0x25/0x50
Mar 27 18:05:01 (none) user.warn kernel: [<c1748920>] ? save_microcode_in_
 ↪initrd+0x1d/0x30
Mar 27 18:05:01 (none) user.err kernel: Cannot save microcode patches from␣
 ↪initrd.
Mar 27 18:05:01 (none) user.info kernel: microcode: CPU0 sig=0x306c3,␣
 ↪pf=0x10, revision=0x24
Mar 27 18:05:01 (none) user.info kernel: microcode: CPU1 sig=0x306c3,␣
 ↪pf=0x10, revision=0x24
```

(continues on next page)

(continued from previous page)

```
Mar 27 18:05:01 (none) user.info kernel: microcode: CPU2 sig=0x306c3,␣
↪pf=0x10, revision=0x24
Mar 27 18:05:01 (none) user.info kernel: microcode: CPU3 sig=0x306c3,␣
↪pf=0x10, revision=0x24
Mar 27 18:05:01 (none) user.info kernel: microcode: Microcode Update␣
↪Driver: v2.00
```

2.5.11.4 Update the processor microcode regularly

Intel provide frequent updates of their microcodes. You can download the latest version of the Intel microcodes at https://downloadcenter.intel.com/download/27591/Linux-Processor-Microcode-Data-File.

As the root user, uncompress the downloaded `microcode-`*`YYYYMMDD`*`.tgz` file in `/lib/firmware`.

```
# tar -xzf microcode-YYYYMMDD.tgz /lib/firmware/
```

Then follow this HOWTO from *Which microcode for your processor* (page 232).

2.6 Server setup and system Administration

2.6.1 Dropbear SSH

author kultex, linea

`Dropbear` is a small SSH server and client written by Matt Johnston.

https://matt.ucc.asn.au/dropbear/dropbear.html

By default SliTaz will not start the SSH server at boot. To be launched automatically, `dropbear` must be added to the variable `RUN_DAEMONS` in the `/etc/rcS.conf` file. (Normally **`Dropbear`** logs in `/var/log/messages`, but in SliTaz it logs in `/var/log/auth.log`)

More info: *Secure SHell (SSH)* (page 366)

A good doku of dropbear is on OpenWRT — http://wiki.openwrt.org/doc/uci/dropbear

2.6.2 The SQLite database

author emgi, linea

`SQLite` is a lightweight SQL database and as such perfectly suitable for use with Slitaz. Lightweight does however not imply that its functionality is also limited; rather the opposite!

To begin with, the program in all its aspects is extensively documented on: http://www.sqlite.org/. This leaves fairly little left to be documented elsewhere. If you have never worked with SQL before, this guide will provide sufficient information to get you started and to determine whether SQLite is the proper choice for your database.

Contrary to virtually all other SQL database solutions, you don't need a server to run SQlite. The database consists of a single file which is accessed via the **`sqlite`** command line tool or through a programming language like Perl or Python. Apart from the inherent simplicity, the big advantage of this approach is that it is VERY fast, even while running on low-end hardware. Backups are easy too; you just save the file holding the database to another location. These properties make SQLite probably the best choice for single user and embedded solutions. Small-scale CGI is another application which works very well with SQLite. All of this and more is described on: http://www.sqlite.org/whentouse.html

2.6.2.1 What do I need to get started with SQLite?

To begin with SQLite on Slitaz, you only need to install the sqlite client package:

```
# tazpkg -gi sqlite
```

This installs the sqlite client in your `/usr/bin` directory. The version offered with slitaz is a stable release but not the latest one available. SQLite is under active development and new versions appear on a regular basis. If you need a more recent version you may use the precompiled binary which is available on the SQLite.org download page. To do this, you can simply download the precompiled binary and save it in the `/usr/bin/` folder, overwriting the file from the tazpkg installation if it exists.

Important: The version of the SQLite client is only backwards compatible with the database file. You cannot access a database which was created using, for example, version 3.7.16.2 with an older release of the client like 3.7.9. The opposite — a newer version of the client accessing a database which was created with a previous version though is generally possible. This can be relevant if you need to access a database from different client systems. Best practice is to use the same version of the client everywhere.

To run SQLite with Perl, one additional package must be installed; the perl-sqlite module:

```
# tazpkg -gi perl-dbd-sqlite
```

Using SQLite with Perl, Python or any other language is documented extensively on various websites and describing it here would be out of scope as far as this document is concerned. The saying “Google is your friend” really applies in this case.

The sqlite client `/usr/bin/sqlite3` allows you to do anything from the command line which also can be done using Perl, Python or any other language. You can print, update, add or delete records and manipulate columns and tables. Starting the program without any parameters or options returns a > prompt. You may enter sql commands terminated by the semicolon (`;`) as is normal in all sql dialects. Queries must contain a reference to the database file and the sql part must be between double quotes and terminated by a semicolon (`;`). One small example:

```
$ sqlite3 /home/tux/yourdatabase "select * from sqlite_master;"
```

This command prints the layout of the tables in the database. To create a new database, set the filename to whatever is deemed suitable. A new database is created if the file does not exist.

Please refer to http://www.sqlite.org/lang.html for the commands to create tables and add columns to them. Its all pretty straightforward and virtually the same as with most other SQL versions. If you are completely new to SQL, you will have something of a learning curve but many things picked up here can be used with other SQL versions too. It’s certainly worth it!

Copying or renaming the file containing the database is also trivial. To make a copy simply type:

```
$ cp /home/tux/yourdatabase /home/tux/mydatabase
```

The filesize depends heavily on the type of database and its actual contents. The more data is entered, the larger the file becomes. Please note that the file does not shrink when a large amount of data is deleted. Instead, the empty space is preserved and re-used before the file starts to increase in size again. Something else to keep in mind is that the file permissions must be set correctly to be able to read or write the database.

2.6.2.2 Keeping sqlite up-to-date

SQLite is still under development and new versions are made available every one-two months. You can run the latest client without compiling and installing a package every time. In the end, it is just one binary file: `sqlite3`. This file is available as a precompiled binary for download from www.sqlite.org. All that's needed is to move it to the correct folder, thereby replacing the old version. A sample script to automate this action could look like this:

```
cd /home/<your-home-dir>
if [ -s sqlite3 ]; then
 rm sqlite3.old
 mv /usr/bin/sqlite3 sqlite3.old
 mv sqlite3 /usr/bin/
 chmod 755 /usr/bin/sqlite3
fi
```

This script looks for a file named `sqlite3` in `/home/`*`your-home-dir`*, if found it moves the old `sqlite3` binary to `sqlite3.old` in `/home/`*`your-home-dir`* and installs the latest version with the correct access mode in `/usr/bin`.

2.7 Kernel and SliTaz Build system

2.7.1 Build your own custom Linux Kernel for SliTaz

author linea, fandesandro, jozee, seawolf, ernia, woodt

2.7.1.1 Prepare your system

- Install the **`slitaz-toolchain`** meta package. This package contains the required base files.
  - **`binutils`**
  - **`linux-headers`**
  - **`glibc-dev`**
  - **`gcc`**
  - **`make`**

  ```
  # tazpkg get-install slitaz-toolchain
  ```

- Install the packages required to configure and compile the kernel sources.
  - **`ncurses-dev`**
  - **`perl`**

```
# tazpkg get-install ncurses-dev
# tazpkg get-install perl
```

- SliTaz provides a **`linux-source`** package. The kernel source tree will be downloaded from the mirror, installed to the `/usr/src/linux-`*`VERSION`* directory, and patched for SliTaz.

```
# tazpkg get-install linux-source
# /usr/bin/get-linux-source
# ls -l /usr/src
lrwxrwxrwx  1 root root       21 Jul 21 21:27 linux -> linux-2.6.25.5-
↪slitaz
drwxrwxr-x 23 root root     4096 Jul 21 22:41 linux-2.6.25.5-slitaz
-rw-r--r--  1 root root 48589640 Jul 21 21:28 linux-2.6.25.5.tar.bz2
```

2.7.1.2 Configure and Compile

The Linux kernel source tree is now ready to configure and compile.

- Go to the directory that contains the kernel source tree.

```
# cd /usr/src/linux
```

- Prepare the build from the default configuration.

```
# make oldconfig && make prepare
```

- Modify the kernel configuration to your needs, and compile.
- The *menuconfig* allows you to customise the kernel, which is built into the `bzImage` file. Any *modules* are built and then *install*ed in to the `/usr/include` directory. You should then copy the kernel itself to the `/boot` directory.

```
# make menuconfig
# make
# make modules
# make modules_install
# cp arch/x86/boot/bzImage /boot
```

- Configure the boot-loader (optional)

```
# leafpad /boot/grub/menu.lst
```

- Add the new kernel to the list

```
# My kernel:
title       SliTaz GNU/Linux (cooking) (Kernel <VERSION>)
            root (hd0,1)
            kernel /boot/bzImage root=/dev/sda2
```

2.7.2 Chroot

author gokhlayeh, linea, jozee

This guide explain how to setup a *chroot* to *cook* packages in a separate environment. The *chroot* can be built on a USB or HDD device to save RAM when using a live session. There is also a script which removes any packages installed during *cooking* on exit in order to keep the *chroot* light and also checks the `build_depends`.

2.7.2.1 With `Tazdev`

Over time the SliTaz developers created tools for the automation of various tasks. The **`slitaz-dev-tools`** package provides the **`tazdev`** utility and its configuration file `/etc/slitaz/tazdev.conf`; it can create a *chroot* to use:

```
# tazpkg get-install slitaz-dev-tools
# tazdev gen-chroot
# tazdev chroot
```

By default the *chroot* is created in `/home/slitaz/cooking/chroot` and is slitaz-based. For more info and available commands you can use **`tazdev usage`** and/or take a look at the configuration file.

2.7.2.2 Create the chroot

```
#!/bin/sh
mkdir /home/chroot
# You can mount a device to /home/chroot
# with mountbox or mount
tazpkg get-install busybox --root="/home/chroot"
echo "No" | tazpkg get-install bash --root="/home/chroot"
tazpkg get-install slitaz-toolchain --root="/home/chroot"
tazpkg get-install tazwok --root="/home/chroot"
tazpkg get-install tazpkg --root="/home/chroot"
tazpkg get-install lzma --root="/home/chroot"
mkdir /home/chroot/home/slitaz
```

Note, you can keep these lines in a script file if needed. Just add `#!/bin/sh` to the first line and make executable with:

```
# chmod +x script_file
```

2.7.2.3 Add a script file to automate some actions

```
# cat > /home/chroot/chroot_script.sh << "EOF"
```

```
#!/bin/sh
/bin/sh --login
for pkg in $(cat /var/log/tazpkg.log | grep -v Removed | sed 's/\(.*\)\(-␣
↪Installed - \)\(.*\)\( (.*\)/\3/'); do
    echo "y" | tazpkg remove $pkg
done
rm /var/log/tazpkg.log
```

```
EOF

# chmod +x "/home/chroot/chroot_script.sh"
```

Note, **/bin/sh --login** logs you into the *chrooted* environment. The commands after that auto-remove any packages added when *cooking* on exit. You can hack this file to execute various automated actions when entering and exiting the *chroot*.

2.7.2.4 Add a script to mount and umount *chroot*

```
# cat > /usr/bin/tazchroot << "EOF"
```

```
#!/bin/sh
cat /etc/resolv.conf > /home/chroot/etc/resolv.conf
if [ ! -d "/home/chroot/proc/1" ]; then
     echo "Mounting virtual filesystems..."
     mount -t proc   proc   /home/chroot/proc
     mount -t sysfs  sysfs  /home/chroot/sys
     mount -t devpts devpts /home/chroot/dev/pts
     mount -t tmpfs  shm    /home/chroot/dev/shm
     mount /home/slitaz /home/chroot/home/slitaz
     chroot /home/chroot ./chroot_script.sh
     until [ "$ps" = "2" ]; do
             echo "Waiting for the end of all other chroot processes..."
             ps=$(ps | grep `basename $0` | grep -v grep | grep -v␣
↪basename | wc -l)
             sleep 1
     done
     umount /home/chroot/home/slitaz
     umount /home/chroot/dev/shm
     umount /home/chroot/dev/pts
     umount /home/chroot/sys
     umount /home/chroot/proc
else
     echo "The chroot is already mounted"
fi
```

```
EOF

chmod +x /usr/bin/tazchroot
```

Note, this script mounts `/home/slitaz` in your *chroot*, so you can use **tazwok** as if it was in your normal environment.

2.7.3 Tazwok Tips

author jozee, linea, brianperry

You must read the *Cookbook* (page 369) first — it's a great collection of documents. This guide supplements the cookbook with some tips that you are most likely to face while compiling packages for SliTaz. Note that it should be mentioned that Tazwok has been deprecated since SliTaz v4 (switched in favor for Tazlito, the ISO hacking method and *Toolchain* (page 393)).

2.7.3.1 Tips

1. Add the **slitaz-dev-pkgs** package — it's a meta package that adds some of the useful ***-dev** packages
2. Compilation Errors like
   - checking for $PKG... configure: error: Couldn't find $PKG >= 1.0.0,
   - configure: error: Couldn't find $FILENAME
   - use: **tazpkg search-pkgname $FILENAME** to find the package with the missing file; Add this package to the `BUILD_DEPENDS` in the receipt
3. Read receipts of the other packages for standardization. The problems you face may already have a solution in one of these receipts. Some example receipts include:
   - **xorg-libSM** (or any **xorg-***): to use a `$SOURCE` variable
   - **cups**: to strip files other than `.so` in `usr/lib`, and to compress some driver files
   - **ndiswrapper-driver** or **broadcom-wl**: to cook Linux modules
   - **mplayer-svn**: to create packages from a SVN/Git repository
   - **915resolution** or **busybox**: to add patches
   - **nvidia**: to create non-free packages that need the **linux** package for compiling modules
   - **get-***: to create non-free packages
   - ***-dev**: to see how to create a dev receipt
4. Create lighter packages. Always remember to disable GNOME dependencies or any other dependencies that are not so useful. Always use **configure --help** to see the compile options. Many packages have auto-dependency tracking, i.e., if it finds a corresponding ***-dev** package installed on the system, it will auto-enable an option. So, if you don't disable an unwanted dependency, then when a package is cooked on the SliTaz tank server, it is likely that the package won't run properly because of the missing dependency. The missing dependency problems that are often reported on the forum are caused due to this problem.
5. Missing `.pc` files: Sometimes a dev package created upstream misses `.pc` files, so **pkg-config** cannot locate the corresponding dev files correctly. This can play havoc sometimes as you expect the upstream packages to be properly distributed. **firefox-dev** is one such example.
6. gcc4.4 patches: The new stricter format of gcc4.4 breaks many packages. This is quite a common problem. You need to either create a simple patch or see if a new upgrade is available upstream. See **mplayerplug-svn**.
7. Use a chroot environment and keep it clean so that you can submit the receipt with the correct `BUILD_DEPENDS`.

2.7.4 hal-removal for 5.0

author kultex, pankso

just to avoid errors its good to plan and document this process

Package	Status	Who	Link / Discussion / Bugreport
udev	open		udev rules maybe should be in `/lib/udev/rules.d` — because applications drop it there https://wiki.archlinux.org/index.php/Udev udev rules written by the administrator go in `/etc/udev/rules.d/`, their file name has to end with `.rules`. The udev rules shipped with various packages are found in `/lib/udev/rules.d/`. If there are two files by the same name under `/lib` and `/etc`, the ones in `/etc` take precedence. There is an error in the arch doku — because all rules are in `/lib` — `/usr/lib/udev` does not exits — corrected here but maybe not a must, because opensuse 12.2 still has the udev rules still in `/etc` http://doc.opensuse.org/documentation/html/openSUSE/opensuse-reference/cha.udev.html also useful — the gentoo doku (nvidia troubles) http://www.gentoo.org/doc/en/udev-guide.xml – Pankso note – Now udev (from version 182) is shipped in systemd source tree… so we will have a systemd package but split udev into a light package and by default last udev install all rules in /usr/lib/udev…
systemd	open		should replace init.d http://en.wikipedia.org/wiki/Systemd — NO (at least not now) — Systemd will replace all our boot scripts and process! First our boot scripts are realy special, even not sysV and easily modifiable since ther pure shell scripts. Second systemd is a huge package for our base system (more than 5 Mb). Third, if we switch we must heavely customize it and have `*.service` file for each daemon (slim, lighttpd, etc). Last, systemd aims is to speed up boot time, but we alredy boot faster than sysV. If one want to see the amount of work before using systemd (even as replacement, not default), you can install it from undigest repo and add an entry to grub menu.list with `init=/bin/systemd` — Pankso
libpng	open		up to libpng 1.5.x
dashel	open		http://svn.gna.org/viewcvs/dashel/trunk/readme.txt?view=markup&pathrev=144 should not be a problem
gnomad2	open		http://gnomad2.sourceforge.net/ Gnomad 2.9.6 is released — we delete HAL support and slam in libgudev support in its place, HAL is deprecated now
gtkpod	open		should not be a problem, when gtkpod finds no hal depencies
gxine	open		should be no problem http://archives.gentoo.org/gentoo-dev/msg_1761183caa2cae78c9623485142a372b.xml

Continued on next page

Table 2 – continued from previous page

Package	Status	Who	Link / Discussion / Bugreport
hal-cups-utils	open		must be replaced by udev-config-printer http://cyberelk.net/tim/2009/07/20/re-writing-hal-cups-utils-to-avoid-hal/
hal-dev	open		just to delete
libexo-dev	open		maybe update to 0.6.0 -1, but perhaps just to delete the depency in libexo-dev, because all other exo packages have no depency on hal https://launchpad.net/ubuntu/+source/exo/0.6.0-1
files hal-extra	open		just to delete
hal-info	open		just to delete
hal-scripts	open		just to delete
ivman	open		just to delete
libgphoto2	open		maybe it must be configured with option without hal libgphoto2_port - added –with/–without configure options for: bonjour, hal http://www.gphoto.org/news/
hplip	open		udev rules should be shipped with hplip https://bugs.launchpad.net/hplip/+bug/401091
libgphoto2-dev	open		same as libgphoto2
sane-backends	open		should be no problem - needs the 55-libsane.rules shiped with sane either in lib or in etc
brscan	open		should not be a problem perhaps edit the the udev-rules http://welcome.solutions.brother.com/bsc/public_s/id/linux/en/instruction_scn1c.html#u9.10
brscan2	open		same as brscan
sane-backends-dev	open		same as sane-backends
xsane	open		indirect depency from sane-backends
pcmanfm	done	Christophe[193] Pascal[194]	just to delete / update in flavors to pcmanfm2 pcmanfm from 0.9 series has ported to use gvfs / udisk for volume management
lxde	done	fixed with pcmanfm	indirect depency from pcmanfm — was fixed with the fix of pcmanfm https://bbs.archlinux.org/viewtopic.php?id=123650 relevant only lxsession (0.4.4-2 supports now power management through upower)
pmount	open		should not be a problem — not much infos https://bbs.archlinux.org/viewtopic.php?id=120558
thunar-vfs	open		Released Thunar 1.1.1 and thunar-volman 0.5.0 with full support for udev and GIO and no dependency on HAL http://wiki.xfce.org/dev/thunar-volman-udev http://gezeiten.org/post/2010/01/Thunar-volman-and-the-deprecation-of-HAL
thunar-vfs-dev	open		same as thunar

2.8 SliTaz Documentation Guidelines

author domcox

This document provides guidelines on writing a wiki article and the tasks to be performed to make SliTaz documentation up-to-date.

2.8.1 Summer of Documentation

1. Centralization of all documentation to http://doc.slitaz.org
2. <del>Complete migration of Handbook and Cookbook (Scratchbook, if possible)</del>
3. Translation of Handbook and Cookbook
4. Link or translate the wiki articles on the labs website[195]
5. Review and update Handbook and Cookbook as of 3.0
6. Add new guides. A wishlist of guides has been predefined on the *guides page* (page 79) as a starting reference
7. Review and update existing guides

2.8.2 General Instructions

- **Add**: Feel free to add any pages to the wiki
  - **Namespaces or Documentation Hierarchy**: Documentation Hierarchy structure has been defined for the English language. Please follow it as a standard while creating pages. Some examples:
    - `en:handbook:start`: Handbook start page
    - `en:handbook:desktop`: Desktop is a page link on the handbook start page. All handbook pages should have the namespace "en:handbook:"
    - `en:guides:faq`: All guides pages should have namespaces "en:guides". To create a faq page, simply do `[[faq | FAQ]]`. This will automatically create a faq page with the en:guides:faq namespace
    - `Index`: Links can be used to see the hierarchy structure
  - **Add Images**: Use the toolbar and add images to the relevant namespaces
- **Delete**: Simply remove all the contents of the pages to delete a page
- **Review**: Each page should contain a review section. For example, *see the bottom of this page* (page 246). A review section is just a wiki table. You are free to edit the table and/or add extra rows to the table. You can translate it into your own language. You can also copy and paste the page review table on this page to any of the wiki pages and edit accordingly

[193] http://hg.slitaz.org/wok/rev/473b2d729747
[194] http://hg.slitaz.org/wok/rev/2f1b673c2701
[195] http://labs.slitaz.org

2.8.3 Page Layout

There are some definite styles of pages to which we can bring a consistent layout.

2.8.3.1 FAQ

This applies to one page, *Frequently Asked Questions* (page 95).

1. **Error Message** — title the individual FAQ with the most likely description, usually the error message displayed.
2. **Symptoms** — a brief description of what the user may experience when this FAQ applies. There may be more than one. Use correct formatting when describing on-screen messages, keystrokes etc. Hopefully, Google results will pick this up.
3. **Explanation** — a not-very-technical explanation of the error message. Users will be able to understand the problem and how it can be resolved in a high-level sense.
4. **Solution** — how to solve the problem, technically. Include brief descriptions of the steps needed rather than merely a list of commands; this is important to understand what the user needs to do. Individual problems have slightly different solutions — use levels of bullet points etc. to organise it.

2.8.3.2 Guides

These resides under the `<lang>:guides:<topic>` name-space. They describe a process to get something working.

1. **Introduction** — Summarize the article
2. **Graphical Way**
   - Instructions — How to use the graphical tool (if it exists)
   - Screenshots — A picture is worth a thousand words
3. **Manual Way**
   - Installation — Define the packages required and how to install them
   - Configuration — Explain how to configure files for the proper functioning of packages
   - Summarize — If possible, summarize all commands in one single script for troubleshooting
4. **Examples and Tips** — Add some examples and advanced tips
5. **FAQ/Troubleshooting** — Some DIY instructions or a sub-section on problems/symptoms/solutions/notes or a link to forum posts. Link to FAQ if answered there
6. **References** — Other good reference material on the Internet. If there aren't any, consider a message asking for some!

2.8.3.3 Handbook

These reside under the `<lang>:handbook:<topic>` namespace. They brief the reader on what SliTaz can offer on a particular topic. They are an overview and description and not a guide, though they may contain (very) few steps on how to get up-and-running.

1. **Blurb** — describes the content of the page, in terms of scope.
2. **Topic** — what the user expects to achieve, e.g. 'Image Processing' or 'Desktop Themes'
3. **Body Text** — an overview of the topic, with links to relevent Guides or external web pages.
4. **Tips** (optional) — any problems the user may experience. Link to FAQ if answered, forum posts, good problem-solving web pages etc.

2.8.4 Formatting

Please use correct formatting wherever possible. It aids readability and often lessens ambiguity between commands that should be entered vs. output etc.

- Learn wiki syntax[196]. For testing wiki syntax, just use the Playground[197] page.

[196] http://doc.slitaz.org/wiki:syntax?s{[]}{]]}=playground
[197] http://doc.slitaz.org/en:guides:playground

CHAPTER 3

References

authoe jozee, linea, seawolf, lexeii, mtas, hgt

This category is to provide useful Linux references/articles/bookmarks. Please feel free to add any topic that you think may be useful for the community.

3.1 Development

- Shell scripts (LinuxCommand.org)[198] — Writing shell scripts
- Advanced bash-scripting guide[199] — Writing bash scripts
- Repology[200] — Comparing packages versions across repositories

3.2 Linux References

- One Page Manual[201] — Not everything applies to SliTaz
- Another command line reference[202]
- Linux How-To Articles, Tips, and Guides[203] — How-To Geek

[198] http://linuxcommand.org/writing_shell_scripts.php
[199] http://tldp.org/LDP/abs/html/
[200] https://repology.org/
[201] http://www.digilife.be/quickreferences/QRC/The%20One%20Page%20Linux%20Manual.pdf
[202] http://www.pixelbeat.org/cmdline.html
[203] http://www.howtogeek.com/tag/linux/

CHAPTER 4

Development Notes

author gokhlayeh, linea, godane

This page contains development information about the cooking version. Some pages will go into the guides or books when the next stable version is released. Others are temporary pages which coordinate development work.

Information for users

- *xorg-server-1.8* (page 249) — Update configuration

Information for contributors

- *xorg-server-1.8* (page 250) — Drivers pre-configuration
- *Prepare the experimental wok* (page 252) — Howto use the experimentals patches with MQ
- *ASH benchmarks* (page 254) — How-to optimize ash scripts
- Quick guide to start cooking SliTaz packages[204]

Deprecated Stuff

- *The new tazwok illustrated!* (page 258)

4.1 xorg-server-1.8

author gokhlayeh, linea

[204] https://oldpapyrus.wordpress.com/2013/08/17/quick-guide-to-start-cooking-slitaz-packages/

4.1.1 Upgrade Xorg configuration

Upgrade Slitaz:

```
# echo 'y' | tazpkg upgrade
```

Backup `xorg.conf`:

```
# mv /etc/X11/xorg.conf /etc/X11/xorg.conf-backup
```

Re-configure **`Xorg`** with **`tazx`**:

```
# tazx config-xorg
```

4.1.1.1 User tweaks

If you have tweaked the `xorg.conf` file, then you have two possibilities. In both cases the tweaks will be read before the default configuration and will not be erased by an upgrade:

- Put your modifications (examples) in the file `/etc/X11/xorg.conf`. This file is now reserved for users, it's the first read by **`Xorg`** when X boots.
- You put them in the correct file in the directory `/etc/X11/xorg.conf.d`. SliTaz use two files for each Xorg section:
  - `n0-Section.conf` for the default configuration and
  - `(n-1)5-SectionTweaks.conf` for the configurations which overwrite the default one.

Example:

`40-evdev.conf` contains the default configuration for all devices.

`35-synaptics.conf` is installed with the package **`xorg-input-xf86-synaptics`** and contains the configuration file for touchpad. It overwrites the default one (**`evdev`**).

If you wish to modify this section manually, take care to rename the file with a lower number like `31-MyDevices.conf`. And don't use a file named `x5` or `x0` to ensure that it will not be overwritten by an upgrade.

4.2 xorg-server-1.8

author gokhlayeh, linea

4.2.1 Introduction

With xorg-server-1.8, the file `/etc/X11/xorg.conf` is deprecated. Though we can actually reserve it for user modifications, it's no longer edited automatically.

The configuration files are now in `/etc/X11/xorg.conf.d`. To ensure that the users configuration will not be erased by a system upgrade, here's a suggestion of numbers and content. The configuration files are read in alphanumerical order and their names must end with `.conf`:

- 00 — ServerFlag

- 05 — ServerLayout Tweaks
- 10 — ServerLayout
- 15 — Files Tweaks
- 20 — Files
- 25 — Modules Tweaks
- 30 — Modules
- 35 — InputClass/InputDevice Tweaks
- 40 — InputClass/InputDevice
- 45 — Monitor Tweaks
- 50 — Monitor
- 55 — Device Tweaks
- 60 — Device
- 65 — Screen Tweaks
- 70 — Screen
- 75 — Modes Tweaks
- 80 — Modes
- 85 — DRI Tweaks
- 90 — DRI
- 95 — Extension Tweaks
- 100 — Extension

Some of the x0 sections are configured by **`tazx xorg-config`** using a template `xorg.conf` file (generated with **`Xorg -configure`**). You can find more information about this in the function xorg_conf_d[205] of **`tazx`**.

If you want add a configuration file to a package, please take care to use a name different from those specified by **`tazx`** or this file will be erased when booting in live mode. You can use the same number (x0) with a different name if this configuration doesn't conflict with those used by default. Otherwise, use (x-1)5 for that configuration overwriting the default one.

Examples:

- 40-evedev.conf[206] — Default configuration for all devices
- 35-synaptics.conf[207] — Advanced configuration for touchpad devices, overwriting the default configuration for these devices
- hwsetup, section nvidia[208] and ati[209] — Add a file `55-DeviceTweaks.conf` configuring a graphic driver other than **`vesa`**

[205] http://hg.slitaz.org/slitaz-tools/file/ca6804d9b56b/tinyutils/tazx#l27
[206] http://hg.slitaz.org/wok/file/c29991cef110/xorg-xf86-input-evdev/stuff/40-evdev.conf
[207] http://hg.slitaz.org/wok/file/c29991cef110/xorg-xf86-input-synaptics/stuff/35-synaptics.conf
[208] http://hg.slitaz.org/slitaz-tools/file/ca6804d9b56b/tinyutils/hwsetup#l685
[209] http://hg.slitaz.org/slitaz-tools/file/ca6804d9b56b/tinyutils/hwsetup#l753

4.2.2 Configuration files list

This avoid naming two files identically, you can use these in the `/etc/X11/xorg.conf.d/readme` for users with the 4.0 release.

`tazx`:

- `10-ServerLayout.conf`
- `20-Files.conf`
- `30-Module.conf`
- `40-Keyboard.conf`
- `50-Monitor.conf`
- `60-Device.conf`
- `70-Screen.conf`

`tazhw`:

- `55-DeviceTweaks.conf`

`xorg-xf86-input-evdev`:

- `40-evdev.conf`

`xorg-xf86-input-synaptics`:

- `35-synaptics.conf`

4.2.3 TODO

- Improve the auto-configuration tool in **`tazx`**: some fonts and modules don't need to be loaded by default or don't exist. Some of these elements can be configured by their original package.
- Pre-configure some drivers to make them work out-of-the-box.

4.3 Prepare the experimental wok

author gokhlayeh, linea

4.3.1 Warning

Before using the experimental wok, it's advisable to read and understand the *Advanced usage of Mercurial* (page 381) as the experimental wok uses MQ to handle the patch queue.

Please note that the source code, wok & packages use a lot of space, don't store all of that in your RAM unless you're sure that it will run fine. If you use a live system, you can use a physical disk to store data using the boot option `home=`. You can also mount the disk (i.e.: on `/mnt`).

4.3.2 Enabling Mercurial Queue

The experimental wok is shared as patches which can't be integrated into the cooking wok for now. To use them, you need the extension MQ. So your `~/.hgrc` should contain these lines:

```
[extensions]
mq =
```

4.3.3 Download cooking wok & patch it

First check that the destination disk is mounted. In the directory which contains the data, create a repository called `experimental` and enter in it:

```
cd destination
mkdir experimental
cd experimental
```

Download the wok:

```
hg clone http://hg.slitaz.org/wok wok
```

If you were root, enable read/write permissions to your normal account; then exit the root one:

```
chown tux.tux -R wok
```

The patches will apply well on the revisions for which they was created, but problems can appear with the other revisions. The revisions correlated to the last update of the patches can be found here: http://hg.slitaz.org/wok-experimental. Update the wok to the good revision:

```
hg update -C rev
```

Download experimental patches:

```
cd .hg
hg clone http://hg.slitaz.org/wok-experimental patches
```

Apply patches:

```
cd ..
hg qpush -a
```

4.3.4 Update the experimental wok

Go into the wok:

```
cd ?/wok
```

Unapply patches (doesn't work if modifications are uncommitted):

```
hg qpop-a
```

Update cooking wok:

```
hg pull
```

Check the revision to use at http://hg.slitaz.org/wok-experimental and update the working directory to this revision:

```
hg update -C rev
```

Update patches repository:

```
cd .hg/patches
hg pull -u
```

Apply the new patches:

```
cd ../..
hg qpush -a
```

4.4 ASH benchmarks

author gokhlayeh, linea

Some tips to speed-up scripts… Add yours, and put the better ones on the top of the page :)

4.4.1 Sed substitution vs Variable substitution

Information

In some cases, both tools can do the same job. As a built-in ash/bash command, variable substitution is faster.

Note: This is always true — you can compare each type of variable substitution with an equivalent using an external tool.

Benchmark

```
$ echo '#!/bin/sh
for i in $(seq 1 1000); do
      echo "slitaz" | sed 's/slitaz/SliTaz/'
done' > /tmp/slow
$ echo '#!/bin/sh
for i in $(seq 1 1000); do
      A=slitaz
      echo "${A/slitaz/SliTaz}"
done' > /tmp/speed
$ chmod +x /tmp/slow /tmp/speed
$ time /tmp/slow
> real        0m 12.40s
$ time /tmp/speed
> real        0m 0.04s
```

4.4.2 Group command vs. Sub-process

Information

Group command "{}" groups several commands into one (as a function). So, output can be grouped too: "{ com1; com2; } | com3". Sub-process "()" achieves something similar, but creates a shell sub-process; which costs a lot more resources. Another difference is that when you kill an application using `Ctrl+C`, a sub-process is killed instead of a main application — while `Ctrl+C` on grouped commands kills the application itself. Finally, changing directory or variables into sub-processes will not affect the main script which it does with grouped commands. Conclusion: always use group commands instead of sub-processes, and take care ;D

Note: Group commands need a newline before closing — or "; }".

Benchmark

```
$ echo '#!/bin/sh
for i in $(seq 1 10000); do
      ( echo yo )
done' > /tmp/slow
$ echo '#!/bin/sh
for i in $(seq 1 10000); do
      { echo yo; }
done' > /tmp/speed
$ chmod +x /tmp/slow /tmp/speed
$ time /tmp/slow
> real        0m 5.36s
$ time /tmp/speed
> real        0m 0.23s
```

4.4.3 Grep vs Fgrep

Information

`fgrep` is exactly the same thing as **`grep`** if you don't use patterns (^,$,*,etc.). **`Fgrep`** is optimized to handle such cases, particularly when you look for several different plain patterns. A difference can be found even if you look in only one pattern.

Benchmark

```
$ echo -e "line1\nline2\nline3" > /tmp/file
$ echo '#!/bin/sh
for i in $(seq 1 1000); do
      grep 3 /tmp/file
done' > /tmp/slow
$ echo '#!/bin/sh
for i in $(seq 1 1000); do
      fgrep 3 /tmp/file
```

(continues on next page)

(continued from previous page)

```
done' > /tmp/speed
$ chmod +x /tmp/slow /tmp/speed
$ time /tmp/slow
> real        0m 11.87s
$ time /tmp/speed
> real        0m 3.21s
```

4.4.4 `[ -n "text" ]` vs `[ "text" ]`

Information

The two commands test if "`text`" exists. Using `-n` slows the process and weighs down the script a little too.

Benchmark

```
$ echo '#!/bin/sh
for i in $(seq 1 1000000); do
      [  -n "$i" ]
done' > /tmp/slow
$ echo '#!/bin/sh
for i in $(seq 1 1000000); do
      [ "$i" ]
done' > /tmp/speed
$ chmod +x /tmp/slow /tmp/speed
$ time /tmp/slow
> real        0m 15.56s
$ time /tmp/speed
> real        0m 14.11s
```

4.4.5 `[ -z "text" ]` vs `[ ! "text" ]` vs `! [ "text" ]`

Information

These three commands test if `text` **doesn't** exist. `[ ! "text" ]` and `[ -z "text" ]` have a similar processing time, while `! [ "text" ]` is speedier.

Benchmark

```
$ echo '#!/bin/sh
for i in $(seq 1 1000000); do
      [  -n "$i" ]
done' > /tmp/slow1
$ echo '#!/bin/sh
for i in $(seq 1 1000000); do
      [  -n "$i" ]
done' > /tmp/slow2
$ echo '#!/bin/sh
```

(continues on next page)

(continued from previous page)

```
for i in $(seq 1 1000000); do
      [ "$i" ]
done' > /tmp/speed
$ chmod +x /tmp/slow1 /tmp/slow2 /tmp/speed
$ time /tmp/slow1
> real         0m 15.53s
$ time /tmp/slow2
> real         0m 15.60s
$ time /tmp/speed
> real         0m 14.27s
```

4.4.6 Awk vs Cut

Information

Awk, as **cut**, can be used to cut a field of a line. **Awk** can do many other things, while **cut** is a tool dedicated to this usage; it's why **cut** is a little faster for this task.

Benchmark

```
$ echo -e "field1\tfield2\tfield3" > /tmp/file
$ echo '#!/bin/sh
for i in $(seq 1 5000); do
      awk '"'"'{ print $2 }'"'"' /tmp/file
done' > /tmp/slow
$ echo '#!/bin/sh
for i in $(seq 5000); do
      cut -f2 /tmp/file
done' > /tmp/speed
$ chmod +x /tmp/slow /tmp/speed
$ time /tmp/slow
> real         0m 16.61s
$ time /tmp/speed
> real         0m 15.90s
```

4.4.7 `[ condition1 -a condition2 ]` vs `[ condition1 ] && [ condition2 ]`

Information

While `&&` is a fast built-in function, in this case it uses two processes (two test functions) instead of one. So, using `-a` is a little faster, as the "AND" function itself is slower but makes it possible to use only one process.

Benchmark

```
$ echo '#!/bin/sh
for i in $(seq 1 1000000); do
      [ "$i" ] && [ "$i" ]
done' > /tmp/slow
$ echo '#!/bin/sh
for i in $(seq 1 1000000); do
      [  "$i"  -a "$i" ]
done' > /tmp/speed
$ chmod +x /tmp/slow /tmp/speed
$ time /tmp/slow
> real        0m 23.94s
$ time /tmp/speed
> real        0m 22.29s
```

4.5 The new tazwok illustrated!

author godane, gokhlayeh, linea, bellard

Note: This wiki tutorial is out of date, please refer to the **cookutils** docs in Mercurial[210] or its backup[211]

Certain elements present in the receipts are not necessary any more with the new version of **tazwok**. During migration, a certain number of problems appear. Information concerning these points are available below, with examples.

By simplifying the writing of the receipts, we simplify any contributions and that benefits all!

DEPENDS/BUILD_DEPENDS:

Tazwok uses from now on the variable DEPENDS to find the necessary dependencies of compilation. Here's how that functions:

- **Tazwok** lists the tree of all dependencies starting with the variable DEPENDS
- For each package, if there exists an associated -dev package, it adds to the dependencies
- **Tazwok** does the same for BUILD_DEPENDS

Up to now, when a package had both a dependency and build dependency, the receipt looked like this:

```
DEPENDS="pkgX"
BUILD_DEPENDS="pkgX pkgX-dev"
```

Now this is enough:

```
DEPENDS="pkgX"
```

Where there were trees of more complex dependencies, the receipt looked like this:

[210] http://hg.slitaz.org/cookutils/raw-file/tip/doc/cookutils.en.html
[211] http://hg.tuxfamily.org/slitaz/cookutils/raw-file/tip/doc/cookutils.en.html

```
# pkgY depend's on pkgX
DEPENDS="pkgY"
BUILD_DEPENDS="pkgY pkgY-dev pkgX pkgX-dev"
```

Now this is enough:

```
DEPENDS="pkgY"
```

The receipts also contain many redundancies in the definition of the dependencies, for example:

```
# pkgY depend's on pkgX
DEPENDS="pkgY pkgX"
```

Here, needless to say `pkgX` will be installed alongside `pkgY` anyway (**Tazpkg** manages dependencies automatically!).

While taking these three examples, it appears that about half of the packages in `DEPENDS`/`BUILD_DEPENDS` can be withdrawn from the receipts without modifying the behavior of the system.

Tip: An automated cleaning using some scripts is envisaged, after all the receipts were compiled at least once successfully by using the new version of **tazwok**. In the meantime, these tips can be applied to writing new receipts for simplicity or manually when updating/correcting.

Examples:

- **graveman**: http://hg.slitaz.org/wok/rev/7f0604e0bde0
- **enlightenment** & cie: http://hg.slitaz.org/wok/rev/85cd798d6997

TARBALL/WGET_URL/SOURCE/download from the VCS

This is important: always put the necessary tools to download/decompress sources in `DEPENDS` or `BUILD_DEPENDS`. This makes it possible for **tazwok** to define a correct order of cooking (try not to cook a package which needs **wget** before **wget** itself).

The packages affected by this are:

- **wget** url for https, ftps and some URLs that **busybox** does not include
- **mercurial/subversion/git**: these are used to obtain the source
- **tar/unzip**: sometimes necessary to unpack the sources

By default, **tazwok** re-compresses sources with the format `.tar.lzma`. It names them `PACKAGE-VERSION.tar.lzma`, or `SOURCE-VERSION.tar.lzma` if the `SOURCE` is defined. Note: choosing the name of the archive is now the only function of the `SOURCE` variable!

Tazwok now supports files or "weird" URLs (`download.php?version=foo&blah=Idontknowwhat`). The logic is: if `WGET_URL` does not end with tarball, then it names the downloaded file as tarball.

Tazwok also supports the use of **mercurial/subversion/git** in the `WGET_URL`. The syntax is:

```
WGET_URL="subversion|svn://svn.mplayerhq.hu/mplayer/trunk
```

An optional variable is `BRANCH`: it allows you to specify the revision / tag / branch to use (see examples below). Where `BRANCH` is used, it is important that the `$VERSION` is also part of its definition.

Note that the sources will be obtained through the requested tool, then packaged in `.tar.lzma`. The archive will be named as explained above. This means that the variable source can be used to ensure that many receipts use the same repository without creating multiple archives.

First, it helps to know what revision is installed when using the package manager. Second, it allows you to differentiate between **tazwok** compressed sources. Indeed, if the archive keeps the same name, it will not be re-downloaded, which is undesirable when trying to update the package.

Examples:

- Here **wget** was necessary: http://hg.slitaz.org/wok/rev/012847ddd0cb
- **Tinyproxy** did not report the URL of its source code and is corrected: http://hg.slitaz.org/wok/rev/25967da0e1af
- `WGET_URL` now supports `xpi`: http://hg.slitaz.org/wok/rev/37738b3ee08f
- `WGET_URL` with a "weird" URL: http://hg.slitaz.org/wok/rev/102de15fea8d
- `WGET_URL` using **git**: http://hg.slitaz.org/wok/rev/e06d60ae03eb
- `WGET_URL` using **subversion**: http://hg.slitaz.org/wok/rev/c4c54646489a
- `WGET_URL` using **mercurial**: http://hg.slitaz.org/wok/rev/756ed4b1daac
- It was difficult to choose how to define `BRANCH` and `VERSION` for **aufs**: http://hg.slitaz.org/wok/rev/67231cfc5475
- Here two sources of records were in conflict and resolved by `SOURCE`: http://hg.slitaz.org/wok/rev/b891cba4f48e
- **slitaz-dev-tools** contains the sources for SliTaz tools that contain very little code. Use `SOURCE="slitaz-dev-tools"` in receipts that use this method to avoid having duplicate tarballs: http://hg.slitaz.org/wok/rev/808826645cc2

Exceptions concerning cooking dependencies

In some cases, no cooking dependencies are installed:

- For receipts with `WANTED`
- For receipts without `compile_rules()`

Note: Note that packages may be required to obtain/decompress the source code and will still be installed if they are in `DEPENDS`/`BUILD_DEPENDS`. These are **wget**, **mercurial**, **subversion**, **git**, **tar** and **unzip**.

If you don't want to use `compile_rules()` but want to force the installation of all cooking dependencies, there's a little hack:

```
compiles_rules()
{
      :
}
```

Examples:

- Removal of `compiles_rules()` to avoid installing unnecessary cooking dependencies: http://hg.slitaz.org/wok/rev/f579356b437f
- Removing a hack with fake `compiles_rules` which was useless… http://hg.slitaz.org/wok/rev/5b4581f8e476

Define `src/_pkg` & move to the right place (hacks in the receipt)

By default, the new sources in **tazwok** are placed in `$WOK/$PACKAGES/$PACKAGE-$VERSION`: it renames the parent directory of sources if necessary. Up to now, `$src` was not properly defined for receipts using both `SOURCE` and `WANTED`. Many receipts implement their own solution in different ways, which is difficult to consider a standardized way and can cause compatibility problems.

If **tazwok** detects `src=/_pkg=` in a receipt, it continues to use the old behavior to ensure compatibility (this produces errors in some cases). It is no longer necessary and not ideal.

The hacks in the receipt that move the source to the right place are no longer needed either and can also cause problems.

In conclusion, it is better to consider that `$src/$_pkg` are defined by default and try to rely on it as much as possible.

Examples:

- Removing `src=` by Godane: http://hg.slitaz.org/wok/rev/a1c1d35d9f92
- `src=/_pkg=` can/should also be removed from `WANTED`: http://hg.slitaz.org/wok/rev/07adb7cbd0c8
- Here, an old hack was the problem: http://hg.slitaz.org/wok/rev/62f6142d9fb3
- Sources are now *always* placed in a sub-directory `$src` http://hg.slitaz.org/wok/rev/e64069568fe7
- Another case: call the **configure** script from a separate compilation folder (`*-build`): http://hg.slitaz.org/wok/rev/7461a0c31d62
- Fixed **dmraid**: http://hg.slitaz.org/wok/rev/f5b7e0c47763 http://hg.slitaz.org/wok/rev/59ea9409ad8a

Set the default paths in `configure`:

Tip: See `/etc/slitaz/slitaz.conf`, `/etc/config.site` and the new review model in place in the new **tazwok** tree

The new version of **tazwok** attempts to pass the default paths to configure using the environment variable `CONFIG_SITE` calling `/etc/config.site`, which works in most cases. Nevertheless

`configure` scripts are specific to each source and sometimes `CONFIG_SITE` may not be supported. For this reason, it's best to remove the definitions of paths if necessary and do so on a case by case basis when updating the receipt and making sure everything works.

In rare cases, this produces functionality problems. It happens that some receipts did not use the default paths used by `CONFIG_SITE` and an update function `genpkg_rules()` is then mandatory.

Examples:

- A file did not install properly in **`acl`** and is corrected by `CONFIG_SITE`: http://hg.slitaz.org/wok/rev/f831ecb652a6
- Another example: http://hg.slitaz.org/wok/rev/259214792e30

Tip: `CONFIG_SITE=` can be used in receipts to use a different file other than the default (can be useful for **`gnome`** packages or something like that. . .)

`DESTDIR=$PWD/_pkg`

`DESTDIR` is passed to **`make install`** using the environment variable of the same name. The new path for installation is `$WOK/$PACKAGE/install`. This will remove the source folder after packaging, it does not contain any file used by a receipt in its `genpkg_rules()`.

The majority of the receipts still use `DESTDIR=$PWD/_pkg`. However, if no receipt redefines the variables `src/_pkg`, **`tazwok`** will automatically move it to `$WOK/$PACKAGE/install`.

In some cases, as with other variables, `DESTDIR` is not taken into account or the package is not installed by **`make`**. In these cases, the variable `$DESTDIR` is available to define the installation directory in the receipt.

In rare cases, this behavior causes incompatibilities. This happens when receipts define the path to the installation folder without using `src/_pkg`. The solution is not to set these paths in the receipt (calling the main receipt with `WANTED` included), make sure the installation is done well in `$WOK/$PACKAGE/install` and trust the variables provided by **`tazwok`**.

Examples:

- Removing `_pkg=` & `DESTDIR=` at the same time for this to work: http://hg.slitaz.org/wok/rev/cf088243a4a5
- Withdrawal of "useless" references to `$src` so that the sources are withdrawn http://hg.slitaz.org/wok/rev/0731792c3994 http://hg.slitaz.org/wok/rev/5d6340961543
- **`Bash`** does not take into account the `DESTDIR` environment variable: http://hg.slitaz.org/wok/rev/fa7b7514e1d8
- **`acl attr`** does not include `DESTDIR` (in this installation the destination was still `$PWD/_pkg`): http://hg.slitaz.org/wok/rev/fa7b7514e1d8

`MAKEFLAGS`

`MAKEFLAGS` is also passed to make using the environment variables; once again this does not always function. In the majority of the cases, then `-J 4` can be removed. In certain cases, it is necessary to pass `MAKEFLAGS` to make directly in the receipt: **`make $MAKEFLAGS`**

`Tazwok` automatically defines the value for `$MAKEFLAGS` according to the number of cores which the processor contains, `-j4` should thus be removed from all the receipts to make it possible to compile on computers that have more resources (4 cores can use `-j5`)

Problems with `MAKEFLAGS`:

So far, only receipts with `-j4` were using multi-threaded compilation, whereas now all **`make`** and **`make install`** commands use it. This behavior can cause errors. Some sources do not support multi-threaded compilation but do not disable it. This is the most common problem associated with the changes explained here.

Problems in compiling:

During compilation, it happens that libraries are based on others compiled with the same sources. If they are compiled at the same time, that causes an error in connection with a missing library. In this case, one sees in the compilation text which library in question started to be compiled some lines earlier, but that this process was not yet finished. To solve this problem, just add `-j1` to **`make`**. It is the most common error, but there are others that are different or rarer which take a similar form.

Problems in installation:

The characteristic of this error is that the installation stops and an error message says that it is impossible to create a folder because it already exists: a parallel process is actually creating it. In this case, just add `-j1` to **`make install`**.

Example:

- Several changes are explained here in the receipt for **`gettext`**: http://hg.slitaz.org/wok/rev/9411655af0e2

Variables `$stuff`, `$wanted_stuff` and `$fs`

From now on, the variable `$stuff` is available and returns the stuff recorded in the receipt, it uses an absolute path. The variable `$wanted_stuff` returns to the file stuff in the package defined in `WANTED`, if any. The variable `$fs` refers to the future contents of the package in `taz/*/fs`, as before, the difference is that now `$fs` uses an absolute path.

Examples:

- A commit with several changes regarding the variable `$stuff`: http://hg.slitaz.org/wok/rev/be13f25e790b
- A correction necessary when we have made an absolute path `$fs`: http://hg.slitaz.org/wok/rev/8c897d2542ab

Do not use `exit` but `return`

Now when cooking several packages with a list: **`tazwok`** does not call for a new **`tazwok`** cook. There is only one **`tazwok`** session so the execution is faster. If a receipt uses **`exit`**, it leaves the **`tazwok`** session and the following list is not cooked.

Example:

- Removing all **`exit`**s from the wok receipts: http://hg.slitaz.org/wok/rev/0b4cf0d9e1b5

Conclusion — What to do when updating a receipt:

- Remove `src=/_pkg=` from the receipts and those which declare it as `WANTED`.
- Remove `DESTDIR=$PWD/_pkg`; if it doesn't function, or if the means to define the repository of installation is not **make**+`DESTDIR`, use `$DESTDIR` rather than `$PWD/_pkg`.
- Remove the definition of default paths and see if it works, otherwise leave.
- Remove `-j4` and see if it works; If multi-threaded does not work, re-activate it using `$MAKEFLAGS`; if multi-threading causes problems, add `-j1` to the right place.
- Remove the redundant `BUILD_DEPENDS` / `DEPENDS`.
- Check that packages are created correctly, otherwise update the paths in `genpkg_rules()`.
- Try to declare all sources in the receipt so that SliTaz can be compiled without an internet connection (requires downloading any sources beforehand).
- Check that the packages needed to download/extract the source code are defined in `BUILD_DEPENDS`.
- Check that **exit** is not used in the receipt.

Some more complex cases...

I put these at the end because there are already too many to be integrated :) The items below correspond to specific cases.

Variable `COOK_OPT`

This new variable contains options that alter the behavior of **tazwok**. These are useful in very special cases.

genpkg=: In the receipt, `PACKAGE` defines a set of priorities to pack up the receipts which contain `WANTED=` "PACKAGES" (and only them!). If you include multiple packages, separate them with double points ':'. If packages are not defined in this option, they will be packaged later, in alphabetical order (default)

Used in **glibc**: http://hg.slitaz.org/wok/file/tip/glibc/receipt

!repack_src: Disables re-compressing sources format `.tar.lzma`.

Ruby-pkgconfig used for the sources remain in `gem`: http://hg.slitaz.org/wok/file/tip/ruby-pkgconfig/receipt

!unpack: Prevents decompression of the archive-source in the wok.

This is used by **ruby-pkgconfig** as well (see link above)

This is the only case yet!

Cooking the toolchain

To cook the SliTaz toolchain, we use a temporary toolchain. Some receipts use specific rules in this step. During the cooking of this temporary toolchain, the software concerned is not packed up but installed directly in the chroot built for this purpose. The packages concerned are listed in the variable `SLITAZ_TOOLCHAIN` in the configuration file `/etc/slitaz/slitaz.conf`

Additional features are:

- `precook_tmp_toolchain()` — Used only by **gcc** & **binutils** for the moment, because they are cooked twice during the preparation of the temporary toolchain.
- `cook_tmp_toolchain()` — Used mostly by `SLITAZ_TOOLCHAIN` packages to define how they should be compiled for the temporary toolchain. When `cook_tmp_toolchain()` is absent, `compile_rules()` is used instead. This avoids writing two identical functions. Note that in this case. **./configure** does not set the default paths in the receipt, because the temporary toolchain must be able to do it via the environment variable `CONFIG_SITE`. Indeed, packages compiled during this stage are not installed in the usual place but in `/tools`.

Examples:

- **Binutils**: http://hg.slitaz.org/wok/file/tip/binutils/receipt
- **Gettext**: http://hg.slitaz.org/wok/file/tip/gettext/receipt
- **Bash**: http://hg.slitaz.org/wok/file/tip/bash/receipt
- **Patch** does not need `cook_tmp_toolchain()`: http://hg.slitaz.org/wok/file/tip/patch/receipt
- **Autoconf** either: http://hg.slitaz.org/wok/file/tip/autoconf/receipt

tazwok get-src / report in receipt

Report is a **libtaz** module for organizing the display commands in the terminal and making logs available including the http://bb.slitaz.org interface. It can also be used in receipts, as follows (this is abstract, examples of actual applications follow):

```
compile_rules() # For example
{
    report open-bloc # compiles_rules is a step, declaring that there␣
↪will substeps
    report step "Action machine"
            ...
    report step "Action true"
            ..
    report close-bloc # Close the previously open block
}
```

Specifically, there is one case where we use it: when using **get-src tazwok PACKAGE --target=...**. This command creates a new step (postponement step). We need to open a block and close it afterwards, as well as adding several other deferral steps "…" so the log and display in the terminal is correct. Each step defers the previous step, if we do not open a block, **tazwok get-src** would close the stage "Executing compiles rules"

Note that report close-bloc must be completely executed, otherwise the log/display will be broken. That's why we use **{ report-block closed; return 1; }** rather than return nothing at all.

The practical use of this `get-src` in **tazwok** is that you can unpack the sources of a designated PACKAGE at target.

In the examples below, observe the correlation between the delay step and that displayed in the log. Observe also the correlation between **tazwok get-src** and the message "Checking for source tarball …" in the log. You learn how the report `open-bloc/closed block` creates a subset in `genpkg_rules` (named "Executing compile_rules" in the log). If it were not for this `open-bloc`/`close-bloc`, these new steps would be posted to the result of "Executing compile_rules" This is not what we wanted.

Examples (receipt + log):

- **`Linux`** needs patches contained in the sources of **`aufs`**, Godane took the opportunity to improve the log. Receipt: http://hg.slitaz.org/wok/file/tip/linux/receipt; log: http://bb.slitaz.org/log.php?version=cooking&package=linux
- **`Gcc`** uses several other sources of packages during the cooking of the temporary toolchain. Receipt: http://hg.slitaz.org/wok/file/tip/gcc/receipt; log: http://bb.slitaz.org/log.php?version=cooking&package=tmp-toolchain-gcc
- **`mingw32-gcc`** was corrected using this approach, it also allowed us to declare all sources used. Commit: http://hg.slitaz.org/wok/rev/fd43246b4613; log: http://bb.slitaz.org/log.php?version=cooking&package=mingw32-gcc

4.6 Cook a flavor from scratch

author gokhlayeh, oui, linea

4.6.1 Introduction

Welcome to this howto! it's the base for a second scratchbook which explains how to recompile SliTaz entirely from scratch including the toolchain. All jobs will be done using tazwok-experimental (actually in Alpha state). Even if the scripts have some major issues, the steps explained here should do the job without too many problems. Please report any problems to the author (tazpkg info tazwok-experimental gives you my mail).

Having some knowledge about how the legacy SliTaz tools and basic commands work will help you to understand what's going on in this howto. However, simply copying/pasting the given command lines will probably work.

4.6.1.1 Explanation

`Tazwok` now provides tools to cook the wok from scratch using a minimal chroot. You have to use a special version of the wok called wok-experimental. It contains the needed patches which should be applied on top of the cooking wok. Explanations about how to obtain the wok experimental are here: *Prepare the experimental wok* (page 252).

Tools needed

- A functional cooking system (live or installed), you also can use the slitaz-experimental-base[212] ISO.
- tazwok-experimental-0.0.2, tazchroot-0.0.1 and libtaz-0.0.1. Note: installing tazwok-experimental removes tazwok legacy.
- An internet connection to download sources.

[212] http://people.slitaz.org/~gokhlayeh/experimental/iso/slitaz-experimental-base.iso

Steps

- Make the chroot (in this how-to we use the packages of the new toolchain from the experimental repository; but this should work with the cooking toolchain).
- Cook a temporary toolchain: it's a cross-compiled toolchain which will cook the "real" toolchain (the packaged one) without linking anything to the host system.
- Cook the definitive toolchain.
- Cook some of the other packages of one of the SliTaz flavors.
- Create the iso.
- Burn, boot & enjoy :).

4.6.1.2 Install the tools

Tip: Note: all the following commands should be executed as root. You don't have to add the experimental repository if you wish to use the cooking toolchain.

Add the cooking-experimental repository as undigest:

```
# tazpkg add-undigest experimental http://people.slitaz.org/~gokhlayeh/
↪experimental/packages
```

Give it priority over the main repository:

```
# echo experimental > /var/lib/tazpkg/priority
```

Install the cooking tools:

```
# tazpkg get-install tazwok-experimental
# tazpkg get-install tazchroot
```

4.6.1.3 Cook toolchain

Tip: The following command-lines work for a wok at: `/home/slitaz/experimental/wok`. If you put the wok elsewhere, you can use the option `--SLITAZ_DIR=address`; where the address is equivalent to `/home/slitaz`. This option must be used each time `--SLITAZ_VERSION` is used. You can also define `SLITAZ_VERSION` & `SLITAZ_DIR` globally using `/etc/slitaz/slitaz.conf`.

Configure the *chroot*:

```
# tazwok configure-chroot --SLITAZ_VERSION=experimental
```

If you have at least 1GB RAM free you can put the minimal chroot in RAM speeding-up the cooking process:

```
# sed 's~chroot_dir=.*~chroot_dir=/tmp/chroot-experimental~' -i /home/
↪slitaz/experimental/tazchroot.conf
```

All-in-one command to cook the toolchain packages:

```
# tazwok cook-toolchain --SLITAZ_VERSION=experimental
```

At the end of this operation the chroot should be removed. If you have modified its address before, you have to do this manually:

```
# rm -r /tmp/chroot-experimental
```

Toolchain packages are now ready to be used. They're actually are in the packages-incoming repository. If all was cooked fine it's possible to push them to the classical packages directory using:

```
# tazwok check-incoming --SLITAZ_VERSION=experimental
```

You need to tell to **`tazpkg`** that you now have a local version of experimental:

```
# If you had already defined the experimental repository:
echo "/home/slitaz/experimental/packages" > /var/lib/tazpkg/undigest/
↪experimental/mirror
tazpkg recharge
# Else:
tazpkg add-undigest experimental /home/slitaz/experimental/packages
echo experimental > /var/lib/tazpkg/priority
tazpkg recharge
```

Recook the toolchain packages except core (linux-api-headers/glibc/binutils/gcc) over themselves to consolidate it; in fact it's not really needed with an actual configuration but it's generally good to do to solve loop dependencies. It warrants consistency of the toolchain before and after an update of non-core toolchain packages. Note that core-toolchain should never be updated in this way — but by reusing the cook-toolchain script:

```
# tazwok chroot --SLITAZ_VERSION=experimental
# tazwok build-depends toolchain-cooklist | sed '1,/^gcc$/d' > /tmp/
↪consolidate.list
# tazwok cook-list /tmp/consolidate.list
# rm /tmp/consolidate.list
```

As packages has been re-cooked, you have to update the packages repository once again:

```
# tazwok check-incoming
```

Still in the chroot for the next step.

4.6.1.4 Cook packages of a flavor

First you need data about cooking flavors:

```
# cd /home/slitaz/experimental/flavors
# hg clone http://hg.slitaz.org/flavors .
```

Generate the cooklist for a given flavor. Note: for the next step, use the chosen flavor name instead of FLAVOR in the command line:

```
# tazwok gen-cooklist --list=/home/slitaz/experimental/flavors/FLAVOR/
↪packages.list > /tmp/FLAVOR.list
```

This list contains some already cooked packages, remove them:

```
cat /tmp/FLAVOR.list | while read p; do
     grep -q ^$p$ /home/slitaz/experimental/packages/packages.txt &&
             sed "/^$p$/d" -i /tmp/FLAVOR.list
done
```

The cooklist is now ready to create the packages:

```
# tazwok cook-list /tmp/FLAVOR.list
```

Update packages repository:

```
# tazwok check-incoming
```

Tip: You can repeat these steps each time you wish to add new packages to your repository.

Still in chroot for the next step.

4.6.1.5 Create the ISO

Actually **`tazlito`** uses `/home/slitaz/flavors`. Create a symbolic link:

```
# ln -s /home/slitaz/experimental/flavors /home/slitaz
```

Set release as experimental for the future ISO:

```
# cd /home/slitaz/flavors/FLAVOR
# mkdir -p rootfs/etc
# echo experimental > rootfs/etc/slitaz-release
```

Configure your ISO to use your local repository:

```
# mkdir -p rootfs/var/lib/tazpkg
# echo /home/slitaz/experimental/packages > rootfs/var/lib/tazpkg/mirror
```

Use tazwok-experimental instead of tazwok and add tazchroot:

```
# sed 's/tazwok/tazwok-experimental/' -i packages.list
# echo tazchroot >> packages.list
```

Pack FLAVOR:

```
# tazlito pack-flavor FLAVOR
```

Get FLAVOR:

```
# tazlito get-flavor FLAVOR
```

Generate ISO:

```
# tazlito gen-distro
```

Save ISO in your home dir:

```
# mv /home/slitaz/distro/slitaz-FLAVOR.iso /home/slitaz/distro/slitaz-
↪FLAVOR.md5 /home/slitaz/experimental/iso
```

Exit chroot:

```
# exit
```

CHAPTER 5

Forum posts

author linea, trixar_za

How to build your own kernel modules[213]

[213] http://vanilla.slitaz.org/index.php?p=/discussion/comment/5404/#Comment_5404

CHAPTER 6

Handbook

author pankso, linea, oui, jozee, domcox, naitsirhc, brianperry, emgi

- *Network Configuration* (page 315) — Ethernet, DHCP, static IP or PPPoE ADSL connection, Firewall.
- *PSTN* (page 322) — Use the telephone network to establish a connection with PPP.
- *System Administration* (page 332) — Mount devices, NFS, manage users and groups, adjust system time.
- *X Window System* (page 341) — Xorg server, Slim Login and Window managers.
- *SliTaz and System Security* (page 344) — SliTaz and system security.

Flavor

- *Generating a Customised LiveCD or LiveUSB* (page 346) — Generate your own flavor using **`Tazlito`**.
- *LiveUSB media* (page 352) — Create bootable USB media using **`TazUSB`**.
- *Hacking SliTaz LiveCD* (page 355) — Manipulate and play with the ISO image of LiveCD.
- *Chroot environment* (page 360) — Build a chroot to protect the host system.

Server Applications

- *Server applications* (page 362) — Install and config of CMS, Wiki, etc.
- *LightTPD Web Server* (page 363) — Configure and use the LightTPD web server.
- *Secure SHell (SSH)* (page 366) — Secure login using **`Dropbear`** SSH client/server.

About this Handbook

This is the SliTaz GNU/Linux English Handbook, a collection of instructions and manuals about the distribution. This book is a community effort to provide high quality documentation for SliTaz users, the first page was created on the 26 of February 2008. The SliTaz Handbook is always in development and follows the distribution changes and improvements.

6.1 General

6.1.1 Using the LiveCD

author pankso, jozee, linea, seawolf, natty537, seacat, brianperry

SliTaz can be used straight from a CD-ROM or USB memory stick, without being installed to a hard drive. There are a number of ways to use the Live media, with options to customise how it starts.

SliTaz runs entirely in memory (RAM), independently of the installed host system. It will not damage your other operating system(s) in any way, so you are perfectly safe to try out SliTaz.

6.1.1.1 Quick Start

To start using SliTaz from a CD-ROM, just burn the ISO image onto a blank disc and reboot your computer, leaving the disc in your CD-ROM drive. SliTaz will load automatically, detecting your hardware configuration to start.

If SliTaz LiveCD Does Not Start

In most cases, your computer is already configured to boot from the CD-ROM. If the SliTaz splash screen does not appear, you can change the boot order via the BIOS set-up interface. This is different with each computer but can often be done by pressing a key such as the `F11`, `F12` or `Esc` button directly after turning on, before your operating system starts. With some BIOS's you can use the 'Boot Selection Popup' by pressing a `F*` key (eg. `F8`). You can then change the boot sequence and settings so that the CD-ROM comes first. Finally, save your changes before leaving the BIOS configuration interface.

When the SliTaz Live media starts, the splash image will be presented. This is the **`isolinux`** bootloader, which affects options to start SliTaz. You can just press `Enter` to use the default settings, or enter options.

Tip: Selecting *Help & Options* at the boot-splash will display help and information.

When the loading process has finished, you can log-in to the desktop as the *tux* user, without any password. To use the administrator account, or **root**, you can start a **`Terminal`** and type the command **`su`**. The default password is **root**.

6.1.1.2 Boot Options

The SliTaz LiveCD accepts various boot options at the prompt. There are two types of options: options handled by SliTaz software and those generally handled by the Linux kernel.

The options for SliTaz are used by various start-up scripts; the parameters such as the VGA mode are managed directly by the Kernel (kernel boot parameters). To pass options at start-up, either press `Tab` at the language selection screen or just precede your commands with *slitaz* when the splash screen and **boot:** prompt is displayed. For example:

```
slitaz modprobe=nvidia nomodeset
```

Tip: The Linux kernel keeps options that were passed. These can be seen in the text file `/proc/cmdline`. You can view this information by running the command:

```
cat /proc/cmdline
```

Parameters of the Linux Kernel

On GNU/Linux systems, parameters specific to the Kernel vary greatly depending on the configuration used during the build. The SliTaz-built kernel has few core modules, compensated by loading others on-demand. This means few modifiable parameters are available at start-up. However, you can disable the emulation of a math coprocessor via:

`no387` Disables the emulation of a math co-processor.

`irqpoll` Turn on in case of problems with interrupts, shown by problems with the CD-ROM.

`vga=XXX` Specifies the kernel graphics mode. The SliTaz kernel displays the Tux penguin logo and manages the display of the Linux terminal by providing a basic video output mode, called the VGA/VESA frame-buffer. The following table lists the codes used; select a resolution and colour combination:

⇓ Colours	640x480	800x600	1024x768	1280x1024	1600x1200
256	769	771	773	775	796
32768	784	787	790	793	797
65536	785	788	791	794	798
16,8M	786	789	792	795	799

Tip: `vga=normal` lets the system set a working resolution automatically.

Tip: Once the system has started you have access to six pseudo-terminals via the key combinations `Ctrl+Alt+F1` through `Ctrl+Alt+F6`. The key combination `Ctrl+Alt+F7` and upwards are reserved for graphical output.

6.1.1.3 Slitaz Parameters

`home=usb` Specifies a `/home` directory to use within the Live environment. This will include your bookmarks, downloads and desktop customisations. To store your data permanently, you need USB media with a partition formatted in ext3; see the *persistence* (page 146) page for more information. In most cases `home=usb` can be used for `sdb1`, or `home={devname}` where the equivalent `/dev` node can be specified. Note that you can also specify the device using the partition UUID or label by using `home=*`. Example:

```
slitaz home=sdb1
```

Prepare USB media All USB media can formatted in the native Linux ext3 filesystem. Ext3 is a journalised, stable filesystem, it allows you to keep permissions on all files and is much more secure than the default Windows FAT32 filesystem. To format USB media you have a few options: the command line with **`mkfs.ext3`**, the **`tazusb`** utility or graphically with **`Gparted`**. To get a full list of available partitions including the USB drive you can use the command **`fdisk -l`** and then format. Example:

```
# fdisk -l
# tazusb format /dev/sda1
```

`lang=XX, kmap=XX` Sets the system language & keyboard mapping. Each are codes such as `en`, `de` or `fr_CH`. Alternatively, set your country code and press `Enter` to sets the locale and refreshes the boot-loader; other options can then be entered as normal. To skip the language and keyboard configuration you can simply type options on the command line, for English/UK:

```
slitaz lang=en kmap=en
```

`config=<device>,<path>` Executes a script at SliTaz boot time. The script can be located on external media or a HD partition, specified with the `device` & `path` variables. For example, the script can mount an ISO image on `/usr` to save memory and boot the LiveCD on computers with only 32 Mb of RAM. An example with a script named `slitaz.sh` located on the first disk and partition:

```
slitaz config=/dev/hda1,slitaz.sh
```

`screen=<height>x<width>x<colours>` Specifies the desktop screen resolution. These follow the standard pattern, e.g. `1024x768x24`

```
slitaz screen=1024x768x24
```

Tip: The `screen=text` option disables the graphical desktop & **`SLiM`** login manager.

`sound=no` Disables sound completely. This loads no sound-related kernel modules.

```
slitaz sound=no
```

`sound=noconf` Skips automatic configuration of your sound card; you must configure it manually later.

`modprobe=<modules>` Loads specific kernel modules. Many can be loaded by separating with commas.

```
slitaz modprobe=module1,module2
```

`laptop` Loads `ac` and `battery` Kernel modules — useful for laptop computers.

`previous` Used by the **`TazUSB`** utility to roll-back to a previous filesystem.

```
slitaz previous
```

6.1.1.4 The Desktop

When the system has finished its initialization, the screen is cleared and the login prompt (**`SLiM`**) is displayed. You can choose here to login as the regular *tux* account (without a password) or as the administrative *root* account (with the **root** password).

The desktop is powered by **`Openbox`**. You can start applications from the menu at the lower-left of the screen. Applications are classified by category and are available in English. Menu, theme and wallpaper can all easily be changed to your needs/preferences, and personal settings and data can be stored on various USB media (Flash key, SD card, etc).

6.1.1.5 Text Mode

Important: If you are new to SliTaz or Linux in general, a graphical desktop is highly recommended.

The above information applies also to the text-mode log-in prompt. Once logged in, you can use the many text-mode applications available in SliTaz, such as the basic BusyBox operation, the GNU text editor **Nano**, or the **Clex** file manager. Just type the name of the application you wish to start. There is a *Command Line Reference* (page 281) page to get you started.

Tip: To launch a graphical desktop session from the text-mode prompt (if you have passed the `screen=text` option, or if **SLiM** is not configured to run at startup, for example) just type **startx**.

6.1.2 Desktop

author jozee, linea, naitsirhc

6.1.2.1 Introduction

The default SliTaz desktop is brought to you using different components of the LXDE project. **Openbox**, **PCmanFM** and **LXPanel** combine to implement a Desktop providing simplicity and functionality.

6.1.2.2 Openbox

Openbox is a fast, lightweight, simple, themeable window manager, it is the window manager by default on SliTaz. A window manager is an application that runs on top of the X server to control the appearance of windows. It can then place, cut and re-size windows at will.

Openbox provides a context menu via a right click on the desktop, this menu can be changed by editing a configuration file. The key combinations `Alt+Tab` allow you to list and navigate through open windows. **Obconf** can be used to graphically configure the window manager and various small tools (specific to SliTaz) allow you to have a simple, stylish and coherent desktop. By default, SliTaz uses 2 virtual desktops.

The Desktop provided by SliTaz complies with the Freedesktop standards, the file manager **PCmanFM** allows for management of desktop icons, drag and drop and the mounting of devices with a solitary click. The panel menu, taskbar, icons, etc are powered by **LXpanel**.

6.1.2.3 Configuring Openbox

Most options can be configured graphically using the **Obconf** utility located in the "Preferences" menu. The configuration file can also be modified using your favorite text editor, this is located in your home directory `~/.config/openbox/rc.xml` and is a XML file. The keyboard shortcuts are defined in the `<keyboard>` section of the configuration file.

6.1.2.4 Themes

Openbox supports themes through a single file using syntax specific to the window manager. Several default themes are provided, they can be selected via **Obconf** and are found in `/usr/share/themes`. Each system user can install their own themes in the directory `~/.themes` either manually or via **Obconf**. If you want to create your own themes for SliTaz, then the easiest way is to copy and rename an existing theme and then edit the file `themerc`. Optional themes can also use images for buttons, icons, etc. These images can be created or modified via an image editor such as the **GIMP** or **mtPaint**.

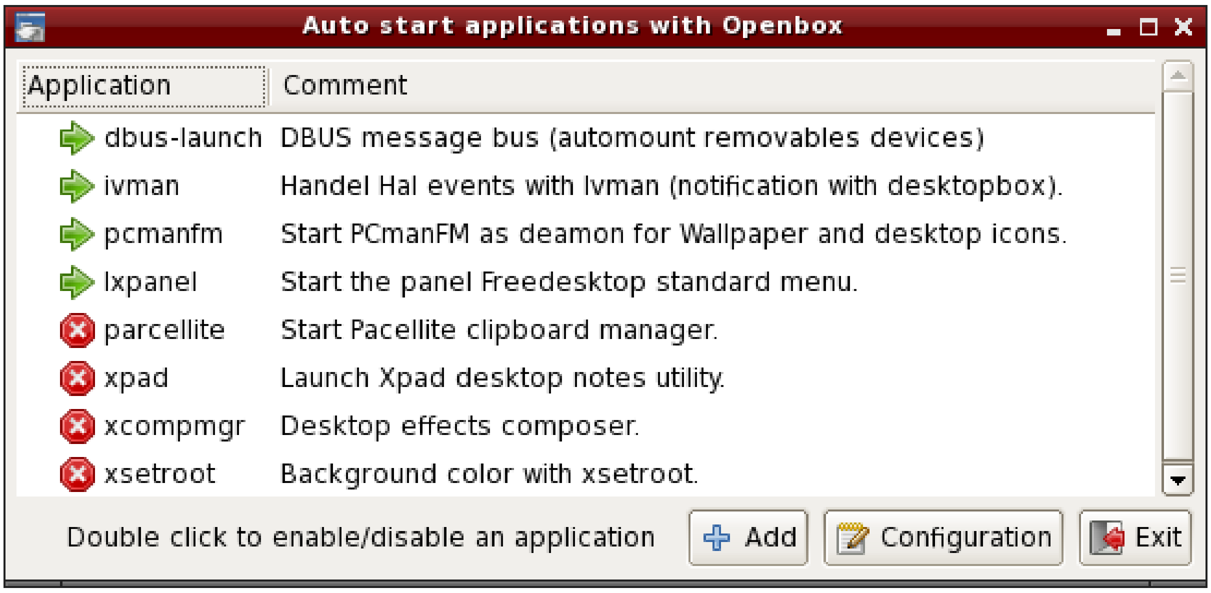
On the Internet you will find many more themes created by the **Openbox** community. More information can be found on the official **Openbox** website[214].

6.1.2.5 Context Menu

The menu is in the file `~/.config/openbox/menu.xml`. Besides editing from a text editor, it also possible to edit this file with **obmenu** (not installed by default).

6.1.2.6 Applications started automatically

When starting in graphical mode, **Openbox** allows many applications to start automatically via the `~/.config/openbox/autostart.sh` script. By default, using this script, SliTaz starts the file manager **PCmanFM** for the management of screen and desktop icons, the panel (**LXpanel**) for the menu, and **DBUS** for the management of devices or media such as USB keys. To add or remove applications launched at startup of the session, you can edit the script or use the small SliTaz GUI located in the menu *Preferences → Auto started applications*:



Application	Comment
dbus-launch	DBUS message bus (automount removables devices)
ivman	Handel Hal events with Ivman (notification with desktopbox).
pcmanfm	Start PCmanFM as deamon for Wallpaper and desktop icons.
lxpanel	Start the panel Freedesktop standard menu.
parcellite	Start Pacellite clipboard manager.
xpad	Launch Xpad desktop notes utility.
xcompmgr	Desktop effects composer.
xsetroot	Background color with xsetroot.

Double click to enable/disable an application

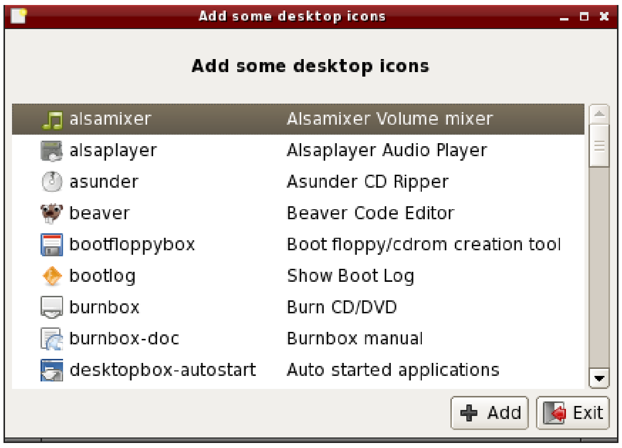
6.1.2.7 Wallpaper and icons using PCmanFM

Openbox doesn't manage the screen natively, you can use an external tool, this allow more freedom of choice. The default desktop on SliTaz uses the file manager **PCmanFM** to display pictures as wallpaper and have desktop icons. Alternatively, you can use the package **hsetroot** to display a picture or **xsetroot** for a solid color. **PCmanFM** is started with the **Openbox** session as a daemon; ie, it runs in the background and launches faster. To change the current background image, you can go through the file manager preferences or via the menu *Preferences → Wallpaper*.

The icons are displayed via a simple text file (`.desktop`), following the Freedesktop standards, you can create your own or customize using your favorite text editor. To add icons to the desktop, SliTaz

[214] http://openbox.org/wiki/Openbox:Themes

provides a tiny graphical box accessible via the menu *Preferences* → *Desktop icons* or alternatively, you can use the **Openbox** context menu *Desktop files & Icons* → *Add new icon*:



alsamixer	Alsamixer Volume mixer
alsaplayer	Alsaplayer Audio Player
asunder	Asunder CD Ripper
beaver	Beaver Code Editor
bootfloppybox	Boot floppy/cdrom creation tool
bootlog	Show Boot Log
burnbox	Burn CD/DVD
burnbox-doc	Burnbox manual
desktopbox-autostart	Auto started applications

6.1.2.8 LXPanel

LXPanel forms part of the LXDE project and handles the taskbar, menus, icons etc. Menus are dynamically generated by adding `.desktop` files to the `/usr/share/applications` or `~/.local/share/applications` directory.

The system configuration file is located in `/etc/lxpanel` and can also be stored locally in `~/.config`, though it is recommended that you configure **LXPanel** graphically by using the *Panel Settings* (right click) entry on the taskbar.

6.1.2.8.1 Panel Preferences

The **LXPanel** configurator has 4 tabs:

- *Geometry* handles the position, icons, and size — either dynamic or fixed of the panel.
- *Appearance* adjusts the background and fonts.
- *Panel Applets* lets you add, remove, edit and move plugins around on the panel.
- *Advanced* allows you to set preferred applications like the file manager, terminal and logout command.

The official website for the LXDE project and **LXPanel** can be found here[215].

6.1.2.9 Visual effects

SliTaz provides several tiny tools to give effects to the **Openbox** windows and menus. You can have transparent windows or use shadows to create depth on the Desktop. The effects are achieved via **xcompmgr** (composite manager) and **transset-df** (transparency) and both can be activated at the same time on the session using the **Openbox** context menu *Desktop Effects*.

[215] http://lxde.org/

6.1.3 Command Line Reference

author jozee, linea, seawolf, naitsirhc, genesis, hgt

6.1.3.1 Introduction to the commands

This document is intended as a quick reference for using commands on SliTaz via a Linux terminal or a graphical terminal (**xterm**). There are many GNU/Linux commands for file handling, system maintenance or network management. You can also browse the web, chat on IRC, download files, edit scripts or even play games in text mode. Note it is necessary to operate as root to assemble the hard drive or CD-ROM. You can use the command **su** to become system administrator.

6.1.3.2 Help and list available commands

Most GNU/Linux system commands have an option for providing information on their use. For support on the use of a command, it is necessary to type the command followed by the `--help` option. Example using the **cp** command to copy files:

```
$ cp --help
```

To list all the commands available on the system, you can simply press the `Tab` button on the left of the keyboard twice. For commands provided by the Busybox utility you can type:

```
$ busybox --help
```

6.1.3.3 List the files in a directory

To list the files and folders contained in a directory, you can use the **ls** command. For all options remember to use the `--help` flag. To simply list the files in the current directory:

```
$ ls
```

List all the files using the `-al` option:

```
$ ls -al
```

List a directory:

```
$ ls /home/slitaz
```

6.1.3.4 Moving around directories

To browse to the files, you can use the **cd** command:

```
$ cd /usr/share/doc
```

Back to the parent directory:

```
$ cd ..
```

To go into the directory of the user (*root* = /root):

```
$ cd
```

Or:

```
$ cd ~
```

Or:

```
$ cd $HOME
```

6.1.3.5 Copy files

The **cp** command copies files or folders. The example copies the info.txt file in the current directory into the Documents directory:

```
$ cp info.txt Documents/
```

Copy a whole directory. Here the command copies the Templates directory into /home/hacker:

```
$ cp -a Templates /home/hacker
```

6.1.3.6 Move (rename) files or directories

When source and target file are in the same file system and the target file does not exist, the **mv** command simply renames the source file:

```
$ mv file1 file2
```

It can also rename directories (provided the new directory doesn't exist):

```
$ mv ~/Documents ~/Docs
```

Move files (and directories) to a new directory:

```
$ mv file1 file2 dir1 dir2 ~/Documents
```

When renaming is not possible, the **mv** command takes the contents of a file and copies it to a new file, then deletes the original file.

6.1.3.7 Create a new file

The **touch** command can create a new empty file:

```
$ touch newfile
```

6.1.3.8 Create a new directory

This command will create a new directory. The following command creates a directory called `Projects`. It will be created in the directory `/home` of the current user or in the directory which one is in. Note you can display your current working directory with the **pwd** command:

```
$ mkdir Projects
```

Creation of a directory named `script-1.0` in the `Projects` folder:

```
$ mkdir Projects/script-1.0
```

You can also create a directory tree with the `-p` parents option:

```
$ mkdir -p one/two/three/four
```

6.1.3.9 Delete files or directories

The command **rm** lets you delete a file. Let's remove the file `work.txt` which is in the current directory:

```
$ rm work.txt
```

The command **rm** has several options. To delete a directory and its contents, we use the `-rf` option. Example:

```
$ rm -rf /home/hacker/Templates
```

Important: Be careful when using this option. It will delete everything without asking!

Note you can also use the `-i` option to remove files or directories and their contents interactively:

```
$ rm -ir /home/hacker/Templates
```

6.1.3.10 View files

To read the contents of a file or script, you can use the **less**, **more** or **cat** commands, or the web browser **Retawq**. Examples with a README file, `essential.txt`, and `script.sh`:

```
$ less -EM essential.txt
```

or:

```
$ more README
```

or:

```
$ cat /path/to/script.sh
```

Display a text or html file with the web browser **Retawq**:

```
$ retawq /usr/share/doc/index.html
```

6.1.3.11 Edit files

Editing text files, scripts, configuration files, etc, can be done easily using the text editor GNU **Nano** in a console or graphical terminal. Example with a file `bookmarks.html` (`Ctrl+X` to quit and save):

```
$ nano Public/bookmarks.html
```

6.1.3.12 `cat`

You can use the **cat** command to create various text files. `EOF` signifies *End Of File*, this is where the file ends. Example with a file `packages.list`, this removes the current contents of the file and lets you add some new text:

```
$ cat > packages.list << "EOF"
The text...
and more text

EOF
```

To append to the following text file, put two *greater than* signs (`>>`) after **cat**, example:

```
$ cat >> packages.list << "EOF"
The text...

EOF
```

6.1.3.13 Navigate the web

Surf the web quickly and simply with the **retawq** text-mode web browser. Note that you can also use the local browser. You can then navigate easily with the arrows on your keyboard — links are colored blue and can be followed by pressing `Enter`:

```
$ retawq http://www.slitaz.org/en
```

or:

```
$ retawq http://localhost/
```

6.1.3.14 Talk on IRC

To discuss and transfer files via the many IRC servers available, SliTaz provides **LostIRC**. The IRC client is simple, fast and lightweight, providing a pleasant, easy to handle GTK configuration menu. One of the main IRC channels for slitaz is `irc.freenode.net#slitaz`

6.1.3.15 Download files

To download various file formats on the internet, you have the **wget** command. To grab a simple html page, the contents of a folder or an entire website:

```
$ wget http://www.slitaz.org/en/doc/
```

6.1.3.16 List the available partitions

To list the partitions on an internal or external hard drive, you can use **cat** to display the contents of `/proc/partitions` or use the **fdisk** utility with the `-l` option meaning *list*. You can then mount the individual partition(s) that you want to use:

```
$ cat /proc/partitions
```

or:

```
# fdisk -l
```

6.1.3.17 Mount a partition, CD or USB drive

To mount a local partition in the SliTaz filesystem, we recommend you use the `/mnt` directory. Example creating the necessary directory and mounting the `hda6` partition of the first local hard drive on `/mnt/hda6`:

```
# mkdir -p /mnt/hda6
# mount -t ext3 /dev/hda6 /mnt/hda6
```

SliTaz functions in RAM, you can mount the same CD-ROM or remove it to mount another (`/dev/cdrom` is a link to the first CD-ROM drive). Note that a CD-ROM is a removable medium and should be mounted on `/media`:

```
# mount -t iso9660 /dev/cdrom /media/cdrom
```

To mount a USB or flash drive you must specify the proper filesystem. Normally a USB key is formatted in FAT32 which can be read from GNU/Linux and Windows operating systems. On a GNU/Linux system is it generally recognized as the `sda1` device — we now prepare a link `sda1` on flash to facilitate the task. Note it is also a removable medium and should be mounted on `/media`:

```
# mount -t vfat /dev/flash /media/flash
```

6.1.3.18 Turn off the system or restart

To stop or restart SliTaz, you can use the **halt** or **reboot** commands or the `Ctrl+Alt+Delete` key combination which enables a system reboot. In case of any problems you can use the `-f` option signifing forced:

```
# halt
```

To restart:

```
# reboot
```

Or:

```
# reboot -f
```

6.1.4 Hard Disk Installation

author jozee, linea, gokhlayeh, seawolf, bellard

6.1.4.1 Introduction

This document gives information and necessary instructions on how to install SliTaz on a hard disk. This should take about 10 minutes, SliTaz core LiveCD expands to 80 MB, so we suggest a minimum of 120 MB of free space. This way you will be able to install a few more packages. If you can use the LiveCD, you should be able to install SliTaz. You may also do a *frugal* (page 113) or *unusual* (page 116) install.

6.1.4.2 SliTaz Installer

SliTaz provides a simple to use Installer which can be launched from the menu on Tazpanel. SliTaz Installer messages are in English and can be used with these complimentary instructions.

Install Type

The first step lets you choose the type of installation: new install or system upgrade. In most cases you will want a new and clean installation. On confirming this the installer will mount the master CD-ROM device and search for the compressed file-system (`rootfs.gz`). If no file-system is found, then the installation will abort.

If you get into trouble because the compressed file-system is not found, please check that SliTaz is in the master CD/DVD device. If the problem persists you can use a downloaded ISO image and mount it on `/media/cdrom` where the installer expects to find it:

```
# mount -o loop slitaz-3.0.iso /media/cdrom
```

Source media

Here you can choose the type of media to install SliTaz from. Either from a LiveCD, LiveUSB, a downloaded ISO file or directly from the web.

Target Partition

Next is the partition configuration. You will need to have a partition ready; the installer does not set-up your disk for you. If you already have a free partition you can use it; if not — you will have to create one graphically using **`GParted`**, or from the command line using **`fdisk`**.

For example, if you want to install SliTaz on the second partition of the first disk recognized as `hda`:

```
/dev/hda2
```

Formatting

Warning: Formatting a partition **permanently** removes all data from it. Be sure you choose your intended partition carefully as *this action is irreversible*.

The next step lets you format the target partition. Ext3 is a robust, stable and journalled file-system. If the partition is already formatted you can skip this step, if not just accept.

Home partition

A separate home partition can be created and also formatted.

Hostname

Hostname configuration lets you set the machine name. The hostname is used internally and to identify the computer on a network. This can be changed after the system is installed. It cannot be longer than 64 characters and can only contain letters, numbers, and dashes.

Root

The root password can be configured here.

User

This allows you to configure a user name and password.

Boot-Loader (GRUB)

With the next step you have the option to install the **GRUB** boot-loader and enable a Windows dual-boot. **GRUB** is capable of booting almost any kind of operating system and can be configured through a human-readable text file; changes to this file are instant and do not require any additional commands to take effect.

If you want to use an existing **GRUB** installation, skip this step and add the correct lines to your **GRUB** configuration file (`menu.lst`); *see below* (page 289) for more information. Note that the SliTaz Installer creates a configuration file on the target which can be used as an example (`/mnt/target/boot/grub/menu.lst`).

Finishing the Installation

When the Installer has finally done its job you have the option to exit or directly reboot your new SliTaz GNU/Linux operating system. First boot is like the LiveCD, you will be prompted for options. Future reboots will not prompt you anymore for configuration details, but all the values can be changed either manually or with the project tools such as **`tazlocale`** or **`tazx`**.

6.1.4.3 Manual ('By Hand') Installation

SliTaz can also be installed 'by hand' from the command line. You can use a CD-ROM or an ISO image. The following commands can be copied/pasted from your web browser to the Terminal.

Firstly, prepare a target partition and mount it. For example, to use the second partition on the first disk drive (`/dev/hda2`), one would type:

```
# mkdir /mnt/target
# mount /dev/hda2 /mnt/target
```

Mount CD-ROM or ISO image

Mount the CD-ROM...

```
# mount /dev/cdrom /media/cdrom
```

...or if you are using an ISO image:

```
# mount -o loop slitaz-3.0.iso /media/cdrom
```

Install and Extract

With a target partition prepared and the installation media made accessible, we need to copy the files from the media into the target partition and then extract the compressed file-system (`rootfs.gz`).

- Create a boot directory and install the Linux Kernel file:

  ```
  # mkdir /mnt/target/boot
  # cp -a /media/cdrom/boot/vmlinuz-* /mnt/target/boot
  ```

- Copy the root file-system:

  ```
  # cp /media/cdrom/boot/rootfs.gz /mnt/target
  ```

  Note: Since SliTaz 4.0, multiple rootfs should be copied.

  ```
  # cp /media/cdrom/boot/rootfs* /mnt/target
  ```

  You can also copy `rootfs4.gz` only to get minimum the text mode installation

Now the necessary files are present, change (**`cd`**) to the target directory and decompress the file-system. This is done with the **`lzma`** and **`cpio`** utilities:

```
# cd /mnt/target
# unlzma < rootfs.gz | cpio -id
# rm rootfs.gz init
```

Note: For SliTaz 4.0 and newer:

```
# cd /mnt/target
# unlzma < rootfs4.gz | cpio -id
# unlzma < rootfs3.gz | cpio -id
# unlzma < rootfs2.gz | cpio -id
# unlzma < rootfs1.gz | cpio -id
# rm rootfs* init
```

That's it; SliTaz is installed! Before rebooting to start your new SliTaz GNU/Linux installation, please check that you have a boot-loader (**GRUB** or **Lilo**) installed and add the necessary lines (see below) to boot SliTaz.

6.1.4.4 GRUB Boot-Loader

GRUB is an universal boot-loader capable of booting almost any operating system, including Linux, *BSD and Windows. **GRUB** uses a single configuration file named `menu.lst`.

If you used the SliTaz Installer and installed **GRUB**, you don't need to manually install **GRUB** — just reboot.

Otherwise, to install **GRUB** onto the MBR (Master Boot Record) using a root directory of `/mnt/target` (the target mounted partition) and the disk named `hda`, use the following command and note the lack of a partition number:

```
# grub-install --root-directory=/mnt/target /dev/hda
```

You can now create a **GRUB** configuration file and add the lines which will boot SliTaz. The `menu.lst` file can be edited with your favourite text editor such as **Nano** or **Leafpad**:

```
# leafpad /mnt/target/boot/grub/menu.lst
```

Example /boot/grub/menu.lst

```
title  SliTaz GNU/Linux 3.0 (Kernel 2.6.34-slitaz)
       root(hd0,0)
       kernel /boot/vmlinuz-2.6.34-slitaz root=/dev/hda1 vga=normal
```

Verify again that everything is in place before rebooting with the **reboot** command:

```
# reboot
```

You should see **GRUB** with a SliTaz item in its menu.

6.1.4.5 Dual-Booting with Windows

A common query asked on the Community Forum[216] is how to dual-boot SliTaz and Windows. This is a straight-forward task that just needs the following lines appended to the /boot/grub/menu.lst file:

```
title   Microsoft Windows
        rootnoverify (hd0,0)
        chainloader +1
```

In this example, the Windows installation resides on the first hard disk (hd0) and the first partition (the second 0) within it. This may need modification to reflect individual cases. If it were the other way around and SliTaz proceeded Windows for instance, the line would read:

```
rootnoverify (hd0,1)
```

Most operating systems will either contain a boot-loader of their own (in the case of Windows and *BSD) or can be booted directly with **GRUB**.

6.1.4.6 Installing SliTaz on an USB Device

If you want to install SliTaz on an USB device, you must give a little rootdelay to allow time for the Linux kernel to detect it.

To include this option, edit your menu.lst to include the argument:

```
title  SliTaz GNU/Linux 3.0 (Kernel 2.6.34-slitaz)
       root(hd0,0)
       kernel /boot/vmlinuz-2.6.34-slitaz root=/dev/sda1 vga=normal␣
↪rootdelay=10
```

6.1.4.7 Sharing a partition

It is not always necessary to format a partition. You can share a partition with another OS and install SliTaz into a *loop file* (page 118) or in a *subdirectory* (page 120).

6.1.5 Accessibility

author linea, gokhlayeh

6.1.5.1 About

SliTaz aims to provide some Assistive Technology (AT) tools.

6.1.5.2 Screen magnifier

SliTaz supplies a screen magnifier whose edges can be resized and that can magnify up to 16$^{\times}$. It also provides graphical tools that enable a crosshair, pixel positions and RGB values. It can be found in the *Menu → Utilities*:

[216] http://forum.slitaz.org/

```
# tazpkg get-install magnifier
```

6.1.5.3 Espeak — Text to speech (TTS)

The **espeak** speech synthesizer run from the command line can speak text from an input file or from *stdin* and supports many languages. **Espeak** can adjust the amplitude, pitch, word gap, speed, etc. It can also write its output to a wave file rather than speaking it directly. Type **espeak --help** for a full list of available options:

```
# tazpkg get-install espeak
```

Example of use:

```
$ espeak -f example.txt
```

6.1.5.4 On-screen keyboard

Virtual keyboard (**xvkbd**) can be used to enter characters into the software of your choice. The menu can be used to change the keyboard layout, function keys, etc. It supports word completion, the removal of unwanted keys and can connect to a remote display. The configuration can be found in the `~/.Xdefaults` file:

```
# tazpkg get-install xvkbd
```

6.1.5.5 Yasr — Screen Reader

Yasr is a console (text-based) screen reader that operates through the **speech-dispatcher** interface and **espeak**:

```
# tazpkg get-install yasr
```

To configure **speech-dispatcher**:

```
# spd-conf
```

Speech-dispatcher can be started/stopped like a daemon:

```
# speech-dispatcher
# killall speech-dispatcher
```

Usage

```
$ yasr
$ exit
```

The configuration files can be found in the `/etc/speech-dispatcher` or `~/.speech-dispatcher` directories and the **yasr** configuration file is located in `/usr/share/yasr`.

6.1.5.6 Sticky/Slow/Mouse Keys in X

To enable the accessibility keys just edit your `/etc/slim.conf` file and logout of your X session:

```
default_xserver     /usr/bin/Xorg
xserver_arguments   +accessx
```

After you login again, to enable/disable:

- **Sticky keys**: Press the `Shift` key 5 times
- **Slow keys**: Hold down the `Shift` key for 8 seconds
- **Mouse keys**: Hold down the left `Shift`, left `Alt` and `Num Lock` keys

There is also a package in the repositories that can do all this in GUI or command line mode: **`accessx`**. To install and run:

```
# tazpkg get-install accessx
$ ax help
$ accessx
```

6.1.6 Web start and possible immediate installation

author oui

6.1.6.1 Special figure in SliTaz!

SliTaz is one of the extremely rare OS's available fully prepared for the web start in live mode using a very small starting program. The starting program can be saved on a old floppy disk or very little special ISO burned on a CD-ROM.

But each starting ISO of SliTaz with the live version of "base", "justX" or "stable" or "cooking" contains also this start program being used through the command line options at starting time of the ISO.

The special advantage of this kind of start is to work with the most actual cooking version and not with the version of the CD.

The same technical ability can of course be used within a closed community.

6.1.6.1.1 Where can I find the starting image for a starting floppy disk?

http://mirror.slitaz.org/boot/floppy-grub4dos

Please read more here: http://boot.slitaz.org/

6.1.6.1.2 Where can I find the small ISO for a small starting CD-ROM?

You can use the little flavor version[217] for that.

Only enter please "web" as command line option!

[217] http://mirror.slitaz.org/iso/2.0/flavors/slitaz-2.0-base.iso

6.1.6.1.3 Starting from the net

The start from net don't differ from usual start from the CD.

Only enter "web" as command line option. The starting program don't follow the CD any more but search the more actual file on the web.

6.1.6.1.4 Install SliTaz on the hard disk after the net start

Of course here is an important difference compared with the usual start:

There is no CD in the drive where the installation program can find the row files.

...

6.1.6.1.5 Regular net start in my community net (school etc.)

Install the adequate SliTaz version on the own server

...

Limit access if required

...

Prepare an adequate ISO for a own starting check card mini CD

...

6.2 Desktop Applications

6.2.1 Utilities

author jozee, linea

6.2.1.1 Galculator

`Galculator` is a simple GTK 2 based calculator that features basic, paper (cmdline) and scientific modes. You can easily switch number bases between decimal, hexadecimal, binary, etc and it supports ordinary notation/reverse polish notation. To locate from the menu, select *Utilities* → *Scientific Calculator*.

6.2.1.2 Cdrkit — Burn and manipulate CD/DVD-R or RW

To burn and handle CD/DVD-R or RW, SliTaz provides the **cdrkit** utility and a graphical interface **burnbox** which you will find in the menu. The tools in **wodim** make it possible to burn CD/DVD and erase CD/DVD-RW. When used with **genisoimage**, it can also create images in the ISO 9660 format. Burning on the command line requires us to know the device/writer (`dev`) name and **wodim** provides several possibilities to know which drive to use and specify it when burning to optical media. If you run **wodim** with the `-devices` option, it will automatically search for a good device and display it, the `-checkdrive` option allows you to check the recognized device and `-scanbus` will display in relation to the system bus. Examples (as root):

```
# wodim -devices
```

Or:

```
# wodim -checkdrive
```

For the bus:

```
# wodim -scanbus
```

Create an ISO 9660 image

To burn data on to a CD/DVD, you must first have an ISO image. To begin we must create a directory to contain all the files to be burned. You can copy your files on the command line with **cp**, the file manager **Clex** or graphically with **PCManFM2**. To create a directory named `iso/` in the root of user space and copy all the files contained in `Documents/`:

```
$ mkdir ~/iso
$ cp -a Documents/* ~/iso
```

Create an ISO image named `image.iso`, using the **genisoimage** tool and specify the root directory containing the files to be included in the ISO:

```
$ genisoimage -o image.iso ~/iso
```

Note that there are many options that you can use to create ISOs, one of the most widely used is the extension `-R`, signifying *Rock ridge*, this allows the use of names of up to 255 characters (with a few exceptions), it also supports symlinks and file permissions. To see all the available options, simply type `-help`. Example of creating an ISO image using the `-R` option:

```
$ genisoimage -o image.iso -R ~/iso
```

Burn an ISO image

To burn an ISO image in the 9660 format either created by yourself or downloaded from the web, you must use **wodim** specifying the proper device (`dev`) to burn the CD/DVD. Example using the verbose mode (`-v`), with the device `hdc` and data contained in an ISO image named `image.iso`:

```
# wodim -v dev=/dev/hdc image.iso
```

Another example setting the (`speed`) to burn as 8×:

```
# wodim -v speed=8 dev=/dev/hdc image.iso
```

Erase CD-RW/DVD-RW

You can quickly erase a CD-RW/DVD-RW using **wodim** with the `blank=fast` option. Example using the device `hdc` in verbose mode:

```
# wodim -v blank=fast dev=/dev/hdc
```

Or you can have a full erase using the `blank=all` option (this will take a bit longer):

```
# wodim -v blank=all dev=/dev/hdc
```

6.2.1.3 Nano

The default text editor in SliTaz is **nano**. Once launched you can use `Ctrl+G` for the help menu. To start **nano**, you can type **nano** from a console, a **xterm** terminal, or from the *menu* → *Editors* → *Nano*.

The initialization file `/etc/nanorc` includes the files of colored syntax found in `/usr/share/nano`. The user configuration file is `~/nanorc`. To edit a file directly, just launch **nano** proceeded by the name of the file. Example (`Ctrl+X` to save & quit):

```
$ nano Templates/script-shell.sh
```

6.2.1.4 LeafPad

Leafpad is an ultra light and quick graphical text editor, handy for taking notes or editing configuration files. You will find it in the menu or you can run it directly on a file via a terminal:

```
$ leafpad Templates/script-shell.sh
```

6.2.1.5 ISO Master — Create and edit ISOs

ISO Master is a graphical tool allowing you to edit, manipulate and create ISO images which you can later store and burn. It's simple and intuitive and lets you create ISO images the size you want.

Website: http://littlesvr.ca/isomaster/

6.2.1.6 Xpad — Mini note-taking application

Xpad is a small application that can take quick notes via a 'sticky note' displayed on the desktop. Each consists of a note pad that you can hide and customise via a right click on the window in question. Once launched you can close **Xpad** via the dock located on the window manager taskbar. The notes are stored in your local directory and are available at each session (if you use USB media with the LiveCD or on an installed system). To install **Xpad**:

```
# tazpkg get-install xpad
```

6.2.2 Office

author jozee, linea, genesis

6.2.2.1 Abiword — Word Processor

Abiword is a word processor application rich in features. It contains fast, simple, intuitive lightweight tools and proposes the proper format (`.abw`), supporting Open Office and Microsoft Word documents, it can also export to PDF or HTML. To install **Abiword** on SliTaz just run:

```
# tazpkg get-install abiword
```

6.2.2.2 Osmo — Personal Organizer

Osmo is a small personal organizer providing a timetable and a list of tasks and contacts with the possibility of opening them directly in a web browser via a URL or a mail client using an email address. **Osmo** also offers a calendar, a date calculator and the ability to take notes classified by day. If you use USB media associated with the LiveCD, it will even retain your data for you. On an installed system, you can syncronize data with USB media by using **Grsync**. **Osmo** keeps its data in the hidden folder `~/.osmo`.

6.2.2.3 SQLite — Tiny SQL Database engine

SQLite is a small relational SQL database engine whose entire database is stored in a single file. It's fast, powerful, speeds applications and implements most of the SQL92 standard. **SQLite** is ideal for managing small websites, while requiring minimal deployment. The official website for the project is: http://www.sqlite.org/

6.2.2.4 ePDFView — Lightweight PDF viewer

To view PDF documents, SliTaz uses the **epdfview** package. This provides a fast, simple, easy to use PDF viewer. **ePDFview** uses GTK+ and the rendering library **poppler**, this enables you to view, move from page to page and search or navigate the index.

6.2.2.5 Gnumeric — Spreadsheet

Gnumeric is free spreadsheet program and has the ability to import/export several file formats such as CSV, Microsoft Excel, Latex, HTML, etc. As well as providing its own format: `.gnumeric`, it is also one of the most statistically accurate spreadsheets around. To install gnumeric on SliTaz:

```
# tazpkg get-install gnumeric
```

6.2.2.6 Homebank — Finance management

Homebank is a handy program to manage and compare bank accounts. Lightweight, fast and comprehensive, **homebank** is a good addition to **Abiword** and **Gnumeric** for an office suite that's light and easy to use. Website: http://homebank.free.fr/. To install **homebank**, you can use the graphical package manager or the command:

```
# tazpkg get-install homebank
```

6.2.2.7 Notecase — Notes manager

Notecase is a software designed to organize and manage notes. It allows you to link pages, import or export notes and format text (bold, italics, etc). After installation **notecase** is located in the *Menu* → *Office* → *Notecase notes manager*.

```
# tazpkg get-install notecase
```

6.2.2.8 Wikipedia — Online Encyclopedia

Wikipedia is a free online encyclopedia where you can find information on various subjects. To open the wikipedia, choose *Office* → *Wikipedia Encyclopedia*. Main website: http://wikipedia.org/

6.2.2.9 Zoho Viewer — Online document viewer

If you need to quickly edit a document and the right spreadsheet or word processor is not at hand, you can use the document viewer to quickly upload your file (up to 10 MB). And within a couple of seconds it will be available for reading and editing. To open Zoho on the menu, select *Office* → *Zoho Document Viewer*.

It currently supports most popular formats:

- Microsoft Word (`doc`, `docx`), Excel (`xls`, `xslx`), PowerPoint (`ppt`, `pptx`)
- OpenOffice.org Writer (`odt`, `sxw`), Spreadsheet (`ods`, `sxc`), Presentation (`opd`, `sxi`)
- Also `pdf` (experimental stage), `rtf`, `html`, `txt`, and others

6.2.2.10 Office Suites

If you want a complete office suite installed at Linux, there are some nice packages at SliTaz repositories:

- Libre Office[218]
- Apache Open Office[219]

6.2.3 Internet

author jozee, linea, genesis

[218] http://www.libreoffice.org/
[219] http://www.openoffice.org/

6.2.3.1 Midori — Lightweight web browser

Midori is a lightweight web browser with very fast page rendering through the rendering engine **Webkit**. This is a serious alternative to **Firefox** for systems with low resources or those seeking a fast and sleek alternative. It supports most web standards, CSS stylesheets and images. **Midori** is configurable via a small interface and is located in the *Internet* menu once installed.

6.2.3.2 Mozilla Firefox — Web Browser

SliTaz is proud to provide Mozilla **Firefox**, one of the world's best web browsers. It is secure, fast, standards compliant and customizable via a system of plugins. To install:

```
# tazpkg get-install firefox-official
```

The web browser configuration files are stored in the hidden (dot) directory `~/.mozilla/firefox`.

Tip: When you combine the LiveCD with USB media, you can keep your bookmarks and plugins wherever you go.

6.2.3.3 Retawq — Text mode web browser

Retawq is an interactive web brower that can be run from the console or a graphical terminal. To install:

```
# tazpkg get-install retawq
```

To start **retawq**, just type (with or without the URL):

```
$ retawq http://www.slitaz.org/en
```

The configuration files are found in `~/.retawq`, you can edit them with a text editor. Pressing `b` will display the bookmarks (`bookmarks.html`) and the `h` key will display the home page.

6.2.3.4 Links — Graphical & Text browser

Links was the first graphical web browser on the SliTaz LiveCD, it has since been replaced by **Firefox**, but **Links** is always available as a package:

```
# tazpkg get-install links
```

Links offers a graphical and a text mode. To use the graphical mode, we can use the option `-g`:

```
$ links -g &
$ links -g http://www.slitaz.org/en &
$ links
```

The configuration files are stored in `~/.links`, though it's not advisable to modify them. However, **Links** provides a configuration interface via the toolbar at the top, where you can configure the languages, bookmarks, etc. When you change options, you must save them via the *menu bar* → *Configuration* → *Save options*.

6.2.3.5 LostIRC — IRC chat client

LostIRC is a simple, yet useful IRC client that supports multiple servers and automatic joining of servers/channels. The configuration files are located in `~/.lostirc`. Simply select from the *menu* → *Internet* → *LostIRC*. The documentation on the website contains a lot of useful information.

Note: SliTaz channel: irc.toile-libre.org / #slitaz

6.2.3.6 Ghost In The Mail — Email client

To send messages quickly without having to set up an email account, you can use **Ghost In The Mail** (**gitmail**). The minimal mail client offers a simple GTK interface and supports attachments. It allows you to send mail using SMTP with your existing mail account. To install gitmail:

```
# tazpkg get-install gitmail
```

6.2.3.7 Sylpheed — Mail Client

If you'd rather have a fully featured email client — you can download **Sylpheed**. Simple, reliable and easy to use, it offers powerful search and filters, and junk mail control.

```
# tazpkg get-install sylpheed
```

6.2.3.8 Transmission — Lightweight BitTorrent client

Transmission is a GTK+ BitTorrent client that is fast, light and easy to use. It offers a *Preferences* option which allows you to limit the rate of uploads/downloads, specify port, download folders, etc.

To install:

```
# tazpkg get-install transmission
```

Project website: http://transmission.m0k.org/

6.2.3.9 gFTP — FTP client

The **gFTP** application is a fast, ergonomic client for FTP transfers. It can resume interrupted transfers, manage bookmarks (favorites) and FTP or HTTP proxies. In addition **gFTP** supports the use of drag and drop, can make several transfers at the same time, compares windows, remembers passwords and can even define external applications for viewing or editing files. To install **gFTP**:

```
# tazpkg get-install gftp
```

6.2.3.10 Gtk-gnutella — P2P client

`Gtk-gnutella` is a P2P file sharing application that uses the Gnutella network. Written in C, it requires a lot less resources than other clients. It supports the use of searches and filters, features for downloading large files and bandwidth control. To install **`gtk-gnutella`**:

```
# tazpkg get-install gtk-gnutella
```

6.2.4 Graphics

author jozee, linea, genesis, hgt

6.2.4.1 `Gcolor2` — Select and manage colors

`Gcolor2` is a tool to select and retain palette colors. It can be useful for the creation of SliTaz themes, for example. It can be found in the *Graphics* category or run from the command line:

```
$ gcolor2 &
```

6.2.4.2 `GIMP` — Manipulate and create images

The **`GIMP`** (GNU Image Manipulation Program) is software that can manipulate images to a very high quality level. It allows you to do what you would expect from an application that processes images, ie. layers, filters, support scripts adding functionality, etc. **`GIMP`** supports a large number of image formats such as: PNG, JPEG, XPM, PPM, TIFF, PostScript, PSD, it also offers its own XCF format. To install **`GIMP`**:

```
# tazpkg get-install gimp
```

`GIMP` is scalable and can be configured with the main interface — configuration files, brushes and personal scripts are located in the `~/.gimp-2.2` directory.

6.2.4.3 `GQview` — Image manager

`GQview` is very light and quick and allows you to navigate rapidly between images by selecting files in a directory tree with a single mouse click. It supports slideshows, image rotation, adding keywords and tags, drag and drop, and can display EXIF data. It also allows you to edit images in the software of your choice (**`mtPaint`**, **`GIMP`** for example). To install **`GQview`**:

```
# tazpkg get-install gqview
```

6.2.4.4 `jpeg` — JPEG command line tools

To allow applications that use JPEG to function, linked libraries must be provided by the package **`jpeg-6b`**, this package also contains some tiny utilities that can be used on the command line such as **`cjpeg`** and **`djpeg`**. To modify JPEG images on the command line you can also use **`jpegtran`**, installed by default on SliTaz; **`jpegtran`** allows you to rotate images via the `-rotate` option. To find out all of the options available for these tools, just specify the `--help` option. Example:

```
$ cjpeg --help
```

6.2.4.5 `mtPaint` — Image processing

`mtPaint` is an application for the creation and retouching of PNG, TIFF, XPM and BMP images. It offers many simple, lightweight, fast functions like capture screen (screenshot) which you can access from the *menu* → *Graphics* → *Grab screenshot*, or via a terminal:

```
$ mtpaint -s
```

6.2.4.6 `Viewnior` — Elegant image viewer

`Viewnior` is a fast and simple image viewer with a minimalistic interface. It can rotate, flip, crop, save, delete images and supports fullscreen, slideshow, etc. To install:

```
# tazpkg get-install viewnior
```

6.2.5 Development

author jozee, linea, genesis, hgt

6.2.5.1 About Development

SliTaz provides development tools for web design, editing scripts and source code. On the website, the Development page will give you general information about the developers and opportunities for involvement.

6.2.5.2 SHell scripts

Writing SHell scripts is the easiest way to start coding, they can provide quick results and the only prerequisites are being able to open a terminal and using a text editor such as **`Nano`**, **`Leafpad`** or **`Geany`**. SHell scripts can do many things on a GNU/Linux system — initialize the system, make backups, perform repetitive tasks, display system information, create or modify files and so on. In a SHell script you can use variables, functions or calls to include a file. Note that you can name your script as you see fit and the `.sh` extension is widely used.

Create a SHell script

Before starting a new SHell script, you must pay attention to the interpreter used. Most SHell scripts use `/bin/sh`, because it's more portable, but there are scripts that rely on `/bin/bash` and this must be installed on the system. For a SHell script to function it must be made executable by changing permissions on the command line using the **`chmod`** tool. To create a `script.sh` and make it executable:

```
$ touch script.sh
$ chmod +x script.sh
```

Now that you have a new executable file, you can edit it. You can continue to stay in the terminal and use the **Nano** editor (`Ctrl+X` to save & exit) or IDE **Beaver** to edit:

```
$ nano script.sh
```

Or:

```
$ beaver script.sh &
```

Here's a script that contains a variable `NAME` and displays the value with the **echo** command:

```
#!/bin/sh

NAME="kayam"

echo "$NAME is nice."
```

Once you have created/modified your `script.sh`, you can execute it to see the result:

```
$ ./script.sh
```

So much for this brief introduction to SHell scripts. The Web is full of information if you wish to explore further.

6.2.5.3 Dialog

Dialog can create GUI-based consoles such as the SliTaz 'installer'. The configuration files are `/etc/dialogrc` and/or `~/dialogrc` for each user. Here's a simple example of using **dialog** via a console or terminal:

```
$ dialog --title "Hello $USER" \
  --msgbox "Message made by dialog." 5 54
```

You can find plenty of example scripts in the `/sample` directory inside the source code of **dialog** which can be downloaded from: http://invisible-island.net/dialog/dialog.html. Download sources and decompress:

```
$ wget ftp://invisible-island.net/dialog/dialog.tar.gz
$ tar xzf dialog.tar.gz
```

6.2.5.4 Beaver

Beaver is a simple, lightweight and fast code editor offering syntax. **Beaver** can be found in the *Menu → Development → Code Editor*. Once launched for the first time you can adjust your preferences through the *Edit* button. You can run it through a X terminal with the following command:

```
$ beaver &
```

6.2.5.5 Geany IDE

Geany is an IDE, or Integrated Development Environment. It is a simple, small and lightweight application, offering syntax, tabs and self-completion.

Executing Geany

You can install **Geany**:

```
# tazpkg get-install geany
```

After it runs for the first time, you can set your preferences through the menu *Edit* → *Preferences* option. It is also possible to run **Geany** via terminal:

```
$ geany &
```

Note: When compiling source code, the `./configure` script offers the `-enable-the-force` option, which you could use if you feel the need to became a Jedi warrior!

6.2.5.6 Perl or Microperl — Code/use Perl scripts

On SliTaz you can use the powerful scripting language **Perl** via the `perl` or `microperl` binary. **Microperl** is a streamlined version of **perl** compiled from official sources, **Perl** scripts running **Microperl** are compatible with the complete version of **Perl**. One of **Perl**'s strengths is its portability, it can be used on any system and it's an interpreted language, which means that the code doesn't need to be compiled and can be used directly. On SliTaz, **Perl** and **Microperl** are not installed by default on the LiveCD: you can either rebuild your ISO or install through the package manager.

Note: **Microperl** is only 1 MB and provides no modules:

```
# tazpkg install perl
```

Or:

```
# tazpkg install microperl
```

Hello world!

The purpose of this script is to display *Hello World*. You can start by creating the file and making it executable on the command line and then editing with **Beaver**. Note the script is called `hello.pl`, but you can name it as you see fit, with or without the `.pl` extension:

```
$ touch hello.pl
$ chmod +x hello.pl
$ beaver hello.pl &
```

The first line of a **Perl** script begins by defining the path to the **Perl** interpreter, usually `/usr/bin/perl` and to display text, just use the **print** command. It should be noted that **Perl** is case sensitive and a line of code should always end with a semicolon. Example code (you can copy and paste):

```
#!/usr/bin/perl
#
```

(continues on next page)

(continued from previous page)

```
print "Hello World!\n";
```

To execute and test the script:

```
$ ./hello.pl
```

CGI Scripts and Perl

CGI scripts are designed to display dynamically generated web pages. The **Perl** language associated with the **LightTPD** web server allows you to use CGI scripts through your public space or via virtual hosts. **Perl** is quite adapted to Web 2.0 and can generate xHTML pages. On SliTaz you must have **Perl** or **Microperl** installed and the **LightTPD** server configured before you can use CGI scripts coded in **Perl**. Note that by default SHell scripts (`.sh`) can be placed in `/cgi-bin/`.

Once the server is properly configured, you can put your CGI in your `$HOME/Public/cgi-bin` using the `.pl` or `.cgi` extension and view them either locally or remotely. Example of using a **Perl** CGI script:

```
#!/usr/bin/perl
#
print "content-type : text/html\n\n";

print "Hello World!\n";
```

6.2.5.7 Python

The **Python** programming language is available as an installable package. Once installed, you can create your own scripts/programs and use CGI applications with the **LightTPD** web server, taking care to configure the server properly. The official SliTaz Mercurial repositories are provided by a CGI/Python web interface — a solution suited to a product that's reliable and robust. To install the **python** package with **tazpkg**:

```
# tazpkg get-install python
```

6.2.5.8 Ruby

The **Ruby** programming language is available as an installable package. **Ruby** is (to quote the official website):

> "A dynamic, open source programming language with a focus on simplicity and productivity. It has an elegant syntax that is natural to read and easy to write".

Ruby handles exceptions, supports Object-Orientated Programming (OOP), automatic memory management and is highly portable. To install the **ruby** package with **tazpkg**:

```
# tazpkg get-install ruby
```

6.2.5.9 Toolchain — Libraries, C compiler and tools

To compile software from sources or your own code, you need at least the basic *toolchain*, comprising of **Binutils**, **Glibc**, C compiler, Kernel headers and the **Make** utility. Note that the *toolchain* is used by the SliTaz developers to compile the entire system from source. To install the meta package and all dependencies:

```
# tazpkg get-install slitaz-toolchain
```

The installation of the *toolchain* can now compile basic applications in console mode without a problem using the **Busybox** Ash SHell, but some other packages will not compile without **Bash**. GNU **Bash** is available as a package along with various other development tools such as **Flex**, **M4**, **Bison** or **Pkg-config**. If you are looking for **pkg-config** for example:

```
$ tazpkg search pkg-config
```

If you would like to compile applications utilizing the **Ncurses** library, you must install the **ncurses-dev** package. Note the **ncurses** package also provides a variety of small programs such as **tic** or **tac**:

```
$ tazpkg search ncurses
```

6.2.6 Multimedia

author jozee, linea, genesis

6.2.6.1 SliTaz sound and video

SliTaz LiveCD mode provides sound support, but no video player, however you can watch YouTube by installing the **get-flash-plugin** package. To watch videos you can use **Xine**, it supports most formats such as avi, mov, mpeg, etc. Note **Xine** can also act as an audio player. At the sound level, you have a mixer (**alsamixer**) and audio player (**alsaplayer**), both installed by default on the standard LiveCD. SliTaz also provides tools to configure your sound card.

6.2.6.2 Configure the sound card

In LiveCD mode SliTaz automatically configures the sound card by launching a dialog. In most cases the sound card is instantly recognized and you just press `Enter` to continue to login and have your sound working. However, sometimes your card is not recognized at startup or simply not supported. Before throwing in the towel, you can try to manually configure your card by asking for help on the mailing list or on the hardware support forum.

Soundconf

To configure or reconfigure a sound card, SliTaz provides a script (**soundconf**), this utility is used at startup and doesn't pose any questions, supports the right modules and automatically configures `/etc/rsS.conf` so that the right driver is loaded on each start. **Soundconf** will also automatically adjust volume levels, you can change the settings later through the mixer and **alsactl**. To start the sound configuration interface, you must have administrator priviledges and type:

```
# soundconf
```

Alsactl

Alsactl controls ALSA and can store settings or restore sound for example, to retain preferences for each start. When configuring sound with **soundconf**, volumes are automatically adjusted. You can use **alsamixer** as root to change the values and then launch **alsactl** to store the configuration.

First, start **alsactl**:

```
# alsactl init
```

Then use **alsamixer** to set up sound levels according to your preferences (you can find them at menu *Applications* → *Multimedia*). Lastly, use **alsactl** again to store your preferences:

```
# alsactl store
```

To restore the configuration you can use the **alsactl restore** command or to restore your preferences at each system startup you can edit `/etc/init.d/local.sh` with your favorite text editor or simply **echo** the changes:

```
# echo 'alsactl restore' > /etc/init.d/local.sh
```

6.2.6.3 Alsamixer

Alsamixer is the official mixer of the ALSA project. It is simple and effective and can be run from the menu or directly from a terminal. The volumes can be regulated with the up/down arrows or muted using the `M` key. To start from a Linux terminal:

```
$ alsamixer
```

6.2.6.4 Alsaplayer

The audio player **Alsaplayer** is designed to be simple, clean and intuitive. **Alsaplayer** on SliTaz provides support for ogg, mp3 (via *libmad*) and wav files, playlists and viewers to accompany sound and the ability to adjust pitch and volume. Once launched from the menu, just click the play button or load a playlist to start.

6.2.6.5 Asunder — Audio CD Ripper

Asunder retrieves and encodes audio tracks from a CD. The toolkit is simple and easy to use, encoding songs into wav, ogg or mp3 (via *lame* package). It can search CDDB, create playlists and edit file names. You will find **Asunder** in the *Multimedia* category on the menu.

6.2.6.6 mhWaveEdit — Audio editor and recorder

mhWaveEdit is a graphical application for playing, recording and editing sound files. Supported formats: wav (default), and a few others. A variety of other formats can also be imported through **Mplayer**. To start from the menu, select *Multimedia* → *mhWaveEdit*.

6.2.6.7 MPD — Music Player Daemon

MPD is a great little music player that uses the server/client architecture, this means it can be even run remotely without a X server operating. To install **MPD** on SliTaz:

```
# tazpkg get-install mpd
```

MPD on SliTaz drops root priviledges, so to start the server, just run:

```
$ mpd
```

And to stop it:

```
$ mpd --kill
```

Simply drag your music files into the `~/music` folder (or create a link) and then run **mpd --create-db** as root to update the database and you're ready to use one of the many clients. The **mpd --version** command lets you have a full list of available formats and outputs.

6.2.6.8 MPC — Music Player Client

MPC is a popular client for **MPD**, to load all the files into **MPC** from the **MPD** database and start to play them, simply install and run:

```
# tazpkg get-install mpc
$ mpc add /
$ mpc play
```

The command **mpc --help** gives you a full list of all the available options that allow you to edit the playlist, enable crossfade, adjust the volume and shuffle tracks, etc.

6.2.6.9 mpg123

mpg123 is a command line audio player and file converter, this means that you can listen to music or convert files from a terminal. To install:

```
# tazpkg get-install mpg123
```

To display help, use the `--help` option. To play a mp3 file, just launch **mpg123** followed by the name of the audio file:

```
$ mpg123 sound.mp3
```

mpg123 can also encode a file into another format, for example you can convert a *wav* file into a mp3 file. Example:

```
$ mpg123 -w sound.mp3 sound.wav
```

6.2.6.10 Mplayer

Mplayer is a popular movie player for Linux suppporting many formats including DVD, VCD, mpeg, wmv, realvideo, etc. It can also play various audio codecs such as aac, wma, realaudio, as well as ogg, flac, etc. **Mplayer** is configurable via a right click menu and customizable using various skins and GUIs that easily enable you to configure your own video drivers, output devices and so on. To install **mplayer**:

```
# tazpkg get-install mplayer-svn
```

6.2.6.11 Xine

Xine is a multimedia project providing various video viewers and audio players. SliTaz provides libraries and a media player contained in the package **xine-ui**. **Xine** uses a *Xlib* interface, a control panel, a right click configuration menu and various plugins. It can play ogg, mp3 and flac audio codecs, and mov, avi or mpg video formats. To install **xine** and its dependencies:

```
# tazpkg get-install xine-ui
```

Official Xine website[220]

6.2.6.12 VLC

VLC media player is a highly portable multimedia player and multimedia framework capable of reading most audio and video formats (MPEG-2, MPEG-4, H.264, DivX, MPEG-1, mp3, ogg, aac…) as well as DVDs, Audio CDs VCDs, and various streaming protocols. To install **vlc**:

```
# tazpkg get-install vlc
```

Official VLC website[221]

6.2.7 System Tools

author jozee, linea, genesis

6.2.7.1 Clex — Command line File Manager

To navigate through your folders and directories you can use **cd** on the command line or install the **Clex** File Manager:

```
# tazpkg get-install clex
```

Using **ncurses**, **clex** is fast and easy to use and can be configured through the files `~/.clexrc` and `~/.clexbm` (bookmarks) or via the panel (`Ctrl+G`). To start **clex** from a terminal or console:

```
$ clex
```

220 http://xinehq.de/index.php/home
221 http://www.videolan.org/vlc/

6.2.7.2 `PCManFM` — File Manager

`PCManFM` is a file manager providing many useful functions for daily tasks such as managing devices, opening terminals in the current directory, tabbed browsing, drag and drop, creating directories or managing file permissions. It contains bookmarks to allow you to browse faster, search functions and much more. PCManFM can be launched with some command line options — you can set the wallpaper to display or open folders in new tabs, etc. For a full list of options:

```
$ pcmanfm --help-all
```

`PCManFM` supports hotkeys (firefox) and the context menu (right click on file/directory) makes it easy to unpack `.tar.gz` archives, compress and create archives.

6.2.7.3 `Htop` — View system processes

`Htop` is a system process viewer that displays CPU load, memory state (RAM) and swap used. It can also display the number of tasks, uptime and PIDs of active processes. **`Htop`** can be used with the keyboard in console mode, the mouse with a X terminal (**`xterm`**) and provides configuration options (`F2`). **`Htop`** can also kill processes — you can select items with the up/down arrows or a mouse click. Note **`htop`** also functions via SSH and can be used to monitor a remote server:

```
# tazpkg get-install htop
```

6.2.7.4 `LXTask` — Graphical system process viewer

`LXTask` is the default process viewer in SliTaz. It offers the same functionality as **`Htop`**, except for the ability to control it remotely. You can start it from the *System Tools Menu → Task Manager*.

6.2.7.5 `Mountbox` — Mount devices

`Mountbox` is a small GTK+ application to quickly mount media such as a USB drive, hard drive or CD-ROM. **`Mountbox`** can be launched from a terminal or via the *Tools* menu (*System Tools*). Simply specify the peripheral (*Device*) and the mount point, i.e. the directory where you want to access the media in question. Typically a CD is mounted on `/media/cdrom`, a USB key on `/media/flash` and disk drives on the local machine on `/mnt`. Note the Handbook also contains more information.

6.2.7.6 `Gparted` — Partition a hard drive

`Gparted` is a graphical application making it possible to manage the partitions of a local hard drive or USB media. It allows you to reformat, resize or check a partition on a hard drive and is the tool of choice if you need to prepare a partition to install SliTaz. **`Gparted`** supports proper GNU/Linux filesystems (ext2, ext3 and ext4) via **`mkfs`**, and **`Parted`** automatically handles dependencies.

Support FAT and NTFS filesystems

To have the support of FAT16 or Windows FAT32 filesystems, you must install the package **`dosfstools`**. To enable read/write support for NTFS partitions: **`fuse`**, **`ntfs-3g`** and **`ntfsprogs`**.

6.2.8 Window Managers

author jozee, linea, seawolf, hgt

6.2.8.1 Openbox

More details in *Desktop* (page 278)

6.2.8.2 Enlightenment (e17)

Enlightenment is a complete desktop environment, fully configurable with the mouse and offering many themes. The version supplied by the SliTaz project is known as **e17** and is still in development, this version is considered stable enough to be incorporated into the distribution. **Enlightenment** was designed to be deployed on systems with limited resources, as well as more powerful systems. It allows for wallpapers, menus, animated and interactive gadgets and knows how to manage virtual desktops. To install **e17**:

```
# tazpkg get-install enlightenment
```

Logout your current X session, type `F1` at SLiM login and choose `e17` to start **Enlightenment**.

Menu and desktop icons

Enlightenment is compliant to the Freedesktop standards. Applications are sorted by category and icons automatically appear if a desktop file is supplied. The (`.desktop`) system files are contained in the `/usr/share/applications` directory or hidden home `~/.local/share/applications` directory. These files have a simple syntax and are editable from the panel or with a simple text editor. The file menus displaying categories in the **Enlightenment** menu are found in the `slitaz-menus` package:

```
# tazpkg get-install slitaz-menus
```

An icon on the desktop

To have an icon on the desktop launching applications, you can create by hand a `.desktop` file (Recognized desktop entry keys[222]) in your local directory `~/Desktop`. Desktop files placed in this directory are automatically recognized by **Enlightenment**. A single `.desktop` file can contain eight lines with respectively: the name (Name), generic name, comment, the command to execute (Exec), icon, type and Freedesktop categories. Example of a `.desktop` file for **Xterm** icon:

```
[Desktop Entry]
Name = XTerm
GenericName = Terminal
Comment = Run commands in a shell
Exec = xterm -bg black -fg white -cr orange -sb -bd violet -rightbar
Icon = /usr/share/icons/Tango/jwm/utilities-terminal.png
Type = Application
Categories = Utility;Terminal;
```

[222] http://standards.freedesktop.org/desktop-entry-spec/latest/ar01s05.html

Additional themes can be found on: http://exchange.enlightenment.org/

6.2.8.3 JWM — Joe's Window Manager

Joe's window manager, written in C is quick, simple, clean, stable and efficient. **JWM** proposes a taskbar, a menu of icons and a pager for the management of virtual desktops. The taskbar can also act as a dock. In addition it is easily configurable with a single text file that can change the menu, fonts and their sizes, and different colors. To install **jwm** on SliTaz:

```
# tazpkg get-install jwm
```

Logout your current X session, type `F1` at SLiM login and choose *jwm* to start **JWM**. To make **JWM** your default Window Manager, just type: **tazx jwm**.

Use and configure JWM

The application of Joe's Window Manager is very fast. To view the menu just click somewhere on the desktop. You can resize a window through the edges or corners, minimize or pass a virtual desktop to another via a pager. You also have configurable keyboard shortcuts for faster access to the applications that you often use. On SliTaz the system configuration file is `/etc/jwm/system.jwmrc`. Apart from this file, each user can use its own configuration file hidden in `~/.jwmrc`. This is a text file using XML syntax, it can edited with a simple text editor — lines beginning with: `<!--` are comments that let you understand what each tag does.

To facilitate the customization of the desktop, SliTaz automatically copies at the launch of the first (graphical) session, a system configuration file to the root directory of the user. You can directly modify this file and test without risk. To edit with your favorite text editor:

```
$ geany $HOME/.jwmrc &
```

To retrieve an original configuration file, you can copy the system configuration file and rename it `.jwmrc` in your home directory:

```
$ cp /etc/jwm/system.jwmrc $HOME/.jwmrc
```

The tag `RootMenu` corresponds to the menu displayed by clicking on one of the (three) buttons on the mouse. To add a category, you must use the tag `Menu` — this contains entries for various programs. Any entry in the **JWM** menu can fit on one line. Example using the **GQview** image management application:

```
<Program icon="gqview.png" label="GQview">gqview</Program>
```

There are still many opportunities to configure RootMenu according to the mouse buttons; the choice of method to move windows, create groups, etc. The Manual is available online at the official website of the project. To view a list of command-line options, just type **jwm -h** in a terminal.

Create your own JWM style

Creating your own graphical style with **JWM** is relatively quick and easy, the tags are clear and the attributes possible are given in the comments. When preparing your work, you can see your amendments by restarting the window manager from the menu or via the **jwm -restart** command. In the configuration file, style tags start after the `<!-- Visual Styles -->` comment. To begin, here is a short list of the main style tags with a short description:

- `Background` manages the wallpaper. This tag supports the solid, gradient, image or tile attributes, to respectively: use a solid color, create a gradient, display a resized image or tile an image.
- `BorderStyle` controls the windows border.
- `TrayStyle` controls a taskbar. The taskbar may, among other things, be automatically hidden or only fill a part of the screen with the width attribute.
- `TrayListStyle` controls the style of the list of open windows on the current desktop.
- `PagerStyle` controls the pager displaying different virtual desktops (4 by default).
- `MenuStyle` defines the menu style.
- The icons are defined by the `IconPath` tag, you can use your own personal icons by specifing the full path to the directory that contains them. Note that you can specify more than one path, if you want, you can use your own icons and those contained in the `/usr/share/pixmaps` and `/usr/share/icons` system directories. SliTaz uses the Tango theme icons: *tango.freedesktop.org* for the menu, these are 16×16 and are stored in `/usr/share/icons/Tango`. You can add, edit, delete these… If you want to install new icons in your user space, we advise you to use `~/Picture/Icons` (set as default) or a hidden directory `~/.Icons`.

The colors can be defined by their name or RGB number such as `#3A4956`. To use colors in their gradient mode, you must specify the two colors separated by a colon, example `#6C0023:#3E1220`. You can change fonts and their sizes by using the `Font` tag. There are still some other small things that you can change to customize your desktop: such as the name of a menu item and its icon. Before restarting **JWM** with your new configuration file, you can check its syntax by using the command: **jwm -p**. To explore further, the official handbook describes all the tags, options and valid attributes. You can view it online at the **JWM** website.

JWM website

- http://www.joewing.net/programs/jwm/ — The official website of Joe's Window Manager, providing news and a comprehensive manual.
- #jwm on irc.freenode.net — The **JWM** IRC discussion channel on Freenode server.

6.2.8.4 Pekwm

Pekwm Documentation[223]

6.2.8.5 DWM

DWM Documentation[224]

6.2.8.6 Xfce

Xfce[225] is a lightweight desktop environment. It replaces the default **Openbox** and **PCManFM**, and is also based on GTK+.

[223] https://www.pekwm.org/doc/git/index.html
[224] http://dwm.suckless.org/tutorial
[225] http://www.xfce.org/

To install **Xfce**, select the 'xfce4' meta-package from the Package Manager. This will install all the related packages.

To use **Xfce**, ensure you have the correct command for **Xfce** in the SLiM (log-in manager) configuration. Do this by appending `xfce4` to the `sessions` line of the `/etc/slim.conf` file — note that you will need root permissions to modify the file:

```
sessions            openbox,e17,jwm,xfce4,
```

You can then select **Xfce** by pressing `F1` at the log-in screen, as you enter your user-name and password.

Extras are available at Xfce Goodies[226], including plug-ins, artwork and bindings.

To remove **Xfce**, use the following command as *root* user:

```
for PKG in xfce4 xfce4-session xfce4-panel xfwm4 libxfcegui4 xfce-utils \
           libxfce4util thunar thunar-volman xfconf; do
  yes y | tazpkg remove $PKG
done
```

6.3 System Administration

6.3.1 Packages

author jozee, linea, seawolf, genesis

6.3.1.1 Tazpkg — Package manager

SliTaz provides a tiny package manager which can easily install more software on the system. **Tazpkg** is a lightweight package manager for `.tazpkg` files. Completely written in SHell script, it works well with Busybox ash shell and **BASH**. **Tazpkg** lets you list, install, remove, download, extract, pack, search, or get information about available or installed packages. You can also repack an installed package and automatically upgrade all installed packages. On SliTaz you can type **`tazpkg usage`** in a terminal to get a list of all the commands with a short description in English.

List of packages

Tazpkg lets you list all installed packages, installed packages by category or it can display the list of available packages on the mirror. To display a single list of all installed packages on the system, just type:

```
$ tazpkg list
```

To display all categories or packages sorted by category, you must specify `cat` or category. Examples:

```
$ tazpkg list cat
$ tazpkg list base-system
```

Tazpkg can also generate a xHTML list (default: `installed-packages.html`) in the local directory of all installed packages on the system:

[226] http://goodies.xfce.org/

```
$ tazpkg xhtml-list
```

To get a single list of all available packages on the mirror you can use the command **`list-mirror`**. You can then examine the list in your favorite editor or use the Web site interface.

Install packages

To install some new applications such as **`GIMP`**, **`AbiWord`**, **`ePDFView`**, **`Perl`** or **`Python`**, you first need to recharge the list of available packages on the mirror and then install. If the package dependencies are not installed, **`Tazpkg`** will install them for you. For example, the installation of **`Gparted`** (a GTK+ partition editor using GNU **`parted`**):

```
# tazpkg recharge
# tazpkg get-install gparted
```

The 'get' Packages

There are a few packages that are prefixed with `get-`. These are not packages per-sé but contain only a script. This script provides binary software by:

- downloading the program
- creating a Tazpkg from it
- installing the generated Tazpkg

This means that the actual program is not contained within the `get` package, but that generated by it.

To manage this style of software:

- install the latest version using the `get` script in the `get` package;
- remove it by using **`tazpkg`** on the **generated** package.

Tip: Please note that after downloading the `get-package`, you need to run the script that has the same name:

```
# tazpkg get-install get-OpenOffice3
# get-OpenOffice3
```

Upgrade installed packages

To keep your system up-to-date and secure after recharging the packages list, you can automatically install all new versions and security updates from the mirror with the command:

```
# tazpkg up
```

Tazpkg Manual

The **Tazpkg** Manual[227] contains a lot more useful information.

6.3.1.2 Cookutils & the wok

All SliTaz packages are built with a tool named **Cookutils** and a receipt found in the *wok*. The receipt provides all the necessary information to build a suitable package for **Tazpkg** including variables to give us the package name, source tarball format, download URL, etc. Given a receipt, the `compile_rules` function has all of the necessary commands to configure, make, and install the package in a specific directory. After compilation, **Cookutils** will execute the function `genpkg_rules` to pick up only the needed/wanted files and generate a *pkg*`.tazpkg` (cpio archive). On SliTaz you will find all installed package receipts in the directory `/var/lib/tazpkg/installed`, feel free to examine them or even use one as an example.

Cookutils will search by default for a *wok* in `/home/slitaz/wok` and put generated packages in `/home/slitaz/packages`. These paths are set by a **Cookutils** configuration file which can be located in `/etc/slitaz/cook.conf` or in the current directory, which is useful if you want to work with multiple *woks*. Now, if the **Cookutils** are setup (**# cook setup**) and the slitaz-toolchain is installed, you can start to create and build a package which doesn't need many dependencies. Small example:

```
# cook new pkgname --interactive
```

When a new package tree and receipt has been created in the *wok*, you can edit the receipt with your favorite editor (**Geany** provides nicely colored code), modify the rules, functions, add dependencies to the `DEPENDS` variable if needed and try a first cook:

```
# cook pkgname
```

Note that you can now browse the generated files, modify the cooking receipt again or just rebuild the package. When you are happy with your work you can install the package with **tazpkg install** and then test the application or library.

6.3.2 Network Configuration

author jozee, linea, domcox, emgi, genesis, mojo

Important: SliTaz Panel/Network replaces **netbox** for SliTaz-4.0 and newer

6.3.2.1 About the Network

By default SliTaz starts a DHCP client (**udhcpc**) on `eth0` at boot time. If your network card has been identified as an `eth0` interface and you use a router, your connection should already be working. DHCP is dynamically configured, on each boot the client asks for a new IP address from the DHCP server which is integrated into the router, or on another computer. If you need a static IP, you can directly edit config files or use the *Network* tab at *System Tools* → *Slitaz Panel*. In a terminal or a Linux console, you can list all available network interfaces with the command **ifconfig** followed by the `-a` option:

[227] http://hg.slitaz.org/tazpkg/raw-file/tip/doc/tazpkg.en.html

```
$ ifconfig -a
```

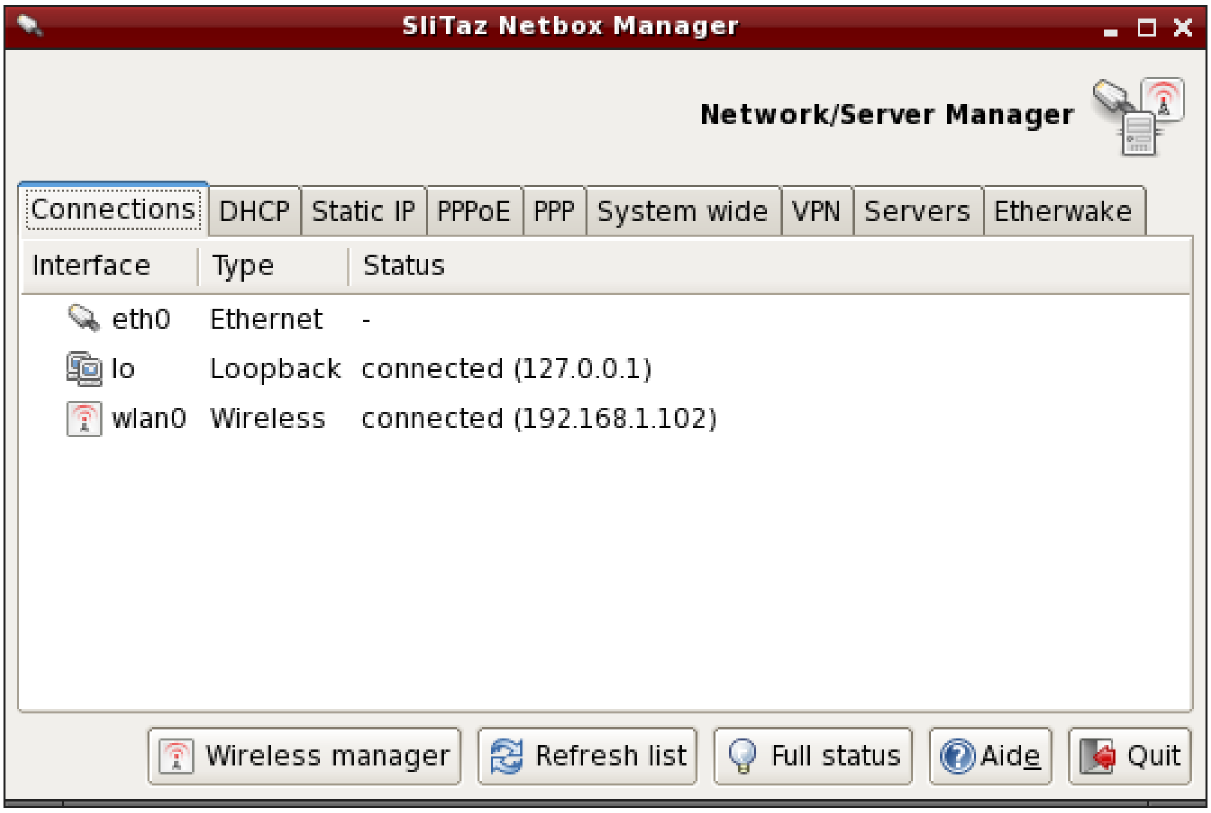
To display the Kernel's IP routing table, you can use the **route** command without any arguments:

```
$ route
```

The system wide network configuration file is `/etc/network.conf`. It can be graphically configured at the *Network* tab on SliTaz Panel or directly edited by the root administrator.

6.3.2.2 Netbox — Configure the network (SliTaz 3.0 and older)

Netbox is a small GTK+ application to configure a network interface using DCHP or a fixed (static) IP address. The tabs can be used to start/stop the connections and automatically change the values in the system files. **Netbox** provides a system wide tab from which you can directly edit network configuration files, and tabs to configure PPP/PPPoE username/passwords. Servers such as SSH, DHCP, PXE, DNS, etc can also be configured and it's possible to create your own virtual private network (VPN) using the tools provided.



```
Interface  Type       Status
eth0       Ethernet   -
lo         Loopback   connected (127.0.0.1)
wlan0      Wireless   connected (192.168.1.102)
```

You can start **netbox** from the *System tools* menu or via a terminal:

```
$ subox netbox
```

Important: SliTaz-4.0 and newer:

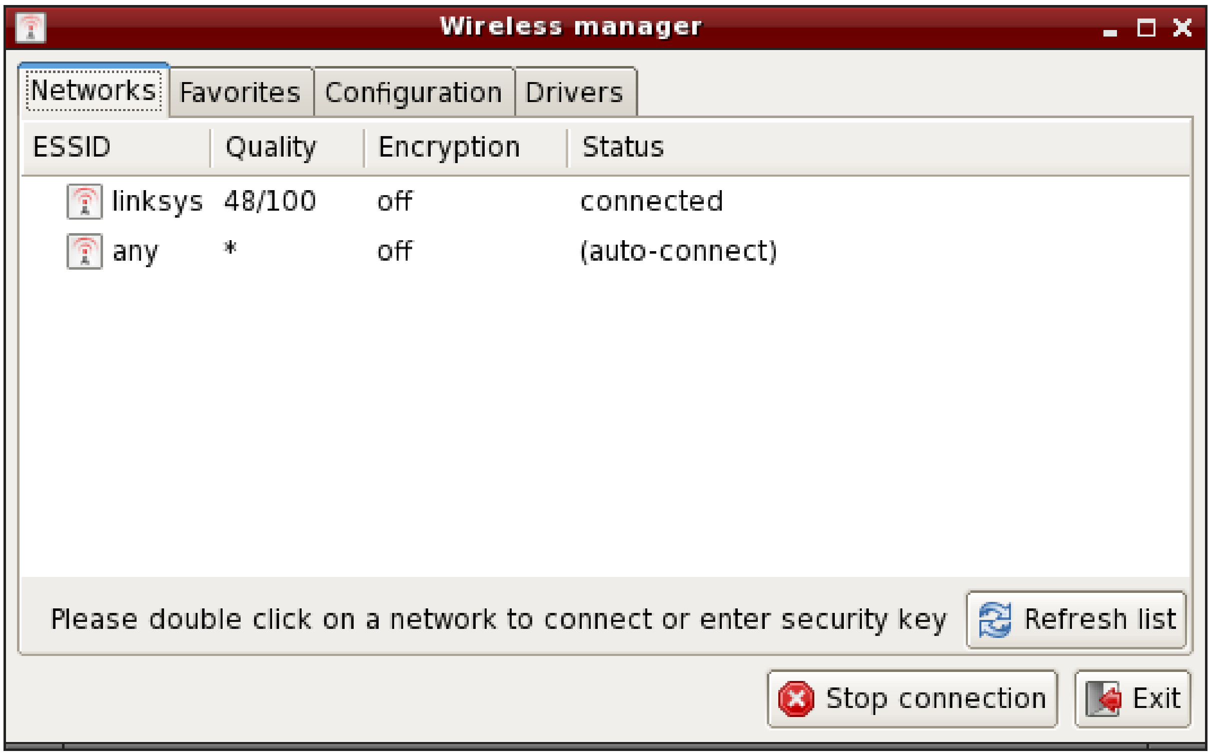
wifi-box was renamed to **wifibox**

SliTaz Panel → *Network* → *Wireless* offers same configuration

SliTaz-5.0 offers **slitaz-config** Wi-Fi configuration using **ncurses** interface.

6.3.2.3 Wifi-box — Graphical configuration of the wireless network (SliTaz 4.0 and 5.0 Weekly)

Wifi-box is small interface to configure a network connection (Wi-Fi, WLAN, or Wireless). The *Networks* tab displays a list of available networks, just double click on a network name to connect. If the network is secure, the key will then be sought.



ESSID	Quality	Encryption	Status
linksys	48/100	off	connected
any	*	off	(auto-connect)

Please double click on a network to connect or enter security key

The *Favorites* tab allows you to set your preferred networks. Once a network is added, just double click on the network name to connect. The *Configuration* tab lets you configure a connection manually using the advanced settings such as the mode or channel. The *Drivers* tab allows you to configure a network card; there are 3 options:

- The card is supported directly by the kernel via a module.
- The card needs a module and non-free firmware that can be installed automatically via the auto-detect tool (**tazhw**).
- The card is not supported by Linux and a Windows driver must be installed via the Windows driver manager (**tazndis**).

You can start **wifi-box** via a terminal:

```
# wifi-box
```

6.3.2.4 `/etc/hostname` — The hostname

The file `/etc/hostname` sets the machine name. This is loaded at system startup with the command **hostname**, without an argument this command returns the current machine name:

```
$ hostname
```

To change the hostname, you can use the **echo** command or a text editor available on SliTaz (you must be root). Example using **echo** and the machine name *kayam*:

```
# echo "kayam" > /etc/hostname
```

6.3.2.5 `/etc/network.conf`

`/etc/network.conf` is the SliTaz system network configuration file. It's syntax is simple and you can edit its contents with a text editor such as **Nano**. `/etc/network.conf` is used by the script `/etc/init.d/network.sh` to configure the network interface at boot time.

6.3.2.6 Dynamic IP — DHCP client `udhcpc`

The DHCP client **udhcpc** supplied with Busybox uses the `/usr/share/udhcpc/default.script` to get an IP address dynamically at boot. It supports various options which you can view with the `--help` option:

```
# udhcpc --help
```

To disable **udhcpc** on `eth0` or modify the interface (eg `eth1`), you must edit the `/etc/network.conf` file and place the value `"no"` in the variable `DHCP=`:

```
# Dynamic IP address.
# Enable/disable DHCP client at boot time.
DHCP="no"
```

6.3.2.7 Static IP — Using a specific address

You can specify a fixed IP address to configure at boot time by using the value `"yes"` in the variable `STATIC=`:

```
# Static IP address.
# Enable/disable static IP at boot time.
STATIC="yes"
```

For the configuration to work, you must specify an IP address, its subnet mask, a default gateway (gateway) and DNS server to use. Example:

```
# Set IP address, and netmask for a static IP.
IP="192.168.0.6"
NETMASK="255.255.255.0"

# Set route gateway for a static IP.
GATEWAY="192.168.0.1"

# Set DNS server. for a static IP.
DNS_SERVER="192.168.0.1"
```

6.3.2.8 Static routes

Static routes can be added at any time via the **route add** command:

```
route add -net 192.168.20.0 netmask 255.255.255.0 gw 192.168.21.2
```

The static route will remain active until the next reboot. In order to make these routes persistent, add them to `/etc/init.d/local.sh`

6.3.2.9 PPPoE connection kernel-mode

PPPoE connection in kernel-mode needs 2 files. The first file is `/etc/ppp/options` where you must specify your login name:

```
plugin rp-pppoe.so
name <your_login>
noipdefault
defaultroute
mtu 1492
mru 1492
lock
```

Now you have to configure `/etc/ppp/pap-secrets` or `/etc/ppp/chap-secrets`:

```
# client       server       secret           IP addresses
"your_login"     *          "your_password"
```

The config file `/etc/resolv.conf` will be automatically loaded. Finished, you can now connect to the internet with **pppd**:

```
# pppd eth0
```

On an installed system you can start **pppd** on each boot using the local startup script: `/etc/init.d/local.sh`

6.3.2.10 Ethernet PPPoE ADSL Modem — PPPoE with `rp-pppoe`

This section is about setting up an ADSL Internet connection using an ethernet PPPoE modem in bridge mode. To set an ASDL protocol via PPPoE, SliTaz provides the utilities package **rp-pppoe**. Using **pppoe-setup** is a snap and you can quickly configure the network. If you use DCHP it's even easier, because the server from your ISP will take care of everything. If you do not have DHCP, you must first disable its use via `DHCP="no"` in the configuration file `/etc/network.conf`. It should be noted that to modify configuration files and system logs you must first become root. To install and change the variable `DHCP` with **Nano** (`Ctrl+X` to save & exit):

```
$ su
# tazpkg get-install rp-pppoe
# nano /etc/network.conf
```

Configure with `pppoe-setup`

To begin to configure your PPPoE connection, you must first open an **`Xterm`** or Linux console and launch **`pppoe-setup`** and then begin to answer the following questions:

```
# pppoe-setup
```

1. Enter your username, please note that this is the username with which you communicate with your ISP.
2. Internet interface, default is `eth0` unless you have more than one, in which case you will have `eth1`, `eth2`, etc. Usually the `Enter` key is sufficient.
3. If you have a permanent ASDL link answer `yes`, otherwise answer `no` (default).
4. Specify the primary and secondary DNS your ISP uses (you may have to ask).
5. Enter the password with which you communicate with your ISP (you need to enter it twice).
6. Choose the firewall settings depending on your hardware. If you have a router you can enter `1` or `2`. If in doubt enter `1`.

Start and Stop the connection

Still using the command line, simply type **`pppoe-start`** to start the connection. A few seconds later the system tells you that it is connected. If it gives you a message like `TIMED OUT`, you may have poorly configured or the connection is defective. Please check the wiring and repeat the installation from the beginning. To start the connection:

```
# pppoe-start
```

To stop the connection, you can type:

```
# pppoe-stop
```

To check the connection status:

```
# pppoe-status
```

6.3.2.11 Install network card driver

In case you need a network card driver and don't know the driver name, you can use the command **`lspci`** to find your card and then **`modprobe`** to load a module. In Live mode you can use the SliTaz boot option `modprobe=modules` to automatically load Kernel modules. To get a list of all available network card drivers, display PCI eth cards and load a module:

```
# modprobe -l | grep drivers/net
# lspci | grep [Ee]th
# modprobe -v module_name
```

On an installed system you just need to add the `module_name` to the variable `LOAD_MODULES` in `/etc/rcS.conf` to load your module on each boot.

Important: SliTaz-4.0 and newer: `/etc/firewall.conf` is moved to `/etc/slitaz/firewall.conf`

`iptables` rules are moved from `/etc/init.d/firewall.sh` to `/etc/slitaz/firewall.sh`

Reference: http://hg.slitaz.org/slitaz-tools/rev/769

6.3.2.12 Manage the Firewall (firewall) using Iptables

SliTaz provides a very basic firewall, the kernel security rules are launched at boot time and **`iptables`** rules are disabled by default. You can activate/disable these at startup by using the configuration file `/etc/firewall.conf`.

The default firewall script begins with its own set options for the Kernel ie. ICMP redirects, source routing, logs for unresolved addresses and spoof filters. The script then launches the rules defined in the `iptables_rules()` function of the configuration file: `/etc/firewall.conf`.

The firewall uses Iptables, it consists of two files: `/etc/firewall.conf` and `/etc/init.d/firewall`, you shouldn't need to modify these. Note Iptables has lots of options. For more infomation see the official documentation available online: http://www.netfilter.org/documentation/.

Start, stop, restart the firewall

The script `/etc/init.d/firewall` lets you start/restart, stop or display the status of the firewall. The restart option is often used to test new rules after editing the configuration file. Example:

```
# /etc/init.d/firewall restart
```

Enable/Disable the firewall at boot

To enable/disable options specific to the Kernel place `"yes"` or `"no"` in the variable `KERNEL_SECURITY=`:

```
# Enable/disable kernel security at boot time.
KERNEL_SECURITY="yes"
```

And to activate/deactivate the iptables rules, it is necessary to modify the `IPTABLES_RULES=` variable:

```
# Enable/disable iptables rules.
IPTABLES_RULES="yes"
```

Add, delete or modify the iptables rules

At the bottom of the configuration file: `/etc/firewall.conf`, you will find a function named: `iptables_rules()`. This function contains all of the **`iptables`** commands to launch when the firewall starts. To delete a rule, it is advisable to comment out the corresponding line with a `#`. It is not advisable to leave the function completely empty, if you want to disable the **`iptables`** rules just add `"no"` to the variable `IPTABLES_RULES=` in the configuration file.

Here's an example of using **`iptables`** rules. It only allows connections on the localhost and the local network, and ports 80, 22, and 21 used by the web server HTTP, the SSH secure server and FTP respectively. All other incoming and outgoing connections are refused, so it's fairly restrictive.

```
# Netfilter/iptables rules.
# This shell function is included in /etc/init.d/firewall.sh
# to start iptables rules.
#
iptables_rules()
{

# Drop all connections.
iptables -P INPUT  DROP
iptables -P OUTPUT DROP

# Accept all on localhost (127.0.0.1).
iptables -A INPUT  -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

# Accept all on the local network (192.168.0.0/24).
iptables -A INPUT  -s 192.168.0.0/24 -j ACCEPT
iptables -A OUTPUT -d 192.168.0.0/24 -j ACCEPT

# Accept port 80 for the HTTP server.
iptables -A INPUT  -i $INTERFACE -p tcp --sport 80 -j ACCEPT
iptables -A OUTPUT -o $INTERFACE -p tcp --dport 80 -j ACCEPT

# Accept port 22 for SSH.
iptables -A INPUT  -i $INTERFACE -p tcp --dport 22 -j ACCEPT
iptables -A OUTPUT -o $INTERFACE -p tcp --sport 22 -j ACCEPT

# Accept port 21 for active FTP connections.
iptables -A INPUT  -i $INTERFACE -p tcp --dport 21 -j ACCEPT
iptables -A OUTPUT -i $INTERFACE -p tcp --sport 21 -j ACCEPT

}
```

6.3.3 PSTN

author domcox, linea

6.3.3.1 Introduction

Dial-up connections use analog modems to transmit digital signals over traditional copper phone lines. These connections require no infrastructure other than the telephone network.

- Dial-up is often the only choice available for rural or remote areas where broadband installations do not exist.
- In case of broadband failure, power outage or governments shutting down internet access and cellphone networks, (already happened in several countries) dial-up is a way to get back on-line.
- Where telephone access is widely available, dial-up remains an option to travellers.

6.3.3.2 Modem choice

Software modem

Any notebook modem is almost certainly a **software modem** and needs a specific driver. The biggest problem users may have with notebook modems is unsupported drivers and this is not specific to Linux: many vendors have disappeared since the launch of Win98/XP, and no drivers have been developed for these old modems to work with Windows Vista or later, and don't expect a 64bit driver. Modems that fall into this category include most Diamond modems (manufacturer gone out of business), most PCTel modems, many Smartlink modems, many Motorola modems, some USR modems, as well as some modems based upon chipsets from Conexant, and Agere.

Linux may provide support drivers for software modems. Though there are a few software modems that you can find drivers for and in general as these drivers are not free software, installation or packaging is far from being an easy task.

Hardware modem

Full **hardware modems** normally work "out of the box" with Linux, including Slitaz GNU/Linux. There is no need to install drivers for such a modem in Slitaz GNU/Linux as they are already provided. Hardware modems can be an internal device (PCI slot), an expansion card (PCMCIA, card-express) or an external peripheral connected to one of the serial or USB ports.

Internal vs External

Note: Although the DC voltage in telephone lines does not cause immediate danger to the user, the AC ring signal (70-120V AC) can still give a nasty shock. Moreover, telephone wires are also exposed to many different environmental effects (nearby lightning, ground potential differences in buildings, interference from power lines) which can sometimes cause high voltage spikes on the telephone wires that may damage your modem. It happened to my modem and I found it easier and cheaper to change an external modem than to replace a full motherboard.

6.3.3.3 Modem installation

Software modem

Software modems will work under Linux only if a driver exists and gets installed. Slitaz offer several packages of drivers:

- **`linmodem-agrsm`**: Agere Modem Drivers for 11c11040 chipsets
- **`linmodem-intel-536ep`**: Intel Modem Drivers for 536EP chipset
- **`linmodem-intel-537`**: Intel Modem Drivers for 537 chipsets
- **`linmodem-slmodem`**: Drivers for the Smartlink winmodems

First, you have to identify the modem's chipset, then select the driver that fits your software modem. You'll find valuable help at:

- http://www.tldp.org/HOWTO/Modem-HOWTO.html
- http://linmodems.technion.ac.il/Linmodem-howto.html

- http://linmodems.org/

PCMCIA modem

If you want to use a PCMCIA Modem card, you have to install the driver for PCMCIA serial devices:

```
# tazpkg get-install linux-dialup
```

When you insert the PCMCIA modem, the driver will generally attempt to allocate the first unused serial device used by the card. This command will tell you the device allocated:

```
# dmesg | tail
```

```
$ dmesg | tail -3
pcmcia 0.1: pcmcia: registering new device pcmcia0.1 (IRQ: 3)
serial_cs 0.1: trying to set up [0x0101:0x0556] (pfc: 0, multi: 1, quirk: ␣
↪ (null))
serial8250: ttyS0 at I/O 0x3f8 (irq = 3) is a 16550A
```

Serial port configuration

All modems, hardware or software modems with the appropriate driver loaded are reached using serial ports. The serial port is either a regular port on your system or provided hard-wired as part of an internal-modem card. In all cases, to communicate with your modem, Slitaz GNU/Linux needs only to communicate with the serial port.

PC systems accept up to 4 regular serial ports:

Num	Device	Port	Irq
1	/dev/ttyS0	0x03f8	4
2	/dev/ttyS1	0x02f8	3
3	/dev/ttyS2	0x03e8	4
4	/dev/ttyS3	0x02e8	3

Software modems may add exotic devices to your system like `/dev/ttyAGS3`, `/dev/536ep`, `/dev/ttySL0`...

In order to test the modem, we first need to configure the serial device. Install **setserial**:

```
# tazpkg get-install setserial
```

And type:

```
# setserial /dev/<serial_device> irq <ii> port <0xzz> autoconfig auto_irq
```

with the proper values for the serial device, irq and port.

Then install the **minicom** package:

```
# tazpkg get-install minicom
```

Then as root, open an **XTerm** and type in:

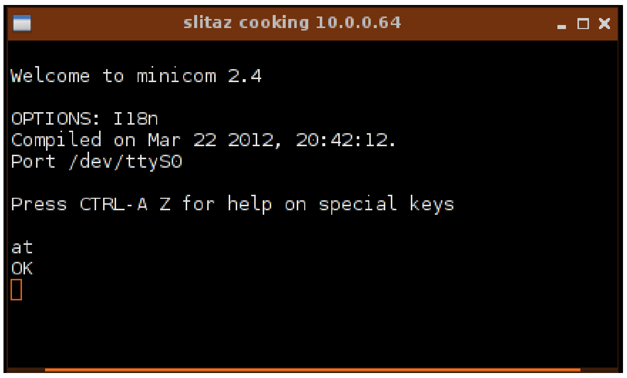
```
# minicom -c on /dev/<serial_device> -o
```

Once **minicom** is up, type:

```
AT
```

Your modem should answer:

```
OK
```


slitaz cooking 10.0.0.64

```
Welcome to minicom 2.4

OPTIONS: I18n
Compiled on Mar 22 2012, 20:42:12.
Port /dev/ttyS0

Press CTRL-A Z for help on special keys

at
OK
```

If your modem doesn't send a response, try another device (or maybe your serial device is badly configured), you'll find help at:

- http://www.tldp.org/HOWTO/html_single/Serial-HOWTO
- http://www.tldp.org/HOWTO/html_single/Modem-HOWTO

Tip: Once you have a valid and tested connection to your modem; to simplify future operations it's recommended to create a symbolic link from the serial device connected to your modem to `/dev/modem`:

```
# ln -s /dev/ttyS0 /dev/modem
```

6.3.3.4 Connect the modem to the phone line

Connect to the telephone wiring

The modem must be connected to an analog telephone line. In some countries, a country-specific modem cable adapter is also required. Jacks for digital PBX systems may resemble analog telephone jacks, but they are not compatible with the modem.

Pay attention to the different kinds of phone lines: analog and ISDN. You can't connect an analog modem to an ISDN port and vice versa. Connecting to the wrong port may even destroy your modem. If your machine features an internal modem as well as an internal ethernet card, also pay attention to plug the right cable into the plug. Otherwise you may easily damage your hardware.

Handling different country standards

Telephone signalling (dial tone, engaged signal, etc) and DC current regulators options are specific to a particular country. The modem needs to recognise each of these to be able to respond accordingly.

Your phone line needs a DC current of 25 to 35mA for a working phone. In some countries the DC current regulator is located in the telephone exchange at the Central Office, but in other countries it's in the telephone set so if you do not properly set the country in which you operate your modem, you risk plugging in your modem to a phone line with no regulator at all and too much DC current (or both regulators and a too low DC current). In both cases, you won't have a reliable connection and could even run the risk of destroying your modem if the DC current is too high.

If you bought and operate a hardware modem in the country where you live, it's not an issue as you can make some assumptions as to that it will meet the national specifications.

Old hardware modems were specific to a particular country. But if you operate a software modem or you're a traveller, you have to set up your modem according to the country you are in. Fortunately, most modems currently in use can be programmed to be used with any country's phone system.

Tip: Commands for listing or setting up Country codes are specific to your modem or manufacturer so you just have to refer to the manual of your modem. The *country-codes* (page 328) page will help you set up the correct country code.

6.3.3.5 Configure and Use a Dial-Up Connection

Launch the Slitaz Netbox Manager from the System tools menu or via a terminal:

```
$ subox netbox
```

ppp-on file

Select the *ppp* tab, then click on the *Tune* Button. Go to to the end of the file and change `/dev/ttyS0` to `/dev/modem` and set the speed to 115200 bds. These settings usually work but you may need to set specific values according to your modem setup.

```
# exec /usr/sbin/pppd debug lock modem crtscts /dev/modem 115200 \
        asyncmap 20A0000 escape FF kdebug 0 $LOCAL_IP:$REMOTE_IP \
        noipdefault netmask $NETMASK defaultroute connect $DIALER_SCRIPT
```

DNS

If you add the option `usepeerdns`, the peer will ask for up to 2 DNS server addresses. The addresses supplied by the peer (if any) are processed by the `/etc/ppp/ip-up` script and the `/etc/resolv.conf` updated with the address(es) supplied by the peer. Otherwise, simply enter the address of a well known DNS server like OpenDNS in `/etc/resolv.conf`:

```
nameserver 208.67.222.222
nameserver 208.67.220.220
```

chat file

If you need to send a specific init option to your modem before starting **ppp**, edit the `/etc/ppp/scripts/ppp-on-dialer` file. Add the new string before `OK ATDT$TELEPHONE`:

```
exec chat -v                                            \
        TIMEOUT         3                               \
        ABORT           '\nBUSY\r'                      \
        ABORT           '\nNO ANSWER\r'                 \
        ABORT           '\nRINGING\r\n\r\nRINGING\r'    \
        ''              \rAT                            \
        'OK-+++\c-OK'   ATH0                            \
        TIMEOUT         30                              \
        OK              ATL1+GCI=20                     \
        OK              ATDT$TELEPHONE                  \
        CONNECT         ''                              \
        ogin:--ogin:    $ACCOUNT                        \
        assword:        $PASSWORD
```

This file is a chat program designed to connect the user with a standard UNIX style getty/login connection. For calling an ISP from a dial-out machine you need in most cases to delete the last two lines:

```
exec chat -v                                            \
        TIMEOUT         3                               \
        ABORT           '\nBUSY\r'                      \
        ABORT           '\nNO ANSWER\r'                 \
        ABORT           '\nRINGING\r\n\r\nRINGING\r'    \
        ''              \rAT                            \
        'OK-+++\c-OK'   ATH0                            \
        TIMEOUT         30                              \
        OK              ATDT$TELEPHONE                  \
        CONNECT         \c
```

Save the file.

Start connection

Enter your Login/password and Telephone number of your ISP in the corresponding fields, then select the *Start* button. The modem should dial your Internet Provider.

debug

As **pppd** is started with a debug option, you can see the debug log. Enter:

```
$ tail -f /var/log/messages
```

Sample output:

```
$ tail -f /var/log/messages
Mar 23 11:25:29 (none) daemon.notice pppd[6240]: pppd 2.4.5 started by␣
↪root, uid 0
Mar 23 11:25:30 (none) local2.info chat[6242]: timeout set to 3 seconds
Mar 23 11:25:30 (none) local2.info chat[6242]: abort on (\nBUSY\r)
Mar 23 11:25:30 (none) local2.info chat[6242]: abort on (\nNO ANSWER\r)
Mar 23 11:25:30 (none) local2.info chat[6242]: abort on␣
↪(\nRINGING\r\n\r\nRINGING\r)
Mar 23 11:25:30 (none) local2.info chat[6242]: send (rAT^M)
Mar 23 11:25:30 (none) local2.info chat[6242]: expect (OK)
```

(continues on next page)

(continued from previous page)

```
Mar 23 11:25:31 (none) local2.info chat[6242]: rAT^M^M
Mar 23 11:25:31 (none) local2.info chat[6242]: OK
Mar 23 11:25:31 (none) local2.info chat[6242]:  -- got it
Mar 23 11:25:31 (none) local2.info chat[6242]: send (ATH0^M)
Mar 23 11:25:31 (none) local2.info chat[6242]: timeout set to 30 seconds
Mar 23 11:25:31 (none) local2.info chat[6242]: expect (OK)
Mar 23 11:25:31 (none) local2.info chat[6242]: ^M
Mar 23 11:25:31 (none) local2.info chat[6242]: ATH0^M^M
Mar 23 11:25:31 (none) local2.info chat[6242]: OK
Mar 23 11:25:31 (none) local2.info chat[6242]:  -- got it
Mar 23 11:25:31 (none) local2.info chat[6242]: send (ATZ^M)
Mar 23 11:25:31 (none) local2.info chat[6242]: expect (OK)
Mar 23 11:25:31 (none) local2.info chat[6242]: ^M
Mar 23 11:25:31 (none) local2.info chat[6242]: ATZ^M^M
Mar 23 11:25:31 (none) local2.info chat[6242]: OK
Mar 23 11:25:31 (none) local2.info chat[6242]:  -- got it
Mar 23 11:25:31 (none) local2.info chat[6242]: send (ATDT0860922000^M)
Mar 23 11:25:31 (none) local2.info chat[6242]: expect (CONNECT)
Mar 23 11:25:31 (none) local2.info chat[6242]: ^M
Mar 23 11:25:56 (none) local2.info chat[6242]: ATDTxxxxxxx^M^M
Mar 23 11:25:56 (none) local2.info chat[6242]: CONNECT
```

6.3.4 AT Codes for Country Select

author domcox, linea

Examples of country settings for some brands.

Country	Conexant ACF	Lucent Apollo/Mars	Motorola SM56	USR Courier
Auto-Select	*NC0	—	—	—
Argentina	—	—	—	C10=24
Australia	*NC40	%T19,0,1	+GCI=09	C10=07
Austria	*NC1	%T19,0,F	+GCI=0A	—
Belgium	*NC2	%T19,0,2	+GCI=0F	—
Brazil	—	—	+GCI=16	C10=09
Bulgaria	*NC27	—	—	—
Canada	*NC20	%T19,0,1C	+GCI=20	C10=00
China	*NC41	%T19,0,11	—	C10=10
CTR21	—	—	—	C10=08
Czech Republic	*NC19	—	+GCI=2E	C10=11
Denmark	*NC3	%T19,0,3	+GCI=31	—
Finland	*NC4	%T19,0,4	+GCI=3C	—
France	*NC5	%T19,0,5	+GCI=3D	C10=04
Germany	*NC6	%T19,0,6	+GCI=42	C10=02
Greece	*NC17	%T19,0,21	—	—
Hong Kong	*NC42	%T19,0,1B	+GCI=50	C10=12
Hungary	*NC23	%T19,0,22	—	C10=13
India	*NC30	%T19,0,1E	—	C10=25
Indonesia	—	%T19,0,17	—	C10=14
Ireland	*NC7	%T19,0,1A	+GCI=57	—

Continued on next page

Table 1 – continued from previous page

Country	Conexant ACF	Lucent Apollo/Mars	Motorola SM56	USR Courier
Israel	*NC18	—	+GCI=58	—
Italy	*NC8	%T19,0,8	+GCI=59	C10=03
Japan	*NC43	%T19,0,10	+GCI=00	C10=06
Korea	*NC44	%T19,0,12	—	C10=15
Luxembourg	*NC9	—	—	—
Malaysia	*NC92	%T19,0,13	+GCI=6C	C10=16
Mexico	*NC21	%T19,0,1D	—	C10=17
Netherlands	*NC10	%T19,0,7	+GCI=7B	—
New Zealand	*NC48	%T19,0,9	—	C10=18
Norway	*NC11	%T19,0,A	+GCI=82	—
Philipines	*NC43	%T19,0,20	—	—
Poland	*NC24	—	—	C10=19
Portugal	*NC12	%T19,0,18	+GCI=8B	—
Romanaia	—	—	—	C10=20
Russia	*NC25	—	—	C10=05
Singapore	*NC47	%T19,0,14	+GCI=8C	C10=21
Slovac Republic	*NC26	—	—	—
South Africa	*NC99	%T19,0,24	+GCI=9F	C10=22
Spain	*NC13	%T19,0,B	+GCI=A0	—
Sweden	*NC14	%T19,0,C	+GCI=A5	—
Switzerland	*NC15	%T19,0,D	+GCI=A6	—
Taiwan	*NC46	%T19,0,14	—	C10=23
Thailand	—	%T19,0,16	+GCI=A9	—
Turkey	—	%T19,0,23	+GCI=AE	—
UK	*NC16	%T19,0,E	+GCI=B4	C10=01
USA	*NC22	%T19,0,19	+GCI=B5	C10=00
Vietnam	—	%T19,0,1F	—	—

Country Codes

Select the proper country code for your current location. The country codes vary depending upon the telephone system. Additionally, some countries may have more than one country code.

Country Location	Country Code
Albania	U.S.
Antigua and Barbuda	U.S.
Armenia	U.S.
Aruba	U.S.
Australia	Australia
Austria	Austria
Azerbaijan	U.S.
Bahamas	U.S.
Bahrain	U.S.
Barbados	U.S.
Belarus	U.S.
Belgium	Belgium

Continued on next page

Table 2 – continued from previous page

Country Location	Country Code
Bermuda	U.S.
Bolivia	U.S.
Bosnia-Herzegovina	U.S.
Brazil	Brazil
Brunei	U.S.
Bulgaria	U.S.
Canada	U.S.
Cayman Islands	U.S.
Chile	U.S.
Chile	Chile
China (Mainland)	China
Columbia	U.S.
Costa Rica	U.S.
Croatia	U.S.
Cyprus	Cyprus
Cyprus	U.S.
Cyprus	Israel
Czech Republic	Czech Republic
Denmark	Denmark
Dominican Republic	U.S.
Egypt	U.S.
El Salvador	U.S.
Equador	U.S.
Estonia	U.S.
Federal Republic of Yugoslavia	U.S.
Finland	Finland
France	France
French Polynesia	France
Georgia	U.S.
Germany	Germany
Greece	Belgium
Greece	Greece
Guadeloupe	France
Guam	U.S.
Guatemala	U.S.
Guiana	U.S.
Haiti	U.S.
Honduras	U.S.
Hong Kong	Hong Kong
Hungary	Hungary
Iceland	Iceland
India	India
Indonesia	Indonesia
Ireland	Ireland
Israel	Israel
Italy	Italy
Ivory Coast	U.S.

Continued on next page

Table 2 – continued from previous page

Country Location	Country Code
Jamaica	U.S.
Japan	Japan
Jordan	U.S.
Kazakhstan	U.S.
Kenya	U.S.
Kenya	Israel
Korea	Korea
Kuwait	U.S.
Kygystan	U.S.
Latvia	U.S.
Lebanon	U.S.
Lithuania	U.S.
Luxembourg	Luxembourg
Macao	U.S.
Martinique	France
Maylasia	Maylasia
Mexico	U.S.
Moldava	U.S.
Morocco	U.S.
Netherlands Antilles	Netherlands
Netherlands/Holland	Netherlands
New Zealand	New Zealand
Nicaragua	U.S.
Nigeria	U.S.
Norway	Norway
Oman	U.S.
Pakistan	U.S.
Panama	U.S.
Paraguay	U.S.
Peru	U.S.
Philippines	Philippines
Poland	Poland
Portugal	Portugal
Puerto Rico	U.S.
Quatar	U.S.
Republic of Macedonia	U.S.
Reunion	France
Romania	U.S.
Russia	Russia
Saudi Arabia	U.S.
Singapore	Singapore
Slovak Republic of Slovakia	Slovak Republic of Slovakia
Slovenia	U.S.
Spain	Spain
St. Thomas	U.S.
Surinam	U.S.
Sweden	Sweden

Continued on next page

Table 2 – continued from previous page

Country Location	Country Code
Switzerland	Switzerland
Syria	U.S.
Tajikistan	U.S.
Thailand	U.S.
Trinidad	U.S.
Tunisia	U.S.
Turkey	Turkey
Turkmenistan	U.S.
Ukraine	U.S.
United Arab Emirates	United Arab Emirates
United Kingdom	United Kingdom
United States	U.S.
Uruguay	U.S.
Uzbekistan	U.S.
Venezuela	Argentina
Vietnam	U.S.
Virgin Islands	U.S.
Yeman	U.S.
Zimbabwe	U.S.

6.3.5 System Administration

author jozee, linea, emgi, hgt

6.3.5.1 Devices and disk access

With Linux your disks and USB media are seen as devices. To access them you must first mount a device on a mount point (directory). On SliTaz you can graphically mount devices by using **`mountbox`** or the command line. To mount the first disk of a local hard disk on `/mnt/disk`:

```
# mkdir -p /mnt/disk
# mount /dev/hda1 /mnt/disk
```

To mount a CD-ROM or an USB media you should use mount points located in `/media`. Note that for a CD-ROM, you just have to specify the device path. For a flash key, the mount point already exists:

```
# mount /dev/cdrom
# mount /dev/sda1 /media/flash
```

NTFS filesystem

If you need read/write access to Windows NTFS filesystems you must install a few additional packages from the mirror. The **`ntfs-3g`** driver provides stable access to NTFS partitions and **`ntfsprogs`** provides manipulation tools dependent on FUSE. Note that you can format, move or resize NTFS partitions graphically with **`Gparted`**.

NFS

NFS (Network File System) is the native Unix/Linux method for sharing file systems. In this respect its function is similar to SAMBA. The most popular version is still NFSv3 which is able to use either UDP or TCP as the network protocol. The older NFSv2 was only capable of using UDP. On a local LAN, UDP is still the fastest protocol; TCP is to be preferred when the machines are connected over a WAN. NFSv3 has been superseded by NFS version4 which has notable improvements (security) over v3 but its configuration has become a lot more complex as a result.

An NFS server exports a part of it's file system; i.e. makes it available on the network. The server is configured with details about client IP addresses or host names and can restrict their access to the file system. Access can be read-only, read-write or no access at all. An NFS client simply mounts the exported file systems as if they were local devices.

The NFS software in SliTaz makes it possible to run both as a server or a client. To start the NFS processes, you need to run the init script in `/etc/init.d/`.

```
# /etc/init.d/nfsd start|stop
```

Tip: The NFS daemons must be running also when the machine acts as a client.

NFS Software Installation

To start using NFS, the following packages are required:

```
# tazpkg -l | grep nfs
linux-nfsd                2.6.37              base-system
nfs-utils                 1.2.2               system-tools
```

Install them using:

```
# tazpkg -gi linux-nfsd
# tazpkg -gi nfs-utils
```

`/etc/exports`

Here is a sample exports file:

```
# /etc/exports: the access control list for filesystems which may be
#               exported to NFS clients.  See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes       hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,
↪no_subtree_check)
#
# Example for NFSv4:
# /srv/nfs4        gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes  gss/krb5i(rw,sync,no_subtree_check)
#
/usb1           192.168.1.0/24(rw,sync,no_subtree_check)
```

The server is only exporting one drive: usb1. Clients must be in the 192.168.1.0/24 network and they have read-write access.

To use nfs on the client; all you need to do is start nfsd and mount the share:

```
# /etc/init.d/nfsd start
# mount server:/usb1 /mnt/usbdrive1
```

Please note the specific format for nfs shares **servername** colon **slash-mountpoint**. Naturally you must also make sure that the specified mount point (directory) exists on the client.

6.3.5.2 Users, groups and passwords

To manage users and groups on your SliTaz system you must use the command line, but file permissions can be changed graphically using the **PCmanFM** file manager. To add or remove users and groups you must be root. Root can also change all user passwords and a single user can only change his/her own password. To add or remove a user named linux:

```
# adduser linux
# deluser linux
```

Manipulating users & group membership

Linux groups are a mechanism to manage a collection of computer system users. All Linux users have a user ID and a group ID and a unique numerical identification number called a userid (UID) and a groupid (GID) respectively. Groups can be assigned to logically tie users together for a common security, privilege and access purpose. It is the foundation of Linux security and access. Access to files and devices may be granted based on a user ID or a group ID. This mechanism is the same for all of Linux but the way it is configured varies per distribution. Sometimes additional or different commands are used, like for example **usermod**, **chgrp**, **useradd** or **groupadd**. Below is an overview of how to handle users, groups and group membership on Slitaz.

The SliTaz way is using only four commands. Simple & Elegant. ;-)

```
# adduser    <username>              # adds a user
# deluser    <username>              # deletes a user
# addgroup   <groupname>             # adds a group
# delgroup   <groupname>             # deletes a group
# addgroup   <username>  <groupname> # adds a user to a group
# adduser -G <groupname> <username>  # adds the user to an additional group
# delgroup   <username>  <groupname> # deletes a user from a group
```

Any user can be member of any group and the combination of user & group permissions allows for granular access to system resources.

passwd

To change the current user's password or change the password of a specific user, you must use the **passwd** command:

```
$ passwd
# passwd username
```

`audio` group

If you want a new user to be able to listen to music he must be in the `audio` group. To add an existing user to the `audio` group:

```
# adduser -G audio user_name
```

6.3.5.3 Language and keyboard layout

SliTaz saves the configuration of the default locale in `/etc/locale.conf` which is read by `/etc/profile` on each login and the keyboard setting is stored in `/etc/keymap.conf`. These two files can be edited with your favorite editor or configured respectively with **`tazlocale`** and **`tazkeymap`**. You can modify the settings you chose on the first boot by typing as root administrator:

```
# tazlocale
```

Or:

```
# tazkeymap
```

To check all available locales or your current configuration, you can use the command **`locale`** as a single user or root (C for English):

```
$ locale -a
$ locale
```

6.3.5.4 Custom SHell

SliTaz uses the ash shell linked to sh provided by busybox. Ash is light, fast and standards compliant. To change the default shell for a user you can edit the `/etc/passwd` file using the corresponding line. After you login, `/etc/profile` is read first and then the user file `~/.profile`. You can edit these files with a text editor to configure the language, any aliases, etc.

Example: `~/.profile`

```
# ~/.profile: executed by Bourne-compatible login shells.
#

# Aliases.
alias ls='ls -F'
alias df='df -h'

# Env variables.
export EDITOR=nano
```

6.3.5.5 BASH Shell

On SliTaz you have the `ash` and `sh` shell with a link to **`Ash`**, this shell is provided by Busybox. If you wish to use the BASH (Bourne Again SHell), first as root install **`bash`**, copy the `.profile` found in your home directory and rename it `.bashrc`, then edit the `/etc/passwd` file with your favorite text editor and change your shell to `/bin/bash`

```
# tazpkg get-install bash
$ cp ~/.profile ~/.bashrc
# nano /etc/passwd
```

The next time you login BASH will be your default shell, you can confirm this by typing **`env`** on the command line.

6.3.5.6 Editors

Busybox supplies a clone of **`vi`** for normal text editing, but it does have its limitations. You can install the full **`vim`** editor with the command:

```
# tazpkg get-install vim
```

Or alternatively if you prefer **`emacs`**, SliTaz offers a tiny version:

```
# tazpkg get-install emacs
```

6.3.5.7 `sudo`

The **`sudo`** command can be applied on SliTaz:

```
# tazpkg get-install sudo
```

Note that the configuration file `/etc/sudoers`, should always be edited by the **`visudo`** command which locks the file and checks for errors.

6.3.5.8 System Time

To check the current system time, you can simply type:

```
$ date
```

TimeZone

On SliTaz, the timezone configuration file is saved in `/etc/TZ`, you can edit this with your favorite text editor or simply **`echo`** the changes. To view the available timezones, you can look in the `/usr/share/zoneinfo` directory. Here's an example using the timezone Europe/London:

```
# echo "Europe/London" > /etc/TZ
```

`rdate`

To synchronize the system clock with a network time server, you can (as root) use the **`rdate -s`** command:

```
# rdate -s tick.greyware.com
```

To display the time on the remote server, use the **rdate -p** command.

```
$ rdate -p tick.greyware.com
```

Using NTP

NTP is a protocol to synchronize the time on many different systems via a network. NTP is a client-server application which uses UDP port 123. This section describes how to configure your system as an NTP client deriving its time from the Internet. There are many servers available on the Internet which provide an NTP service.

Tip: Experience has shown that NTP servers seldom have a high availability, rather the opposite! This means it may not be such a good idea to configure a particular server as your time source, not even two or three. After a while you may find out that none of them is active any more and your time is drifting freely! A better way is to use the service from ntp.org[228]. They provide pools of NTP servers per country or region. Selecting one of these provides a more reliable connection to an NTP time source.

Although SliTaz is a small distribution, it provides several NTP implementations. Most notably:

1. **busybox ntpd** (included in the base installation). Using **busybox ntpd** from the command line:

   ```
   # busybox ntpd -p nl.pool.ntp.org
   ```

   or

   ```
   # ntpd -p nl.pool.ntp.org
   ```

2. **ntp.tazpkg** (install from packages repository). To install **ntp.tazpkg**:

   ```
   # tazpkg -gi ntp
   ```

   This package includes a decent set of NTP related binaries + the config file /etc/init.d/ntp

   ```
   # tazpkg list-files ntp

   Installed files with: ntp
   =========================
   /etc/init.d/ntp
   /etc/ntp.conf
   /usr/bin/ntpd
   /usr/bin/ntpdate
   /usr/bin/ntpdc
   /usr/bin/ntp-keygen
   /usr/bin/ntpq
   /usr/bin/ntptime
   /usr/bin/ntptrace
   /usr/bin/ntp-wait
   ```

(continues on next page)

228 http://www.ntp.org/

(continued from previous page)

```
/usr/bin/sntp
/usr/bin/tickadj
```

Be aware that SliTaz has several NTP daemons available. We have the Busybox app but also the **ntp** package. Both provide virtually the same functionality. If you have limited resources, the busybox version can provide all you need. If you want all the diagnostic stuff as well, you should rather go for installing **ntp.tazpkg**.

Starting `ntpd` at boot

Probably the easiest way to start **busybox ntpd** at boot is to add an entry like above to `/etc/init.d/local.sh`. The explanation below focuses on **ntp.tazpkg**. It is unclear which one was intended by the developers to be started by the Server Manager. This can be somewhat confusing. The verified way to configure the NTP daemon is to use the command line as detailed below.

To start `/usr/bin/ntpd` (**ntp.tazpkg**) at boot:

1. Make sure to install the package as shown above ;-).
2. Edit `/etc/daemons.conf` as follows. Add one line at the end:

   ```
   NTP_OPTIONS="-p xx.pool.ntp.org"
   ```

   (where xx = country.)

   Tip: The NTP daemon gets it options from `/etc/daemons.conf`. The configuration file `/etc/ntp.conf` which is referred to by the Server Manager seems to be unused and may be deleted.

   ---

3. Edit `/etc/rcS.conf` as follows. On the line with daemons to start, add `ntp` to the list:

   ```
   RUN_DAEMONS="inetd dbus hald slim firewall httpd ntp "
   ```

   Tip: Make sure to enter just `ntp`, not `ntpd`! The name is a reference to `/etc/init.d/ntp`

   These are the required steps. After completion, you may reboot to verify everything is indeed working as expected.

Use the following to check if the daemon is running:

```
$ ps -ef | grep ntpd
 1934 root       0:00 /usr/bin/ntpd -p nl.pool.ntp.org
 2193 root       0:00 grep ntpd
```

In this example, the first line shows the process we want to see.

Tip: Use `/etc/init.d/ntp` {start | stop | restart} to control the NTP daemon or specify the full path (`/usr/bin/ntpd`). Using **ntpd** on the command line without the full path causes the busybox version to be invoked.

Verifying `ntpd` operation

You may use **`ntpq`** to verify your connection to NTP servers

```
# ntpq -p nl.pool.ntp.org
     remote           refid      st t when poll reach  delay  offset  jitter
==============================================================================
*ntp0.nl.uu.net  .PPS.            1 u  632 1024  377  2.700   0.233   0.096
+ntp1.nl.uu.net  .PPS.            1 u  504 1024  377  1.742   0.356  41.789
-chime1.surfnet. 194.171.167.130  2 u  298 1024  377  0.677   0.102   0.134
+chime4.surfnet. .PPS.            1 u  422 1024  367  9.652  -2.669   0.366
 tt165.ripe.net  .STEP.          16 u    - 1024    0  0.000   0.000   0.000
```

The ⋆ at the start of a line indicates the server you are currently synchronized to.

The column “st” shows the *stratum* or quality of the time source. 1 is best, 16 means unavailable. Important to check are the columns “reach” and those behind. Reach should be 377, everything else means polls were missed. Your daemon should be running for a while to get reliable output.

`hwclock`

`hwclock` allows you to synchronize the time of your hardware clock to the system clock or vice versa.

Synchronize the system clock to the hardware clock (`--utc = universal time`, `-l = local time`):

```
# hwclock -w --utc
```

Synchronize the hardware clock to the system clock:

```
# hwclock -s --utc
```

```
hwclock -u, --utc | -l, --localtime
```

Indicates that the Hardware Clock is kept in Coordinated Universal Time or local time, respectively. It is your choice whether to keep your clock in UTC or local time, but nothing in the clock tells which you've chosen. So this option is how you give that information to **`hwclock`**. If you specify the wrong one of these options (or specify neither and take a wrong default), both setting and querying of the Hardware Clock will be messed up.

Note: On SliTaz, **`hwclock`** must always be set to UTC. The result of a non-UTC hardware clock setting is an incorrect time for your timezone.

Synchronizing the `hwclock` with NTP

There are several ways to set the hardware clock to NTP time:

```
# busybox ntpd -dnqp nl.pool.ntp.org && hwclock -w -u
```

or:

```
# ntpdate -u nl.pool.ntp.org && hwclock -w -u
```

Alternative three: (quite old, may not work on all servers)

```
# rdate -s nl.pool.ntp.org && hwclock -w -u
```

Note that in all examples we used the `-u` option to set **hwclock** to UTC time.

Further reading: http://linux.die.net/man/8/hwclock

6.3.5.9 Execute scheduled commands

The daemon **crond** allows you to run commands automatically at a scheduled specific date or time. This is very useful for routine tasks such as system administration. The directory **cron** uses is `/var/spool/cron/crontabs`.

Each user on the system can have his/her own tasks, they are defined in the file: `/var/spool/cron/crontabs/`*`user`*. This file can be created oder modified by any user with the **crontab -e** command, using the default text editor. The crontab utility allows you (amongst other things), to list the tasks specific to the user.

```
# crontab -l # To list the crontab for user root.
```

or:

```
# crontab -l -u tux # To list the crontab for another user.
```

The syntax of the crontab files is as follows:

```
mm hh dd MMM DDD command > log
```

We will create a file with root privileges and test the daemon **crond** with a task performed every minute — writing the date to a file `/tmp/crond.test`, using GNU **nano** (`Ctrl+X` to save & exit):

```
# nano /var/spool/cron/crontabs/root
```

Add the line:

```
* * * * * date >> /tmp/crond.test
```

Launch **crond** with the option `-b` (background), configured via `/etc/daemons.conf` and using the startup script:

```
# /etc/init.d/crond start
```

You can wait a few minutes and view the contents of the file: `/tmp/crond.test`... OK:

```
# cat /tmp/crond.test
```

To stop or restart the daemon **crond**:

```
# /etc/init.d/crond stop
```

or:

```
# /etc/init.d/crond restart
```

Invoke the daemon crond on every boot

To launch the daemon **crond** each time you boot the system, just add it to the variable `START_DAEMONS` in the configuration file `/etc/rcS.conf`, either before or after the web server or SSH server.

6.3.5.10 Add commands to be executed at boot

During the boot process, various scripts are executed to configure services, such as the start of the web server, networking, etc. On SliTaz there is a script `/etc/init.d/local.sh` which allows you to add commands to be launched at system startup. You can also create new scripts in `/etc/init.d`, their links in `/etc/rc.scripts` for shell scripts and use `/etc/rc.d` for links to the startup script daemon in `/etc/rcS.conf`:

```
# nano /etc/init.d/local.sh
```

6.3.6 X Window System

author jozee, linea, jpeg

6.3.6.1 X11 — X Window System

The X Window System or X11 provides a window manager running on top of a X server.

SliTaz 1.0 and 2.0 by default use the lightweight X server called **Xvesa** from the Xorg project (www.x.org).

Slitaz 3.0 by default uses the **Xorg** server, there is however a **Xvesa** flavor.

The X server can be started with the SLiM login manager or directly from a Linux console with the command **startx**, but for this you must first disable the Login Manager. To reconfigure your X session you can use **tazx** as root or as the current user if you start X from the command line.

6.3.6.2 Tazx — SliTaz X configuration tool

Tazx is the configuration tool to manage your X window sessions on a SliTaz box. Simply select a resolution and press *OK*. You can also select a (Xorg) session by selecting a video driver best suited to your hardware. After you first run **startx**, the configuration is saved in the executable files `~/.xsession` and `~/.xinitrc`. These files are then used to start a X session with either **startx** or via the SLiM login manager and can be easily altered with a text editor. **Tazx** can also be used to change your default window manager. Example:

```
# tazx jwm
```

6.3.6.3 SLiM — Simple Login Manager

SLiM is a lightweight session manager that is very easy to configure and is customizable using system themes. The configuration file is found in `/etc/slim.conf`. It defines window managers available via the `F1` key, the default user or theme, and the X window system parameters. SLiM offers special user commands like `console` to help manage the session.

In LiveCD mode you can disable SLiM with the boot option `screen=text`. On an installed system you can remove the package or delete `slim` from the `RUN_DAEMONS` variable in `/etc/rcS.conf`.

More details and themes can be found on the website

Default user

SLiM offers a way to pre-load a user login name, by default `tux` is configured for convenience. You can change this by editing the SLiM configuration file `/etc/slim.conf` and modifying the line `default_user` or just leave the line blank to avoid pre-loading a user name. Example:

```
default_user        tux
```

6.3.6.4 Xorg

Xorg is the default server on SliTaz and designed to work out of the box on most systems. It should detect and configure most devices such as keyboards, mice, displays, etc. Once installed, running **Tazx** allows you to reconfigure/reinstall the **xorg-server** package and select the correct driver for your card. Example:

First stop the **Xorg** server using `Alt+Ctrl+Backspace`, you should now be in console mode. Then run **tazx** as root:

```
# tazx
```

Then select Xorg and select your video driver, this reconfigures **Xorg**. Then restart the **SLiM** login manager:

```
# /etc/init.d/slim start
```

You can also do this by searching for and installing a video driver and reconfiguring **Xorg** manually (after stopping the server):

```
# tazpkg search xorg-xf86-video
# tazpkg get-install xorg-xf86-video-nv
# Xorg -configure
```

Then copy the newly generated file to `/etc/X11`:

```
# cp /root/xorg.conf.new /etc/X11/xorg.conf
```

And restart the login manager:

```
# /etc/init.d/slim start
```

`xorg.conf.d` — Configuration files

`Xorg` uses the configuration files found in the `xorg.conf.d` directory which are automatically setup when you first boot and can be easily edited with your favorite text editor. The files are configured separately into sections such as modules to be loaded, default screen, mouse, keyboard, etc. This document provides a few examples:

`10-ServerLayout.conf`:

```
Section "ServerLayout"
     Identifier     "X.org Configured"
     Screen      0  "Screen0" 0 0
EndSection
```

`30-Module.conf`:

```
Section "Module"
     Load  "dbe"
     Load  "dri2"
     Load  "extmod"
     Load  "dri"
     Load  "record"
     Load  "glx"
EndSection
```

Note that a `xorg.conf` file can also be found in `/etc/X11` as another way to configure **`Xorg`**. This file is read before all files in `/etc/X11/xorg.conf.d` and will NOT be erased by any updates.

6.3.6.5 Use `Xvesa` as X terminal (Deprecated)

You can use **`Xvesa`** as X terminal, if you have a machine on the network that accepts Xdmcp connections. To enable this, you can start the server with the option `-query` followed by the machine name or IP address. Example of machine 192.168.0.2 on a local network:

```
$ Xvesa -ac -shadow -screen 1024x768x24 -query 192.168.0.2
```

The use of a graphical remote server can be of great use, although response times of applications depend greatly on Internet speed and the remote machine's power. This technique works very well within a local area network (LAN) and allows you to control applications installed on the remote machine directly from the screen of the local machine from which you work. Note that the distant remote machine may have multiple accounts in use simultaneously and/or direct access.

6.3.6.6 Fonts

The management of Fonts (fonts) is powered by the package **`fontconfig`**. This package provides tools to add, list and manipulate fonts. The fonts can be installed in user space or at the system level, this means that each user can use his/her own fonts or the system administrator (root) can install fonts available to all users of the system. If you use USB media associated with the SliTaz LiveCD, you can easily install fonts and retain them for the next time you use the CD-ROM.

Installing fonts

At the system level fonts are installed in the directory `/usr/share/fonts`, core SliTaz provides TTF Vera fonts, they take up little space and are rendered correctly. At the root of user space `~/`, fonts are found in the hidden directory `.fonts`. To create a home directory to accommodate new fonts, you can use the graphical window manager **`emelFM2`**, **`Clex`** or the command line:

```
$ mkdir ~/.fonts
```

Once you have installed the fonts you need to run the **`fc-cache`** tool to generate configuration files, this ensures that your fonts are available for use in applications:

```
$ fc-cache
```

6.3.7 SliTaz and System Security

author jozee, linea

6.3.7.1 Security Policy

SliTaz has given a lot of consideration to system security. Applications are tested for many months before being included in the distribution. At boot time, a minimum of services are launched by the rc scripts. For a complete lists of daemons enabled, you can look at the `RUN_DAEMONS` variable in the `/etc/rcS.conf` configuration file:

```
$ cat /etc/rcS.conf | grep RUN_DAEMONS
```

To view the actual processes, their PID and memory usage, you can use the **`ps`** command or the **`htop`** utility:

```
$ ps
$ htop
```

6.3.7.2 Root — The system administrator

In a GNU/Linux system, the **root** user is the system administrator. **root** has all the rights to the system files and that of the users. It is advisable never to log in as **root** by using the command **`su`** followed by the password to obtain absolute rights over the system. Never log in as **root** and surf the internet for example. This allows you to create a double barrier in the case of an attack or intrusion after a download and makes it harder for a cracker to take control of your machine — first he must crack your password and then crack the **root** password of the system administrator.

A GNU/Linux system has secured at least two users, one to work and another to administer, configure or update the system (**root**). It's also advisable to entrust the administration of the system to a person.

6.3.7.3 Passwords

By default the SliTaz user *tux* doesn't have a password and the system administrator **root** comes with the password (`root`). You can easily change these by using the **`passwd`** command:

```
$ passwd
# passwd
```

6.3.7.4 Busybox

The file `busybox.conf` configures the applets and their respective rights. On the SliTaz LiveCD the commands **`su`**, **`passwd`**, **`loadkmap`**, **`mount`**, **`reboot`** and **`halt`** can be initiated by all users — the owner and group of these commands is **root** (`* = ssx root.root`). The `busybox.conf` file is readable by **root**, using the rights `600`. Note that the **`passwd`** command will not allow users to change their own password if it is not `ssx`.

6.3.7.5 LightTPD web server

On SliTaz the LightTPD web server is enabled by default at system startup, if you don't intend to use SliTaz in a server environment, you can safely disable it by removing it from the `RUN_DAEMONS` variable in the `/etc/rcS.conf` configuration file or to stop it manually:

```
# /etc/init.d/lighttpd stop
```

6.3.7.6 SSH Server

This small section is a compliment to the Secure SHell (SSH) page. On SliTaz the **`Dropbear`** SSH server is not run by default, we must add it to the variable `RUN_DAEMONS` in the configuration file `/etc/rcS.conf` for it to be enabled at system boot. Or to start the server manually:

```
# /etc/init.d/dropbear start
```

By default, **`Dropbear`** is launched with the following options:

-w	Disallow root logins.
-g	Disallow logins for root password.

You can add new options by editing the daemons configuration file `/etc/daemons.conf`. For all options, you can type: **`dropbear -h`**.

6.3.7.7 Pscan — Ports scanner

`Pscan` is a small utility of the Busybox project that scans the ports of your machine. You can use **`pscan`** to scan the localhost or a remote host using the name or IP address of the machine. **`Pscan`** will test all the ports from 1 – 1024 by default and list those that are open, their protocol and associated service (ssh, www, etc):

```
$ pscan localhost
```

6.4 Flavor

6.4.1 Generating a Customised LiveCD or LiveUSB

author jozee, linea, seawolf, toasted-ravioli, brianperry, hgt

Tazlito can create your own, personalised version of SliTaz in very little time. It allows you to choose precisely the software packages that you want and make a flavor file (`.flavor`) with it. You can choose to make your own file starting from a general slitaz flavor (such as *base*, *justx* or *core*), or make it based on an already customised flavor supplied by the SliTaz community.

Once you've downloaded (and possibly — further — customized) the flavor you chose, you can make an iso-file with it using **tazlito**, and then burn it to a CD.

Overview

Tazlito automates the process of building a customized ISO, and the method is quite straight-forward:

1. find a template (flavor) to work on

   ```
   $ tazlito list-flavors
   ```

2. download the template

   ```
   $ tazlito get-flavor flavor_name
   ```

3. look at the flavor's packages

   ```
   $ tazlito show-flavor flavor_name
   ```

4. extract the flavor

   ```
   $ tazlito extract-flavor flavor_name
   ```

5. put in a rootfs folder (make one by copying in files from `/etc/` to `/home/slitaz/distro`)
6. add additional files using the folder `/home/slitaz/distro/addfiles` (optional)
7. change options

   ```
   $ tazlito configure
   ```

8. create the ISO image

   ```
   $ tazlito gen-distro
   ```

9. burn to CD or USB stick

   ```
   $ tazusb gen-iso2usb
   ```

Notes

- Note that you can't delete packages, but you can use a stripped-down flavor to build your customized slitaz version.

- Note that you can modify the distro further by hacking the ISO once created, after which you can make it into a flavor again (see details below)
- Note that you can add some last changes (that require modification from inside SliTaz; like changing appearance, modifying the taskbar, . . .) by simply making an additional rootfs file with a usb stick made with the ISO of the SliTaz version you just made (using the **`# tazusb writefs lzma`** command), and then adding it to the extracted ISO (using the *hacker account* (page 355)). You can probably also reinclude these changes finally into your flavor file by means of the **`# tazlito iso2flavor <your-ISO_image> <your-flavor>`** command.
- Creating a LiveCD requires a minimum of 256MB of RAM.
- Compressing the file-system with LZMA may take some time. GZip can also be used, though its compression ratio is not as effective.
- It's possible to create a distribution from the LiveCD or from an installed system. In LiveCD mode, it's advisable to use persistent media or an USB drive to store generated files and save space in RAM.
- **`Tazlito`** is used by developers to generate the official LiveCD. Unlike the (now deprecated **`tazwok`**), it is installed by default on SliTaz. Note that you can also use it an a different distribution (non-SliTaz) if that Linux distro has the necessary packages (they can be downloaded if not). See *the instructions* (page 352) at the bottom of the page for more details.

6.4.1.1 A customized ISO in three commands

A good way of becoming familiar with **`Tazlito`** is to re-build the LiveCD. To do this, download the default *core* flavor (a `.flavor` file that contains all the necessary information to create a LiveCD) and build it with the default options. This takes only three commands and will help to demonstrate the process:

- Get the default (*core*) flavor

```
# tazlito get-flavor core
```

- Extract the flavor

```
# tazlito extract-flavor core
```

- Optional: Amend the list of packages or add additional files.
- Create the LiveCD

```
# tazlito gen-distro
```

That's it! You will find the LiveCD ISO image in `/home/slitaz/`*`version`*`/distro`.

6.4.1.2 The steps laid out in detail. . .

6.4.1.2.1 Part I: Downloading a Flavor

Flavors can be used to build various styles of LiveCDs by containing the list of packages (`distro-packages.list`) and sometimes additional files. This saves you the effort of saving the configuration files and provides a quick way of switching between them. A list of flavors available on the SliTaz servers can be downloaded and updated with **Tazlito**:

```
# tazlito list-flavors
```

or:

```
# tazlito list-flavors --recharge
```

The list of flavors will be automatically displayed, showing the size of the LiveCD and a brief description. To use a flavor, use the **get-flavor** command to download the flavor file, automatically extract the package list and description in the current directory, and put additional files in the `/home/slitaz/distro/addfiles` directory:

```
# tazlito get-flavor flavor_name
```

The description of the flavor can be displayed with the **show-flavor** command. Once your chosen flavor is downloaded, you can start to customise your distribution!

Tip: To store flavors, we suggest that you use a specific directory such as `/home/slitaz/flavors`. You can create this yourself and proceed inside:

```
# mkdir -p /home/slitaz/flavors
# cd /home/slitaz/flavors
```

6.4.1.2.2 Part II: Generating the Distribution

Important: When generating a distro, there are some essential **base-system** packages that must be included. The dependency information in each package will take effect so these will be automatically included, even if they are not specified in the package list.

For your first ISO, we advise you re-build the default LiveCD without modifying the list. When you are comfortable with the process, you can start to delete or add packages as required. The **gen-distro** command generates an ISO image with all the packages on the list. All dependencies will be fulfilled automatically, just as with normal package installation. It must be run as root and be in the same directory as the list and the configuration file.

Tazlito uses the `distro-packages.list` file to download each package and 'install' it into a pseudo-file-system. This is the distro tree and contains the whole of the LiveCD's operating system. This is not unlike *chroot*-ing in to another system from a LiveCD. It is then compressed and added to a CD ISO image with booting information.

To generate a LiveCD:

```
# tazlito gen-distro
```

Voilà! Your first LiveCD ISO image is generated in `/home/slitaz/distro`!

Cleaning & Re-Generating

As you have generated a LiveCD, you need to clean-up so another can be generated. Removing the resulting ISO image, cleaning the generated distro tree (but not your additional files!) can be done through the `clean-distro` option:

```
# tazlito clean-distro
```

You can now customise your LiveCD further. To start, add one or two packages to the list. When you are ready, re-generate a new ISO image with the `gen-distro` command:

```
# tazlito gen-distro
```

All the customisations applied so far should be effective on your new LiveCD ISO image.

Tip: **`Tazlito`** also offers several possibilities such as burning the ISO, copying additional files into the system or directly on to the CD-ROM. See the Tazlito Manual[229] for more information.

Hacking your customized SliTaz ISO

See also *Hacking SliTaz LiveCD* (page 355)

If you want to modify the splash screen, boot loader, configuration files, GRUB itself, … you need to "hack" the ISO you made using your flavor file using a "hacker user account". Note that you can also update your flavor-file with these extra modifications you made at the boot loader, splash screen, … by means of the

```
# tazlito iso2flavor <your-ISO_image> <your_flavor>
```

command.

To make a hacker user account, we must decompress the root file-system, add the account, then re-compress the root file-system. The first two stages can be executed as one command, the last stage another:

```
# tazlito addhacker
# tazlito gen-initiso
```

This adds the account only to the LiveCD; your running system will not be affected.

6.4.1.3 Adding the final changes

See also *Persistence & splash* (page 146)

As mentioned above, you can add some last changes (that require modification from inside SliTaz; like changing appearance, modifying the taskbar, …) by simply making an additional `rootfs` file with a USB stick made with the ISO of the SliTaz version you just made (using the

```
# tazusb writefs lzma
```

[229] http://hg.slitaz.org/tazlito/raw-file/tip/doc/tazlito.en.html

command), and then adding it to the extracted ISO (using the *hacker account* (page 355)). You can probably also reinclude these changes finally into your flavor file by means of the

```
# tazlito extract-iso <your-ISO_image> <destination>
```

command, combined with the

```
# tazlito pack-flavor <flavor_name>
```

command. These 2 commands replace the old **tazlito iso2flavor** command (which can't be used any more).

It should be noted that using this technique, you can probably delete packages too (see *the manual on making many-in-one flavors* (page 133)). However, you best avoid this method however since the files are then probably still in the flavor/ISO and the flavor/ISO is hence larger than what it needs to be. The files/applications will simply not be used.

6.4.1.4 Including Additional Files

The files containing flavors usually have additional files copied directly into the file-system or the root of the CD-ROM. The path to the files used is specified in the configuration file `tazlito.conf` — by default, these files are located in `/home/slitaz/distro/addfiles`. The additional files the *core* flavor provides are for example: the window manager **JWM** and some wallpapers. It is easy enough to modify, add or delete files in the root file-system (rootfs) or the root of the CD (rootcd) and regenerate the distribution. To clean and regenerate the distribution:

```
# tazlito clean-distro
# tazlito gen-distro
```

6.4.1.5 Packages Used

To create your distro, you need SliTaz packages. You can easily create your own packages with **Tazpkg**, or recreate packages from a SliTaz system in operation with **Tazlito**. By default, SliTaz packages are located in the `/home/slitaz/packages` directory but you can change this in the **Tazlito** configuration file (see *the bottom of this page* (page 350) for more information).

If you want to retrieve packages manually, you can use FTP software such as **gFTP** (installed by default on SliTaz) or the command line and **wget**. The direct URL to package versions is: `ftp://download.tuxfamily.org/slitaz/packages/`

You can also use your own packages by putting them in the `packages` directory. For more information on creating your own SliTaz packages, see the *Cookbook* (page 369).

6.4.1.6 Configuration File

Tazlito uses the `tazlito.conf` located in the current directory, if it exists, or defaults to the system configuration file `/etc/tazlito/tazlito.conf`. This means that you can use the default file or a configuration file specific to the distro you want to create.

This file specifies a number of properties:

- the name of the ISO image (`ISO_IMAGE`)

- the label used when mounted (VOLUME_NAME)
- the name of the maintainer (PREPARED)
- the paths to the working directories:
  - where the distro tree is located (DISTRO); default: /home/slitaz/distro
  - any packages that may be installed (PACKAGES_REPOSITORY); default: /home/slitaz/packages
  - the files that should be added to the built LiveCD (ADDFILES); default: $DISTRO/addfiles
- options for running **Tazlito** inside QEMU (QEMU_OPTS)
- the compression algorithm used on the root file-system (COMPRESSION):
  - lzma (default)
  - gzip
  - bzip2

To create and configure your own configuration file, you just need to run **Tazlito** with the configure command from the working directory of the distro. Assuming you have the tree of the distro in /home/slitaz/distro and you're working from it, use the command:

```
$ tazlito configure
```

Once you've answered the questions, you can either create the ISO image, rebuild the root file-system and ISO, or generate a new distro with the list of packages.

Tip: If you generate many LiveCDs, including the **date** command in the ISO name may be beneficial. To do this, use a line such as:

```
ISO_NAME="MyLiveCD-`date +%Y%m%d-%H%M`"
```

6.4.1.7 Advanced: Creating a Flavor

Tazlito allows you to easily create your own flavor from the results of generating the distribution (gen-distro). The flavor will contain all the additional files, a description and a list of packages (which can be reused and updated later, according to their versions). To generate your own flavor responding to one or two questions:

```
# tazlito gen-flavor new-flavor
```

Once your flavor is fully-functioning and well-tested, you can send it to SliTaz to make it available to everyone! It will then be listed via list-flavor and usable via get-flavor. You can send a flavor by several ways:

- announce through a post on the Mailing List[230];
- save the file on-line and report on the Forum[231];

[230] http://www.slitaz.org/en/mailing-list.html
[231] http://forum.slitaz.org/

- send an e-mail to a SliTaz contributor.

A contributor with sufficient privileges will then publish your flavor on the SliTaz servers.

6.4.1.8 Installing Tazlito on Another Distribution

Tazlito does not generate dependencies for the LiveCD, but it depends on **Cdrkit** to burn the image and **Tazpkg** to regenerate SliTaz packages you want installed. To retrieve and install **Tazlito**, follow the instructions in the `README` file in the sources of Tazlito[232].

6.4.2 LiveUSB media

author jozee, linea, totoetsasoeur, ceel, mojo, brianperry

6.4.2.1 Tazusb — Live USB Tool

Tazusb is a tool that can create bootable USB media with a few simple commands. It can also compress and backup the entire current filesystem to the media, thus preserving all modifications for future use. Type **tazusb usage** for a full list of commands or read the TazUSB manual[233].

Generate your own LiveUSB media

To generate your own LiveUSB media, first you need to login as *root* with **su** and locate your USB storage device using the command:

```
# fdisk -l /dev/sdxx
```

Then format and label your device (Warning: this will erase all data, make sure you specify the right device):

```
# tazusb format /dev/sdxx
```

And finally generate your LiveUSB media device with either:

```
# tazusb gen-liveusb /dev/sdxx
```

(if you're working with a LiveCD as the install source)

or

```
# tazusb gen-iso2usb slitaz.iso /dev/sdxx
```

(if you're working with an ISO file as the install source)

Note that the `/dev/sdxx` part of the command above specifies the location to where you need to write the ISO's files to; it is not the source media where the ISO is on! With the ISO file as install source option, you should hence specify the exact location where the slitaz ISO source file is located. If it is located at `/home/tux` (while running SliTaz from a non-live version), write

[232] http://download.tuxfamily.org/slitaz/sources/tazlito/
[233] http://hg.slitaz.org/tazusb/raw-file/tip/doc/tazusb.en.html

```
# tazusb gen-iso2usb /home/tux/slitaz.iso /dev/sdxx
```

If the source file is found on an external hard disk, mention the folder in which the media's files are outputted to — check this with **PCManFM** (in practice the folder can be something like `/media/disk`). So, the command you'd need to write with the latter would be something like

```
# tazusb gen-iso2usb /media/disk/slitaz.iso /dev/sdxx
```

And that's all you need to do, you can now boot SliTaz from USB media (providing your computer BIOS supports this method).

Tazusb Manual

The official **Tazusb** manual can be found in the SliTaz Web site documentation[234]; this contains a lot more useful information. The manual is also available through the documentation menu on the LiveCD.

6.4.2.2 Install SliTaz on USB from Windows

SliTaz can be installed on USB media from windows. The following description has been tested on Windows XP and Vista. Before starting you need to download:

1. the latest SliTaz iso from Get SliTaz[235].
2. the syslinux tool for the SliTaz version you install:
   - syslinux-4.05.zip[236] for SliTaz 4.0
   - syslinux-3.82.zip[237] for SliTaz 3.0
   - syslinux-3.73.zip[238] for SliTaz 2.0
   - syslinux-3.61.zip[239] for SliTaz 1.0

1. Extract the SliTaz ISO file to the USB stick. For this, use a tool like 7-Zip[240] or Winimage (shareware). Once extracted, you should get the following:

   ```
   \boot
   \boot\isolinux
   \boot\grub
   ```

   Note: Depending on the version of SliTaz you install, you can find on root other directories and files but they are not used by the LiveUSB.

2. On your USB stick, rename the `\boot\isolinux` directory to `\boot\syslinux`.
3. In the new directory `\boot\syslinux`, rename `isolinux.cfg` file to `syslinux.cfg` and `isolinux.msg` file to `syslinux.msg`. This is not mandatory but makes more sense.

[234] http://hg.slitaz.org/tazusb/raw-file/tip/doc/tazusb.en.html
[235] http://www.slitaz.org/en/get/#stable
[236] https://www.kernel.org/pub/linux/utils/boot/syslinux/4.xx/syslinux-4.05.zip
[237] https://www.kernel.org/pub/linux/utils/boot/syslinux/3.xx/syslinux-3.82.zip
[238] https://www.kernel.org/pub/linux/utils/boot/syslinux/3.xx/syslinux-3.73.zip
[239] https://www.kernel.org/pub/linux/utils/boot/syslinux/3.xx/syslinux-3.61.zip
[240] http://www.7-zip.org/

4. Now extract the syslinux archive previously downloaded and copy the `syslinux.exe` file from its `win32` directory into the `\boot\syslinux` directory of the USB stick.

5. Modify the `\boot\syslinux\syslinux.cfg`: replace all `isolinux.*` appellations with `syslinux.*`.

   - example for SliTaz 4.0:

     ```
     ...
     kernel /boot/isolinux/ifmem.c32
     ...
     ```

     to

     ```
     ...
     kernel /boot/syslinux/ifmem.c32
     ...
     ```

   - example for SliTaz 3.0 and previous versions:

     ```
     display syslinux.msg
     label slitaz
       kernel /boot/bzImage
       append initrd=/boot/rootfs.gz rw root=/dev/null lang=C␣
     ↪kmap=uk vga=normal autologin

     include common.cfg
     ```

   Note: In the example above, compared to the original file, the line `display isolinux.msg` has been replaced by `display syslinux.cfg` and the parameters `lang=C kmap=uk` have been added to the line `append` so that, lang and keyboard are automatically selected at boot.

   In file `\boot\syslinux\common.cfg`, modify

   ```
   ...
   F3 isolinux.msg
   ...
   ```

   to

   ```
   ...
   F3 syslinux.msg
   ...
   ```

6. Now, from the terminal in Windows, run **`syslinux.exe`** to install the *bootloader*. For example, here the USB stick is shown as letter K, type:

   ```
   k:
   cd \boot\syslinux
   syslinux -ma -d \boot\syslinux k:
   ```

7. You can now restart your computer. Change option in BIOS to boot first from USB.

6.4.2.3 See also

Live USB (page 126)

6.4.3 Hacking SliTaz LiveCD

author jozee, linea, brianperry

6.4.3.1 Introduction

Hacking SliTaz LiveCD or how to have fun with the LiveCD ISO image. Note that you can also create a custom flavor with **Tazlito**. Creating your own bootable ISO image is easily achievable and the steps are carefully described here. The manipulation of a personal ISO image can add new files or modify existing ones found on the SliTaz Live CD. The SliTaz ISO image is less than 30 MB and a CD-R or CD-RW provides around 700 MB, so there's plenty of scope for expansion. For example, you could store your images and even provide a live slideshow using **GQview**. The hacking of the ISO image allows you to modify boot loader configuration files (boot loader), splash images and **GRUB** itself. You could also add the **Memtest86** utility (tool used to test system RAM). Using the same techniques it's even possible to modify the filesystem — this does however require some extra manipulation and a bit more time.

6.4.3.2 Organization and preparation

To begin, first we must define where we are going to work by creating a directory and several sub directories to accommodate all the different files. The hacking of the ISO can be done from within a SliTaz system or any other GNU/Linux distribution such as Debian, Fedora, PCLinuxOS, etc. If you use SliTaz LiveCD mode (where you can remove the CD once SliTaz has launched in RAM and burn your new ISO), it's advisable to use USB media to carry on working, otherwise your work will be lost on shutdown. To begin you need to create a hacking directory that you can use inside `/home/slitaz` within the root of your user space. The use of a `/home/slitaz` directory enables you to store an original ISO image and gives you the option to create a `src/` directory to download possible source packages. All the various stages of hacking can be done on the command line via a X terminal (**Xterm**) or in console mode on a Linux terminal. It's advisable to run all commands as root to avoid any permission problems. To become the (root) adminsistrator, create a `/home/slitaz/hacked` directory and proceed inside:

```
$ su
# mkdir -p /home/slitaz/hacked
(# mkdir -p /home/slitaz/src)
# cd /home/slitaz/hacked
```

Getting the contents of the ISO

Now that you are in the working directory, we must create the root of the amended CD-ROM and retrieve the files contained on the original SliTaz ISO — namely, the Linux Kernel (`bzImage`), the compressed filesystem (`rootfs.gz`) and the `isolinux` bootloader files. To recover these files you have two options, either take them from a burned CD or from an ISO image stored locally. To create the root of your CD (`rootcd`) and copy files from the CD-ROM device `/dev/cdrom` mounted on `/media/cdrom`:

```
# mount -t iso9660 /dev/cdrom /media/cdrom
# mkdir rootcd
# cp -a /media/cdrom/* rootcd
```

To mount an ISO image using loop in the temporary directory `/tmp/loop` (with the ISO image `slitaz-cooking.iso`), create the root of the CD (`rootcd`), copy all the files and dismount the ISO image:

```
# mkdir /tmp/loop
# mount -o loop slitaz-cooking.iso /tmp/loop
# mkdir rootcd
# cp -a /tmp/loop/* rootcd
# umount /tmp/loop
```

Voilà, all the necessary files should now be present in the `rootcd/` directory. To be sure, you can list all of the files recursively with the **`ls`** command:

```
# ls -R rootcd
```

6.4.3.3 Adding the files to the ISO

The addition of various files and directories to the ISO image simply consists of copying data to the root of the CD-ROM (`rootcd/`) and generating a new image. The data may be classified in one or two directories created in the root of the CD. Once the ISO image is burned to a CD-R/CD-RW, you can use SliTaz as before, mounted on `/media/cdrom` and navigate through your data using **`emelFM2`**, **`Clex`** or the command line. Your data will also be legible from all GNU/Linux systems, BSD or even . . . Windows.

Create directories and copy data

To create and copy files, you can start by using the command line and then continue on graphically as a simple user. We will create an `images/` directory as root and change the permissions so that all users have write access:

```
# mkdir rootcd/images
# chmod 777 rootcd/images
```

Now that a directory exists that anybody can write to, you can start to fill it. Once you've finished you can then generate a bootable ISO image.

6.4.3.4 Modify the `isolinux` configuration

The modification of **`isolinux`** allows you to create custom entries with pre-boot parameters, for example you can add a label launching SliTaz with the `lang=en` and `kmap=en` options. At the design level you can easily change the splash image displayed at startup. The **`isolinux`** application manages the starting of the boot loader of the LiveCD and is provided by the **`Syslinux`** package. The source file of **`Syslinux`** provides various applications whose role it is to start a GNU/Linux system. The binary `isolinux.bin` controls the actual boot loading. The boot loader is simple, fast and easily configured either graphically or using a text editor. The syntax of the configuration file `isolinux.cfg` is easy to understand — to add new entries just copy and paste using the original file. To edit the file graphically using **`Leafpad`**:

```
# leafpad rootcd/boot/isolinux/isolinux.cfg &
```

Configuration file `isolinux.cfg`

The `isolinux.cfg` file found on the standard LiveCD of SliTaz, begins with the value `display`, this will either display a text file or a (`isolinux.msg`) file using 24 ASCII characters and a splash image. The default value defines the name of the label started by default after the (`timeout`) waiting time. Timeout is the number of seconds to wait before booting the system, you can make it 0 to start booting immediately or choose a waiting time as long as 80s. Finally the prompt can be deactivated using the value 0. `F1`, `F2`, `F3` display help files and `F4` displays a text file. Example configuration:

```
display isolinux.msg
default slitaz
label slitaz
      kernel /boot/bzImage
      append initrd=/boot/rootfs.gz rw root=/dev/null vga=788
implicit 0
prompt 1
timeout 80
F1 help.txt
F2 options.txt
F3 isolinux.msg
F4 display.txt
```

Example of a label `slitazen` which you can add to the original to directly configure the language of the system as English and use the UK keyboard:

```
label slitazen
      kernel /boot/bzImage
      append initrd=/boot/rootfs.gz rw root=/dev/null lang=en kmap=en
```

Once you've finished modifying the configuration file, don't forget to save your changes and generate a bootable ISO image with **`isolinux`**.

6.4.3.5 Install and use `Memtest86`

The application **`memtest86`** (92 kB) is a tool for testing your system memory (RAM). **`Memtest86`** performs indepth tests, that if failed, point heavily towards a hardware fault. The tool resides in the `boot/` directory and can be launched directly by typing `memtest` at the **`isolinux`** boot prompt. Navigate to `/home/slitaz/src` (if the directory doesn't exist: **`mkdir -p /home/slitaz/src`**), download the source and unpack:

```
# cd /home/slitaz/src
# wget http://www.memtest86.com/memtest86-3.3.tar.gz
# tar xzf memtest86-3.3.tar.gz
```

On unpacking the source of the **`memtest86`** package you'll find a `README` providing information about the tool. Now you can install into the root CD of your hacked ISO. Based on the premise that you'll be working with a `/home/slitaz/hacked` directory, we will copy the binary you precompiled into the `boot/` directory of the root of the CD:

```
# cp memtest86-3.3/precomp.bin \
  /home/slitaz/hacked/rootcd/boot/memtest
```

Now that the binary is installed in the root CD, we can just add an entry for **`memtest86`** to the **`isolinux`** configuration file and generate a bootable ISO image. Navigate to `/home/slitaz/hacked` and edit `isolinux.cfg` using **`Leafpad`**:

```
# cd /home/slitaz/hacked
# leafpad rootcd/boot/isolinux/isolinux.cfg &
```

Adding the following lines:

```
label memtest
      kernel /boot/memtest
```

Official website of Memtest86[241]

6.4.3.6 Manipulate the Live root system

Changes to the Live root system allow you for example, to add a new user and password, customize graphics or execute commands automatically at boot time. The necessary operations for changing the root file system are: extract the compressed file system `rootfs.gz`, modify, rebuild the image and generate the ISO. Based on the assumption that you've prepared a working directory, we begin by creating a directory to contain the files on the changed system. Since the compressed root file system is named `rootfs.gz`, we suggest you use `rootfs/` to extract to. Navigate to the `hacked/` directory, create the root directory and copy the compressed file system from `rootcd/boot/` (the root of the CD-ROM):

```
# cd /home/slitaz/hacked
# mkdir rootfs
# cp rootcd/boot/rootfs.gz rootfs
```

Now that you have the compressed copy of the system, just unpack with **`cpio`**. Technically `rootfs.gz` is a **`cpio`** file compressed with **`lzma`** or **`gzip`**. It's recognized like an initramfs image by the Linux Kernel. At the start up of the machine, the Kernel is loaded into memory and then decompresses the system image and carries out the initialization scripts.

To extract the file system into `rootfs/` and delete the unarchived copy (remember you can copy & paste):

```
# cd rootfs
# (zcat rootfs.gz 2>/dev/null || lzma d rootfs.gz -so) | cpio -id
# rm rootfs rootfs.gz
```

The system is now ready to be hacked, you can list all files at the root of your system by using the **`ls`** command.

Modify a file

To keep things simple and to help you understand the principle, we are going to change a script file in order to execute some commands to be carried out automatically when the CD starts up. The target is

[241] http://www.memtest86.com/

`etc/init.d/local.sh` — just open with your favorite text editor such as **Geany**:

```
# geany etc/init.d/local.sh &
```

We'll add a command displaying a message and letting the system sleep for 4 seconds. Example using local script:

```
echo "* Hacked SliTaz version booting..."
sleep 4
```

Rebuilding the image of the compressed system

Once the changes are completed, you can rebuild a compressed image of your system by using **find** to find the files, **cpio** for archiving, **lzma** and **gzip** for compression and the pipe **|** to connect everything together. This command must be launched from the root system (`rootfs/`) and creates a compressed file `rootfs.gz` in the preceding directory:

```
# find . -print | cpio -o -H newc | lzma e -si -so > ../rootfs.gz
```

Or with **gzip**:

```
# find . -print | cpio -o -H newc | gzip -9 > ../rootfs.gz
```

Finally copy the compressed file system into the `boot/` directory at the root of the CD and generate a bootable ISO image with **isolinux**. To copy the newly compressed rootfs into the working directory:

```
# cd ../
# cp -a rootfs.gz rootcd/boot
```

6.4.3.7 Generate a bootable ISO image

The following commands create an image with the boot loader **isolinux**, using the **genisoimage** application and a few options. The name of the ISO is specified at the beginning, after the `-o` option and the root directory (`rootcd/`) at the end, after the `-boot-info-table` option:

```
# genisoimage -R -o slitaz-hacked.iso -b boot/isolinux/isolinux.bin \
  -c boot/isolinux/boot.cat -no-emul-boot -boot-load-size 4 \
  -V "SliTaz-Hacked" -input-charset iso8859-1 -boot-info-table rootcd
```

If you want to check the contents of the ISO before burning, just mount the image in loop and list the files. On SliTaz and most GNU/Linux systems, you can burn images in ISO format with the **wodim** utility.

Generate a new ISO via a script

If you wish to test out a lot of new possibilities and generate a lot of ISO images, you may want to semi-automate the process via a simple SHell script. This tiny script can be created on the command line or edited graphically, but don't forget to make it executable. You can create the script with **cat**; note that `EOF` signifies *End Of File*. To create the script `gen_hacked_iso.sh` using two variables to change the name of the ISO image and the path to the root directory of the CD-ROM:

```
# cat > gen_hacked_iso.sh << "EOF"
```

```
#!/bin/sh
# Gen a new hacked ISO image.
#
ISO_NAME="slitaz-hacked.iso"
ROOTCD="rootcd"

genisoimage -R -o $ISO_NAME -b boot/isolinux/isolinux.bin \
   -c boot/isolinux/boot.cat -no-emul-boot -boot-load-size 4 \
   -V "SliTaz-Hacked" -input-charset iso8859-1 -boot-info-table $ROOTCD

EOF
```

To use the script, just make it executable and execute:

```
# chmod +x gen_hacked_iso.sh
# ./gen_hacked_iso.sh
```

6.4.3.8 See also

- *Persistence & splash* (page 146) — Guide on adding persistence

6.4.4 Chroot environment

author jozee, linea, domcox

This document describes the necessary steps to create a chrooted environment, in order to change the root of the system so that you can work. This makes it possible to compile, test and develop SliTaz without any risk to the host system you're working on. The host system can be SliTaz installed to a hard drive or any other GNU/Linux system such as Debian, Fedora, PCLinuxOS and so on. You can also create a chrooted environment in LiveCD mode associated with USB media. The only prerequisite is to have a SliTaz ISO image available and a little time. Note that all commands are carried out as system administrator (root).

6.4.4.1 Prepare the environment

To begin, we must extract the contents of the ISO image into the directory that will serve as our chroot. The directory can be created any place you choose, we'll use a directory `/home/slitaz/chroot-env`. To extract the contents of an ISO image, we must mount it in a loop directory and then copy the compressed root filesystem (`rootfs.gz`) into the chroot directory. Assuming the ISO is in the current directory:

```
# mkdir /tmp/loop
# mount -o loop slitaz-cooking.iso /tmp/loop
# mkdir -p /home/slitaz/chroot-env
# cp /tmp/loop/boot/rootfs.gz /home/slitaz/chroot-env
# umount /tmp/loop
```

Now we have a copy of the compressed filesystem, we must extract and unpack it (this is a **cpio** archive compressed with either **gzip** or **lzma**). To complete this stage, we can remove the rootfs which is no longer required:

```
# cd /home/slitaz/chroot-env
# (zcat rootfs.gz 2>/dev/null || lzma d rootfs.gz -so) | cpio -id
# rm rootfs rootfs.gz
```

If the unpacking of the rootfs compressed with **lzma** fails; you can use the following method:

```
# unlzma rootfs.gz -S .gz
# cat rootfs | cpio -id
```

6.4.4.2 Using the environment

To begin using the chrooted environment, you just need to mount some virtual filesystems and use the **chroot** command. To simplify things, we can write a small script automating the process. Example using the chroot directory `/home/slitaz/chroot-env` and creating a script `chroot_in_env.sh` in `/home/slitaz`. On any systems other than SliTaz you can uncomment the lines about `/dev` and `/tmp` — Note to save typing you can copy and paste:

```
# cat > /home/slitaz/chroot_in_env.sh << "EOF"
```

```
#!/bin/sh
# Chroot in SliTaz to hack.
#
ROOTFS="/home/slitaz/chroot-env"

# Mount virtual Kernel file systems and chroot.
#
#mount --bind /dev $ROOTFS/dev
#mount --bind /tmp $ROOTFS/tmp
mount -t proc proc $ROOTFS/proc
mount -t sysfs sysfs $ROOTFS/sys
mount -t devpts devpts $ROOTFS/dev/pts
mount -t tmpfs shm $ROOTFS/dev/shm

echo "Chrooting into $ROOTFS... "
chroot $ROOTFS /bin/sh --login

# Unmount virtual Kernel file systems on exit.
#
umount $ROOTFS/dev/shm
umount $ROOTFS/dev/pts
umount $ROOTFS/sys
umount $ROOTFS/proc
#umount $ROOTFS/tmp
#umount $ROOTFS/dev

echo "Exiting $ROOTFS chroot environment... "

EOF
```

To finish and test the environment, you just make the script executable and run:

```
# chmod +x /home/slitaz/chroot_in_env.sh
# sh /home/slitaz/chroot_in_env.sh
```

To activate the network

In order to have the network up to download and install some development packages, just start the DHCP client on the correct interface. Example using `eth1`:

```
# udhcpc -i eth1
```

Installing packages

If the network is functional, just reload the list of packages and use **`tazpkg get-install`** to install them. If a connection is not possible, you can download the packages from another system, copy them to the chrooted environment and install them with the **`tazpkg install`** command. To install the basic compilation tools:

```
# tazpkg recharge
# tazpkg get-install slitaz-toolchain
```

Once the environment is configured, you can compile applications from source to create packages, test scripts, etc. The Cookbook should help you out here:

Exit the environment

To exit the chrooted environment, just type **`exit`**, the `chroot_in_env.sh` script will then end by unmounting the virtual filesystems from the Linux Kernel:

```
# exit
#
```

6.5 Server Applications

6.5.1 Server applications

author pankso, linea

6.5.1.1 Dokuwiki

Dokuwiki[242] is a light and powerful Wiki engine using **`PHP`** and flat files as a backend; so no database server is needed. To use **`Dokuwiki`** you just need to install a web server (**`lighttpd`** or **`apache`**) and **`PHP`**.

[242] http://www.dokuwiki.org/

6.5.1.2 Drupal

`Drupal` is a powerful CMS (Content Management System) using **`PHP`** server side language and a MySQL database. To install **`Drupal`** and have it running in a few minutes you must first install the web server (**`lighttpd`**), **`PHP`** and **`MySQL`**:

```
# tazpkg get-install lighttpd
# tazpkg get-install php
# tazpkg get-install mysql
```

Download the latest version from http://drupal.org/ and create a virtual host or go into your public directory and untar the **`Drupal`** sources:

```
$ mkdir ~/Public && cd Public
$ tar xzf drupal-*
```

Now we have to modify some file permissions so that **`Drupal`** can write to the filesystem during the installation process and when running:

```
$ cd drupal-*/sites
$ chmod 777 default
$ cp default/default.settings.php default/settings.php
$ chmod 666 default/settings.php
```

Create a MySQL database and then use your favorite web browser to install **`Drupal`** through the web interface. When the installation process has ended you must modify file permissions and can start to customize the application. Change permissions for production and clean-up:

```
$ cd ..
$ chmod 755 sites/default
$ chmod 644 sites/default/settings.php
```

To ensure easy upgrades of your **`Drupal`** core installation you should put all additional modules and themes into the directory: `site/all/`. So to prepare the addition of any future modules and themes:

```
$ mkdir sites/all/modules
$ mkdir sites/all/themes
```

6.5.1.3 Online

- Drupal Modules[243]
- Drupal Themes[244]

6.5.2 LightTPD Web Server

author jozee, linea, gokhlayeh, trixar_za

[243] http://drupal.org/project/modules
[244] http://drupal.org/project/themes

6.5.2.1 About LightTPD

This chapter describes the configuration and use of the LightTPD web server. It's a fast, secure, flexible HTTP server, using a small memory footprint. It enables intelligent management of the CPU load and offers FastCGI support, CGI, Auth, Output compression and the rewriting of URLs, etc. LightTPD is a cheap way to host your own site on an old machine.

On SliTaz the server is automatically launched at system startup and is preconfigured with PHP. The root documents served by default are in `/var/www`, this contains the default page `index.html`, images are stored in the `images/` directory. LightTPD website: http://www.lighttpd.net/

6.5.2.2 `/var/www` — Root directory of documents

The `/var/www` folder is the root directory of documents — you can access this via the URL http://localhost/. If you want to host a site, you can save all your documents in here. If you want to host multiple sites, you'll need to create virtual hosts. Note you can also check the http://localhost/server-status.

6.5.2.3 `~/Public` — Public directory of users

SliTaz provides the users of the system a public space to place documents, HTML in general. This directory is named `Public` and must be within the root of your user space, such as `/home/`*`hacker`*`/Public`. To create this directory, use the **`mkdir`** command:

```
$ mkdir ~/Public
```

You can then have access via the URL: http://localhost/~hacker/. You can also use the machine name or IP address if you connect from another computer.

6.5.2.4 `/etc/lighttpd/lighttpd.conf` — LightTPD configuration file

The main configuration file for LightTPD (`lighttpd.conf`) is located in `/etc/lighttpd/`. This file provided by SliTaz is self-explanatory, just browse. You can find other examples on the LightTPD website. On SliTaz you'll also find a `vhosts.conf` file for the configuration of any virtual hosts (hosting several sites on the same server).

6.5.2.5 Start, stop, restart the web server

By default, SliTaz starts the server automatically at boot, to prevent this you need to remove `lighttpd` from the variable `RUN_DAEMONS` located in the system file `/etc/rcS.conf`. To start, stop or restart the server, you can use the commands: **`/etc/init.d/lighttpd [start|stop|restart]`**. Example to restart the server after changing the configuration file:

```
# /etc/init.d/lighttpd restart
```

6.5.2.6 CGI scripts using Perl

To configure the LightTPD server to locate the path of the perl binary and use CGI/Perl, you'll need to install **`perl`** and modify the server configuration file. Example using **`Geany`**:

```
# tazpkg get-install perl
# geany /etc/lighttpd/lighttpd.conf &
```

```
# CGI module. You can install Perl and assign .pl and .cgi scripts
# to /usr/bin/perl
$HTTP["url"] =~ "/cgi-bin/" {
  cgi.assign = (
    ".sh" => "/bin/sh",
    ".cgi" => "/usr/bin/perl",
    ".pl" => "/usr/bin/perl"
  )
}
```

6.5.2.7 CGI scripts using Python

To configure the LightTPD server to locate the path of the python binary and use CGI/Python, you'll need to to install **python** and modify the server configuration file. Example using **Geany**:

```
# tazpkg get-install python
# geany /etc/lighttpd/lighttpd.conf &
```

```
# CGI module. You can install Python and assign .py and .cgi scripts
# to /usr/bin/python
$HTTP["url"] =~ "/cgi-bin/" {
  cgi.assign = (
    ".sh" => "/bin/sh",
    ".cgi" => "/usr/bin/python",
    ".py" => "/usr/bin/python"
  )
}
```

For the changes to be taken into effect and to use your first CGI scripts on SliTaz, just restart the LightTPD server:

```
# /etc/init.d/lighttpd restart
```

6.5.2.8 Authentication — Protection for the directories

LightTPD provides authentication modules that can for example, protect a directory. The server offers several authentication methods, but we will begin by using the *basic* method without encrypting any passwords. In order to be able to use the module `mod_auth`, you must install the **lighttpd-modules** package (**tazpkg get-install lighttpd-modules**). Once installed `mod_auth` must be added to the list of modules:

```
# Modules to load.
# See /usr/lib/lighttpd for all available modules.
#
server.modules = (
  "mod_access",
  "mod_auth",
  "...",
)
```

Now you can configure the modules by specifying the debug level and method (`plain`) and the path to the file containing a list of names using a protected password to access the directories. You must also define the directories that require authorization. In this example we'll protect the `admin/` directory and authorize its access to user `tux` (`user=tux`):

```
# Authentication for protected directory.
auth.debug = 2
auth.backend = "plain"
auth.backend.plain.userfile = "/etc/lighttpd/plain.passwd"
auth.require = ( "/admin/" =>
  (
  "method" => "basic",
  "realm" => "Password protected area",
  "require" => "user=tux"
  )
)
```

Finally, we now create the file containing the passwords, add a user and restart the server for testing. The basic syntax for the file is `user:password`. You can create the file and add a user with the **echo** command or edit with your favorite text editor. To add `tux:root` to the password file `/etc/lighttpd/plain.passwd`:

```
# echo "tux:root" > /etc/lighttpd/plain.passwd
```

or:

```
# nano /etc/lighttpd/plain.passwd
```

To test the address http://localhost/admin/, just restart the server:

```
# /etc/init.d/lighttpd restart
```

6.5.3 Secure SHell (SSH)

author jozee, linea, bellard

6.5.3.1 About Dropbear

Control and administer remotely with the **Dropbear** SSH secure server. **Dropbear** is a small SSH client/server supporting SSH 2. It's compatible with **OpenSSH** and uses `~/.ssh/authorized_keys` for the management of public keys. **Dropbear** also provides its own version of **scp**, allowing you to copy files between machines in a secure manner.

Project website: http://matt.ucc.asn.au/dropbear/dropbear.html

6.5.3.2 Connecting to a remote host with `dbclient`

The configuration files for the SSH client are located in the `~/.ssh` directory of each user, this contains the `authorized_keys` and `known_hosts` files. The directory `~/.ssh` and `known_hosts` file are automatically created the first time you run the **Dropbear** client (**dbclient**).

To connect to a remote host employing the user and machine name:

```
$ dbclient user@machine.org
```

You can also connect using the IP address of the machine:

```
$ dbclient user@192.168.0.2
```

6.5.3.3 Transfer of remote files with `scp`

To copy a file from one computer to another, **`scp`** can be utilized in the following ways. To copy a file named `page.html` to a remote directory of the user (don't forget the **`:`** after the machine name or IP address):

```
$ scp page.html user@machine.org:path/remote/directory
```

Copy a file from a remote machine to your local machine:

```
$ scp user@machine.org:path/remote/directory/page.html /path/your/directory
```

6.5.3.4 Generate RSA/DSS keys with `dropbearkey`

`Dropbear` provides **`dropbearkey`** to generate the protected RSA and DSS keys. Note that when you start the server for the first time, secure keys will be automatically generated if they don't already exist. You can use **`dropbearkey`** with the following arguments:

```
# dropbearkey -t rsa -f /etc/dropbear/dropbear_rsa_host_key
# dropbearkey -t dss -f /etc/dropbear/dropbear_dss_host_key
```

6.5.3.5 Start, stop, restart the SSH server

By default SliTaz will not start the SSH server at boot. To be launched automatically, `dropbear` must be added to the variable `RUN_DAEMONS` in the `/etc/rcS.conf` file. To start, stop or restart the SSH server, use the following commands: **`/etc/init.d/dropbear [start|stop|restart]`**. Example to start the server:

```
# /etc/init.d/dropbear start
```

Note that the server supports the passing of various options when launched. To change the default values, simply modify the daemons configuration file `/etc/daemons.conf`.

6.5.3.6 Dropbear and the X server

`Dropbear` supports X11 tunneling on the server side only. The client **`dbclient`** has no support for X11 tunneling. SliTaz provides a tiny shell script named `/usr/bin/sshx` to do the work. It opens a terminal with remote X11 protocol support if the environment variable `DISPLAY` is set. You can launch any remote X windows application on this terminal.

6.5.3.7 Dropbear and VNC

SliTaz provides a tiny VNC client named `/bin/fbvnc`. The VNC connections are not secure (neither encryption nor authentication). A tiny shell script named `/usr/bin/sshfbvnc` move the VNC connection to a SSH tunnel ending at the localhost interface of the remote VNC server. You will have an encrypted connection and authentication for your VNC sessions.

CHAPTER 7

Cookbook

author jozee, int3, seawolf, linea, gokhlayeh, pankso, brianperry, bellard

The Cookbook brings together information about the project management, operation and development of the distribution. It talks about creating packages, receipts, the wok, and scripts that start SliTaz.

At the base of the Cookbook is the *Scratchbook* (page 397), this contains instructions to create your own LiveCD by describing the creation of the first ever public version of SliTaz in March 2007. The Cookbook is modified by the SliTaz community and steadily improved, it provides technical instructions about the project useful to developers and advanced users.

Table of contents

- *Contributors* (page 395) — Contributors access to the SliTaz servers.

7.1 Recipes

author jozee, int3, linea, pankso, godane, bellard, hgt

This document describes the opportunities offered by the recipes used by **Cookutils** to compile and generate packages for SliTaz and **Tazpkg** through the wok and tools. The recipe for a package is also used by **Tazpkg** to install/uninstall and provide information about a `.tazpkg` package. Each recipe begins with a comment in English:

```
# SliTaz package receipt
```

7.1.1 Variables

The first 5 variables should always be present and defined. They respectively configure the package (`$PACKAGE`), its version, its category, provide a short description and the name of the maintainer. Example for the package, file manager Clex:

```
PACKAGE="clex"
VERSION="3.16"
CATEGORY="base-apps"
SHORT_DESC="Text mode file manager."
MAINTAINER="pankso@slitaz.org"
```

7.1.2 Variables (optional)

Cookutils also knows how to use various optional variables. It can, for example, use the name of another source package. There are also variables that are used by **Tazpkg** to manage dependencies or provide information about the package.

$DEPENDS: Set dependencies, there may be several dependencies separated by a space or on several lines. This variable is used mainly by **Tazpkg** when installing the package and **Cookutils** to build large packages such as **Xorg**. Example for **Clex** which depends on **ncurses**:

```
DEPENDS="ncurses"
```

$BUILD_DEPENDS: Set compilation dependencies, again separated by a space or several lines. This variable is used by **Cookutils** during the cooking of a package. Example:

```
BUILD_DEPENDS="ncurses-dev"
```

$TARBALL: The archive is a source with the extension (`tar.gz`, `tgz` or `tar.bz2`). In general, the variables `$PACKAGE` and `$VERSION` are used to just change the extension, it helps to upgrade the package without changing the `$VERSION` variable. Generic example (see also `$SOURCE` example):

```
TARBALL="$PACKAGE-$VERSION.tar.gz"
```

$WEB_SITE: The official website of the package. It may be that some libraries have no website, in this case, there is no need to specify a URL. Note **Cookutils** and **Tazpkg** both expect to find a URL with the complete HTTP:

```
WEB_SITE="http://www.clex.sk/"
```

$WGET_URL: URL to download the source file. In general the variable `$TARBALL` should be used to facilitate the updating of the package without changing the `$VERSION`. Using a configuration file, **Cookutils** also configures by default 3 mirrors: `$GNU_MIRROR` for the GNU mirror, `$SF_MIRROR` for SourceForge and `XORG_MIRROR` for mirroring the graphical server **Xorg**. Example for **Clex**:

```
WGET_URL="http://www.clex.sk/download/$TARBALL"
```

$CONFIG_FILES: Some packages provide customized configuration files. The `$CONFIG_FILES` variable provides a list of these files that can be saved by the **tazpkg repack-config** command. These files are not overwritten when reinstalling the package if they already exist and the package can be successfully recreated with **tazpkg repack**, (even if they have been modified since). **Netatalk** for example:

```
CONFIG_FILES="/etc/netatalk/AppleVolumes.* /etc/netatalk/*.conf"
```

$SUGGESTED: Lists useful packages without being essential. Also used to activate optional features.

$WANTED: SliTaz packages normally depend on the compilation of a source package. Sometimes the recipe of a package requires no compilation of rules, then `$WANTED` is used to copy files from the source of another package by using the variable `$src`.

$SOURCE: It may be that the **Tazpkg** package name differs from the name of the source package. Example for **Xorg** packages, the name of **Tazpkg** library **X11** is `xorg-libX11` and the name of the package source is `libX11`. `$SOURCE` allows you to use the variables `$src` and `$install` during the cooking of a package. It should be noted that in the case of `libX11`, the name of the source archive becomes `$SOURCE-$VERSION.tar.gz`.

$PROVIDE: Some packages offer the same functionality, for instance the web server was at first **lighttpd**; now **apache** is available. All packages dependent on a web server refer to **lighttpd**. The `PROVIDE="lighttpd"` variable in the **apache** recipe states that packages dependent on **lighttpd** do not need to install the **lighttpd** package if **apache** is already on the system. Some packages may vary according to the webserver you choose, ie. the **php** package is dependent on **lighttpd**, as is **php-apache** on **apache**. The `PROVIDE="php:apache"` in the **apache** recipe says that you must install **php-apache** instead of **php**, if **apache** is already on the system. Therefore each package dependent on **php** will install either **php-apache** or **php** according to the webserver on the system. This variable also chooses packages compiled with different options. The `PROVIDE="epdfview:cups"` in the **epdfview-cups** recipe allows you to install **epdfview** with printer support via **cups** if **cups** is already on the system.

You can also define virtual packages with this variable. The lines `PROVIDE="libgl"` in the **mesa** package and `PROVIDE="libgl:nvidia"` in the **nvidia-glx** package, define that **libgl** is an optimized version when the **nvidia** package is installed.

$SELF_INSTALL (obsolete): Certain packages use commands provided by the package itself in the `post_install` function. To install this package into a directory other than root and still be able to use these commands, the package must have been installed in / in earlier stages. The line: `SELF_INSTALL=1` alerts **tazpkg** to this feature. This variable is depreciated. The command `chroot "$1/" a_package_command` in `post_install` does the job.

7.1.3 Variables generated by Cookutils

Certain factors are known only during the cooking of a package or after the package has been cooked. **Cookutils** will add them to the recipe automatically.

$PACKED_SIZE: **Tazpkg** file size.

$UNPACKED_SIZE: Space taken up by the package after installation.

$EXTRAVERSION: Some packages have 2 different versions. This is in case of modules added to the Linux kernel, such as **squashfs**, because the module depends on the version of the kernel with which it was compiled. In this case `$EXTRAVERSION` contains the kernel version and **Cookutils** determines the module from the contents of `/lib/modules`.

7.1.4 Variables used in functions

Cookutils configures several variables that facilitate the compilation and construction of **Tazpkg** packages. These variables are controlled automatically by **cookutils** using the information contained in the recipe; they can be used by the functions `compile_rules` and `genpkg_rules` described in the chapter Functions.

$src: Defines the path to the directory of unarchived sources.

$install: Defines the path to the compiled binaries installed via **make DESTDIR=$DESTDIR install**. This variable is used to copy the generated files and create **Tazpkg** packages.

$_pkg: Same as `$install`.

$fs: Defines the path to the pseudo filesystem (fs) in each package. The `fs` of the package corresponds to the root of the system, a bit like **Clex** will for example be in `$fs/usr/bin/clex`. Note the need to create the necessary directories via function `genpkg_rules()` before copying the files.

$CONFIGURE_ARGS: This variable is defined in the **cookutils** configuration file (`cook.conf`). It allows you to specify generic optimization arguments during construction of a package. Default is the i486 architecture.

$DESTDIR: Defines the path to install compiled binaries after the build via **make DESTDIR=$DESTDIR install**.

7.1.5 Functions

A recipe may contain 4 functions. Cookutils knows how to deal with functions containing compilation rules (compile_rules) and rules used to generate a package (genpkg_rules). These functions may contain all sorts of GNU/Linux standard commands, such as sed, awk, patch and variables automatically configured.

compile_rules()

To compile a package you can use the variable `$src` to move (**cd**) in the directory of sources and use `$CONFIGURE_ARGS` to include arguments from the **Cookutils** configuration file. To build the package you usually launch **make** without any arguments, and to install the package into the directory `$DESTDIR`: it's necessary to use the command **make DESTDIR=$DESTDIR install**. Generic example:

```
# Rules to configure and make the package.
compile_rules()
{
     cd $src
     ./configure --prefix=/usr --infodir=/usr/share/info \
     --mandir=/usr/share/man $CONFIGURE_ARGS
     make
     make DESTDIR=$DESTDIR install (or simply make install)
}
```

`genpkg_rules()`

To generate a **`tazpkg`** package we must specify commands in the function `genpkg_rules`. In this example we create a pseudo directory `/usr` in the filesystem of the package, copy the whole binary(s) and finally use **`strip`** to clean the files:

```
# Rules to gen a SliTaz package suitable for Tazpkg.
genpkg_rules()
{
     mkdir -p $fs/usr
     cp -a $install/usr/bin $fs/usr
     strip -s $fs/usr/bin/*
}
```

`pre_install()` and `post_install()`

These functions are initiated by **`Tazpkg`** when installing the package. They must be defined before generating the `.tazpkg` package with **`Cookutils`**. If no rules are given for these functions, they have no raison d'etre and can be removed. Example using **`echo`** to display some text (no function should be empty):

```
# Pre and post install commands for Tazpkg.
pre_install()
{
     echo "Processing pre-install commands..."
}
post_install()
{
     echo "Processing post-install commands..."
}
```

`pre_remove()` and `post_remove()`

These functions are initiated by **`Tazpkg`** when removing the package. They must be defined before generating the `.tazpkg` package with **`Cookutils`**. If no rules are given for these functions, they have no raison d'etre and can be removed. Example using **`echo`** to display some text (no function should be empty):

```
# Pre and post remove commands for Tazpkg.
pre_remove()
```

(continues on next page)

(continued from previous page)

```
{
     echo "Processing pre-remove commands..."
}
post_remove()
{
     echo "Processing post-remove commands..."
}
```

7.2 Wok & Tools

author jozee, claudinei, linea, seawolf, hgt

7.2.1 Overview

Cookutils[245] is used to compile and generate code (*cooking*) via instructions found in a receipt. It places compiled files in to a directory and calls upon **`Tazpkg`** to package said directory. The receipt found in a wok has a different "bottom half" to that of a **`Tazpkg`**; **`Cookutils`** has rules for compilation *and* packaging (which it forwards to **`Tazpkg`**), whereas **`Tazpkg`** is only concerned with packaging.

`Cookutils` is one of many small utilities the SliTaz project uses to automatically rebuild the distribution from source. The project also offers an archive of *tools* (page 380) containing various small utilities, examples and configuration files. The distribution generator Tazlito[246] is designed for users and developers; it can retrieve and reconstruct a LiveCD ISO image and generate a distribution flavor from a list of packages, a configuration file and a description. The utilities are all distributed as a source archive and are installed by default on SliTaz.

Tip: Developers and future contributors can refer to the development[247] page that provides information on SliTaz project management.

7.2.1.1 Wok Structure and Organisation

The *wok* is a directory structure that houses all the available packages. Each directory contains at least one receipt to download, unpack, compile and generate a package. **`Cookutils`** also needs to create a directory to store downloaded sources (`$SOURCES_REPOSITORY`, usually `/home/slitaz/src`) and a repository of generated packages (`$PACKAGES_REPOSITORY`, usually `/home/slitaz/packages`); these values can be configured in the `/etc/slitaz/cook.conf` file.

There is more than one Wok on the Mercurial repositories[248]:

- **wok-undigest**: contributions awaiting testing/bug-fixing for inclusion in the unstable Wok
- **wok**: packages for the unstable, Cooking release
- **wok-stable**: packages for the stable SliTaz release

[245] http://hg.slitaz.org/cookutils/raw-file/tip/doc/cookutils.en.html
[246] http://hg.slitaz.org/tazlito/raw-file/tip/doc/tazlito.en.html
[247] http://www.slitaz.org/en/devel/forge.php
[248] http://hg.slitaz.org

Initially, any contributions will be committed to the *undigest* repository. When the package has seen sufficient testing with regards to automatic generation, it can be moved to the Wok.

7.2.2 Preparation

The Developer's Corner[249] provides invaluable background information. Please ensure you have read and understood it before continuing.

To begin using the Wok, Cookutils[250] must already be installed on the system along with the main development tools (**binutils**, compiler, libraries-dev, **make**). This requires you to install the meta-package **slitaz-toolchain**:

```
# tazpkg recharge
# tazpkg get-install slitaz-toolchain
```

To access the SliTaz repositories, you will need to install the **mercurial** package:

```
# tazpkg get-install mercurial
```

More information on the use of the **Mercurial** VCS is available from its website[251] and the "Hg Book[252]".

7.2.2.1 Cloning the Wok

If you are to generate a package for inclusion in the SliTaz repositories, it is necessary to first obtain the current wok by using **Mercurial**. This is called *cloning* the Wok, a procedure that downloads the entire Wok and all its history to a working directory. **If you wish to only use Cookutils to build packages for personal use, this is not necessary.** See the *Creating a Personal Wok* (page 376) section below instead.

The usual destination for a Wok clone is `/home/slitaz/wok`:

```
# cook setup --wok
```

This download may take some time; you will have a complete directory structure of the Cooking wok[253] as a working directory.

Important: The Wok is one of many projects hosted in the Mercurial repositories[254]. Individual packages are grouped as a large project (the Wok, Wok-Stable or Wok-Undigest) and is not its own sub-project but merely a sub-directory; Mercurial cannot (yet) clone specific parts of a project thus you cannot clone an individual package.

[249] http://www.slitaz.org/en/devel/forge.php
[250] http://hg.slitaz.org/cookutils/raw-file/tip/doc/cookutils.en.html
[251] http://mercurial-scm.org/
[252] http://hgbook.red-bean.com/
[253] http://hg.slitaz.org/wok/
[254] http://hg.slitaz.org

7.2.2.2 Creating a Personal Wok

If your packages are only for personal use and are not intended for inclusion in the SliTaz repositories, a wok can be created from scratch.

```
# cook setup
```

7.2.3 Compiling and Generating Packages

Before compiling your first package, **Cookutils** must know where your working directory is. By default the path is `/home/slitaz/wok` but you can change this or rename the wok that you want to download.

The process for generating a SliTaz package from source can be summarised thus: configure[255], compile[256] & strip[257].

Note: We do not carry out the '**make install**'-style step ourselves; the built files are not to be installed in the system but left in the output directory (`$DESTDIR`), ready for packaging.

When generating your first package, it is advisable to keep it simple[258] and build your package without changing its receipt or seeking dependencies. **M4** is an ideal candidate for your first *cook*:

```
# cook m4
```

When **Cookutils** has finished building **M4**, its package is placed in the directory specified by the configuration file (`/home/slitaz/packages` by default). If all went well, you can install the package on the host system or use it to generate a LiveCD distribution via **Tazlito**!

When you are familiar with *receipts* (page 370) and the compilation process, you can use the following command to create a new package, after interactively writing its receipt:

```
# cook new <packageName> --interactive
```

Be sure to read the documentation on the options provided by the *receipt* (page 370) and the *Tazwok Tips* (page 240) to avoid frustration!

7.2.3.1 Cooking Multiple Packages with a cook list

Cookutils can compile several packages with a single command. This is achieved with a *cooking list*, a text file of one package per line. **Cookutils** can accept a cook-list with the command of the same name; for example, to cook the *mypkgs* cook-list:

```
# cook list mypkgs.cooklist
```

[255] http://www.tuxfiles.org/linuxhelp/softinstall.html#s2
[256] http://www.tuxfiles.org/linuxhelp/softinstall.html#s3
[257] http://linux.die.net/man/1/strip
[258] http://doc.slitaz.org/en:cookbook:devcorner#kiss-comply-to-standards

7.2.4 Package Compilation Options

While you are free to use any options you want, it is necessary to respect the FSH, the documentation in `/usr/share/doc` and follow the FreeDesktop standards (`.desktop`).

7.2.4.1 Package-Specific

Package-specific options are your choice; for example, you can disable support for XML, have smaller binaries for PHP and get rid of libxml2, but in the case of PHP, it's not worth the cost in terms of loss of functionality. If you have any doubts, look at the receipts and compiler options in `compile_rules`.

7.2.4.2 Optimization

The official SliTaz packages are optimized for **i486**, the optimization arguments used to configure are specified in `/etc/slitaz/cook.conf` and can be called via the variable `$CONFIGURE_ARGS`. If you want to compile a package with different arguments, you can modify the **`Cookutils`** configuration file:

```
CONFIGURE_ARGS="--build=i486-pc-linux-gnu --host=i486-pc-linux-gnu"
```

7.2.4.3 Files to Include/Exclude

Generally, the base packages contain no `man`, `info` or `doc` files, nor static libraries; we have to create them via a `package-doc` or a `package-dev`. Note that SliTaz does not intend to use the **`man`** or **`info`** command so there's no manual or GNU info file. The creation of packages containing docs is really optional. By contrast, writing documentation in the Handbook is more appreciated as it is widely-available and can be updated and improved easily.

In terms of configuration, the aim is to offer basic configuration files to run the package directly. Special cases exist such as the web server **`LightTPD`**, for example, where SliTaz supplies configuration files and start-up scripts in `/etc/init.d` (documented in the Handbook). For a new package, you are free to choose its default configuration depending on what you think is best for the end-user. The `/usr/share/examples` directory has example configurations and other kinds of useful information.

7.2.5 Package Categories

The categories of packages exist only for informational purposes and are not fixed. The idea is to classify packages so that a web page that recovers data in the package receipt, can be generated each night. For the short term, place development packages in `devel`, Xorg in `x-window` and the variety of new packages in `extra`.

7.2.6 Structure of a Wok Package

The structure of the packages in the wok should always be respected so that **`Cookutils`** can find the correct files and directories. Possible contents of a package (note the directory `taz/`` is created at time of cooking):

`stuff/`: The material used to configure, compile and generate the package (patch(es), Makefile, pseudo fs, etc);

receipt: The ever-present *receipt* (page 370);

description.txt: (optional) The description of the package is included in the final package, copied to its root. Once installed, **Tazpkg** identifies this file as the description and can display it via **tazpkg desc pkgname**.

taz/: Directory tree containing the package **Tazpkg** generated, the compressed package is stored in the directory specified by `$PACKAGES_REPOSITORY` in the **Cookutils** configuration file.

Cookutils will automatically call upon **Tazpkg** to package the `taz` directory. It also forwards any packaging instructions found in the receipt.

7.2.7 Structure of a Tazpkg

The SliTaz packages are cpio archives containing files and a file-system compressed with gzip:

fs/: Pseudo-file-system containing all the files to install.

receipt: The *receipt* (page 370).

files.list: A list of files in the package.

description.txt: The description of the package (optional).

7.3 Build Bot — Cooker

author jozee, linea, pankso

See Cookutils documentation and http://cook.slitaz.org/

7.4 SliTaz Build Host (tank)

author jozee, linea, pankso, mojo, bellard

SliTaz build host info and howto.

7.4.1 Folders in `/home/slitaz`

- `cooking/` — Cooking chroot and flavors.
- `stable/` — Stable tree.
- `undigest/` — Undigest tree.
- `repos/` — All the project repos (where the commits are pushed).
- `www/` — Virtual hosts (`boot`, `people`, etc).

7.4.2 Using `tazdev`

To help maintain the mirror, flavors and other services, the **tazdev** command is used. It is mostly configured for the cooking version. Usage:

```
$ tazdev usage
```

7.4.3 Cooking official packages

Maintainers have root access and some have write access to the main mirror at mirror.slitaz.org, if you want to help in this task please contact one of the active developers (check Hg repos).

Everything is cooked in a chroot environment, the default path for the build wok is `/home/slitaz/cooking/chroot`. No changes must be done in the chroot and don't install any packages to keep it clean. We run the cooking chroot under `tty4`:

```
# conspy -f 4
```

Tip: Don't forget to update the chroot with **`tazpkg up -r`**

In the event of a reboot or trouble you may be prompted to a login, login as root then use **`tazdev chroot`** to chroot into the build env. To cook all the last committed packages or cook a single package:

```
#/ cooker
#/ cooker pkg pkgname
```

To get out of conspy press `Esc` 3:sup:× times, don't **`exit`** the chroot. If you have write access to the mirror, you can make a dry-push to check and then upload; push will also remove any old packages on the mirror.

7.4.4 Cooking undigest packages

On Tank we run the undigest cooker under tty6, using **`conspy`**:

```
$ su
# conspy -f 6
# cooker
```

or

```
# cooker pkg pkgname
```

7.4.5 Stable packages

Packages for the stable release are also built in a chroot environment like the Cooking packages:

```
# conspy -f 2
```

7.4.6 Upload by hand on mirror.slitaz.org

Mirror maintainers can upload by hand with **`tazdev`** (`-dp` for a dry push):

```
# tazdev -p $USER
```

7.4.7 Recreate a chroot

If a chroot seems broken or needs to be rebuilt. Exit the chroot and unmount the virtual system (`/home` is kept) as root. From the corresponding **`conspy`** session:

```
# tazdev clean-chroot
# tazdev gen-chroot
# tazdev chroot
#/ tazpkg recharge
#/ cook setup
#/ cooker
```

7.5 SliTaz Tools

author jozee, linea, llev

The SliTaz Toolbox

- Mercurial repository.
- **`*box`** — dialog/GTKdialog.
- Archives source: HTTP[259].

The SliTaz Tools contain useful scripts that enable you to customize SliTaz, such as a script to create a new initramfs or an ISO image, `Makefile`, etc. They accompany the *Scratchbook* (page 397) and help the SliTaz developers. The archive is also distributed because it contains files that might be useful to hacker type individuals… The tools are constantly evolving and continue to expand, following the cycle of changes made by the Cooking and Stable versions.

7.5.1 Mercurial repository

The SliTaz Tools have their own Mercurial repository on the SliTaz server, they can be cloned via the command:

```
$ hg clone http://hg.slitaz.org/slitaz-tools/
```

7.5.2 `*box`

`Mountbox`, **`Netbox`**, **`Bootfloopybox`**, **`Tazlocale`**, etc are tools for creating SliTaz using **`dialog`** (**`ncurses`**) or **`GTKdialog`**; the scripts are contained in the directory `tinyutils/` . **`Desktopbox`** is able to launch various boxes (**`desktopbox usage`**) scripted or created with **`Glade3`**.

[259] http://download.tuxfamily.org/slitaz/sources/tools/

7.6 Advanced usage of Mercurial

author gokhlayeh, linea, llev

7.6.1 Use an external tool to merge

If you want to use tools presented on this page, **MQ** particularly has a tool to manage merges (when several commits overwrite each other and you have to edit the result manually) which will probably be useful to you. SliTaz proposes **Meld**, a light software that can do that well. After installation, tell **Mercurial** to use it if necessary by putting it in your `~/.hgrc`:

```
[ui]
merge = meld
```

7.6.2 Useful extensions

To add an extension, you can use the `~/.hgrc` file:

```
[extensions]
name = adress
```

Some extensions are packed within **Mercurial**, so it's not necessary to give them addressess. It's the case of the four following:

color: Add color in **Mercurial**. Useful when displaying differences between several versions of a file.

hgext.fetch: Add the command **hg fetch**, which regroups **hg pull && hg merge && hg update**.

hgext.graphlog: Add the command **glog**, which displays the revision tree along with the log. It's advised to limit the length of the log with option `-l` (i.e.: `-l 10`). Option `-p` displays the detail of differences introduced at each commit.

mq: This tool is explained in detail below. It allows you to manage a patch-list for a Mercurial repository: apply them, unapply, update, etc. This extension adds several commands which generally start with 'q'. Some webpages detail this tool, search: `mercurial mq`.

7.6.3 Basic functionality of MQ

In a mercurial repository, create a patch repository with a controlled revision; it's a repository of patches into which the changes can be committed using **Mercurial**, like a repository within a repository:

```
hg qinit -c
```

After modifications, save them as a patch instead of committing them:

```
hg qnew nom_du_patch
```

List applied/unapplied patches:

```
hg qseries -v
```

Add changes to the current patch (the last one applied):

```
hg qrefresh
```

Apply the next patch from the queue:

```
hg qpush
```

Apply all patches:

```
hg qpush -a
```

Unapply current patch:

```
hg qpop
```

Unapply all patches:

```
hg qpop -a
```

Go to a given patch in the queue:

```
hg qgoto patch
```

Add a message to the current patch (before committing it):

```
hg qrefresh -m "Message"
```

Transform a patch into a commit:

```
hg qfinish patch
```

Commit changes made in the patch repository:

```
hg qcommit -m "Message de commit"
```

Note: Patches are saved into `.hg/patches`. The file `.hg/patches/series` can be manually edited to change the application order of the patches; but take care if several patches have the same target file: it can create problems.

7.6.4 MQ & Merge

General idea

Patches can be updated using the merge tool of **`Mercurial`**: it's easier than editing them manually. To do this, it's necessary to have two heads in the repository. One being the repository with patches applied ontop; the other the repository with the new commits/updates/etc:

```
o New repository revision
|
|
| o Patches
| |
| /
|
o Repository before patch application
```

The patches branch will next be merged into the new branch and **MQ** will use the merge function from **Mercurial** to update the patches. Please note that using an external merge tool (such as **meld** proposed previously) is highly recommended.

Create the head patches:

```
qpush -a
hg tags  # Remember/Note the number of the revision qparent
qsave -e -c # Save the status of the patches,
            # this save will be used during the merge.
            # (Remember/Note the number at the end of patches.N;
            # it's generally 1)
```

Create the new head:

```
hg update -C <N qparent> # Go to the revision noted before.

# Next, do what you planned to:
# Update:
hg pull -u
# Commit new changes, make the modifications then:
hg commit -m "message"
# Edit a patch:
hg qgoto patch # Then do the modifications and:
hg qrefresh
```

To launch the merge:

```
hg qpush -a -m
```

Clean the repository:

```
hg qpop -a
hg qpop -a -n patches.N
rm -r .hg/patches.N
```

Save the changes made into the patch repository:

```
hg qcommit -m "Updated to match rev???"
```

Re-apply the patch queue:

```
hg qpush -a
```

7.7 Release tasks

author pankso, linea, erjo, mojo

Things to do before publishing a new version.

Note: 20140502 <pankso> 5.0 is out RC. The current things to do are:

- Write the *5.0 SliTaz Release Notes (draft)* (page 385) in English here on the Wiki, then static xHTML for translation on the list and commits in **slitaz-doc**
- Update docs in this wiki. The Handbook must be hugely improved or removed since people use the guides section (ex liveUSB has 2 pages…)

7.7.1 Cooking & RC

1. Make sure all repos are tagged to include latest changes and are updated into the wok. Release source tarballs and push them to mirror with **tazdev**

```
# hg tag 4.4 && hg push
# tazdev relpkg slitaz-tools 4.4
```

2. Make sure all packages are built on Tank, generate a new package list with cook.

```
# cooker
# cook pkgdb --flavors
```

3. Sync the mirror with newly built packages.

```
# tazdev -p username
```

4. Use a chroot and **tazdev** (or **conspy -f 4** on Tank) to build ISOs
5. IMPORTANT: Use the packages on mirror.slitaz.org to build the ISOs:

```
# rm -rf /home/slitaz/packages && tazpkg -cc
```

6. If the flavors repo has been modified don’t forget to update it in the chroot
7. Build a core with **Tazlito**:

```
# tazlito pack-flavor core
# tazlito get-flavor core
# tazlito gen-distro
```

8. Boot this ISO with **Qemu**, burn it to a CD and boot, make a live USB and boot to make sure everything works.
9. When on the desktop, try ALL desktop entries and a few commands or latest code.
10. Connect to the web and install a few packages to test them
11. IMPORTANT: Install to HD and reboot, users should got an X environment without any other modifications. Installation and upgrade is really important for 4.0 since we have a new installer.

12. Prepare website and Distrowatch announcement and send it on the list.
13. Build all flavor ISOs and upload them to the mirror
14. Commit website news and update website — Post on SCN and Twitter
15. Open a new thread on the forum for feedback and share the URL

7.7.2 Stable

1. After some Cooking and a few RCs
2. Release version is specified by `/etc/slitaz-release`, this file is controlled by **`slitaz-base-files`**. Base files are tagged just before a stable release to change the version string and let **`Tazpkg`** use the new packages.
3. Check that the SliTaz version specified in `/etc/issue` message is the same as displayed in `isolinux.cfg` for **`Vesa`** menu, here the **`syslinux`** package must be modified.
4. When all other repos are tagged like for a *cooking*, update all packages, we can tag the main wok.
5. Remove the current stable wok and copy the cooking wok in place.
6. Sync Tank with mirror
7. Go on mirror and copy all cooking packages to the stable string.
8. Remove stable string in cooking wok and slitaz-base-file repo (update it again) so it goes back to a normal state and devs can start commiting. Toolchain updates may be discussed at this moment.
9. Build ISOs from a stable chroot or running system and use packages on the main mirror as usual.
10. Test, test, and test again…
11. Prepare the website announcement and RSS feeds. The Mailing list is used for translation and any text should be submitted 1 or 2 days before release.
12. Upload ISOs to mirror, commits news and update the website
13. Spread the word and go to sleep…

7.7.3 Stable documentation

SliTaz stable release provides the release notes on the LiveCD through the package **`slitaz-doc`**, the repos are tagged just before release and are archived on the mirror. After the wok has been copied to wok-stable, the docs are back to a cooking cycle and just provide an index with basic information. On the system, docs are located in `/usr/share/doc/slitaz`, a desktop file and icon are provided in the sources package and can be used to have quick access to the documentation.

- SliTaz Doc repo[260]

7.8 5.0 SliTaz Release Notes (draft)

author pankso, domcox, bellard, trixar_za, linea, godane, paranor, mojo

[260] http://hg.slitaz.org/slitaz-doc/

Note: Can the developers and coders also add to these notes as they are probably the only ones fully aware of all the changes since 4.0. 20140511 — Feel free to edit, modify and add new sections. Thanks

Welcome to SliTaz 5.0! Modern, secure, fast and flexible, the new SliTaz stable version is now out for Desktops and servers in production use.

- Over ??? commits in the repos
- 4400 packages in the packages database
- Slitaz ported to the ARM architecture
- Official Raspberry Pi flavors
- New Slitaz-configs dialog tool
- Frugal install tools added
- Many tools updated and improved (tazbox, tazweb, ???)

7.8.1 Overview

SliTaz GNU/Linux is a free, open source community project. Version 5.0 was released on ??? after two years of hard work. SliTaz comprises of 4400 software packages easily installed via the package manager “Tazpkg”. The LiveCD can be fully configured to taste, to easily create a custom distribution specifically for tasks such as multimedia, graphics or development.

SliTaz can also be installed to your hard drive, or used with USB media — with “TazUSB” you are only a few simple commands away from a fully formatted and configured USB device, ready to boot and the Tazlito tool allows you to fully remaster your own your Live CD. SliTaz provides also a Cookutils suite of tools to build your own packages.

Technical support is provided to users via the mailing list and the official forum. The “Slitaz Handbook” is an instructive manual on how to use and finely configure the system. SliTaz can be updated easily via the graphic installer or by using the simple fast text installer.

7.8.2 Hardware

As well as most i486 or x86 architectures, SliTaz also supports ARM v6 and work is on the stove for armv6hf (hard float) and armv7. A minimum of ??? is required to use the LiveCD and our Russians dolls system will automatically boot one of 4 systems, (currently core, gtk only, X windows and base) depending on the amount of memory available. The kernel supports a wide variety of wireless, network, ???, sound drivers and firmware.

7.8.3 Kernel & Toolchain

The kernel currently stands at 3.2.53 and uses Binutils 2.23.1, GCC 4.6.3, Glibc 2.14.1. Cookutils now boasts two new tools: ‘cross’ which gives you the ability to cross compile i?86, arm and x86_64 packages and ‘cooklinux’ which generates custom kernels from scratch.

7.8.4 LiveCD core flavors

SliTaz GNU/Linux is distributed as a bootable LiveCD allowing you to graphically install to the hard drive and retain the use of your previous system including all settings, applications, documents, etc. The core LiveCD contains 4 different operating systems and will automatically select the one that your hardware will support. You can also select images from the menu and hit `Tab` to edit any options. The menu also provides a command line, help options, languages and the ability to webboot. The ISO image uses a 'hybrid' system which can also be directly copied (**`dd`**) onto a USB stick without formatting. SliTaz also provides the tools to remaster your LiveCD both graphically and from a command line. The SliTaz community provides many custom and preset flavors which can be downloaded from our mirrors.

7.8.5 Installation

Installation can be fully automated with the Tazinst tool found in the SliTaz panel, this allows you to install SliTaz from a Live-CD, a LiveUSB key, a downloaded ISO image, or directly from the web. Frugal tools have been added to 5.0 to assist in a minimal HDD install and Tazinst also supports command line installation.

7.8.6 Raspberry Pi

The SliTaz port to the ARM processing architecture consists of a text mode base system and a lightweight desktop powered by JWM and the Fox toolkit. Both can be installed to a SD card by using the instructions found in the download directory. SliTaz RPi comes with its own tools including 'spk' — a tiny package manager, 'slitaz-config' — a system configuration menu and the 'tazberry' tool for RPi specific configurations. Boot scripts have been optimized for SliTaz and you will be prompted for a root password, keymap and default user on first power on. SliTaz RPi has nearly 900 packages in its database and can also keep up to date with system and kernel updates. The SliTaz Pi Book[261] contains detailed instructions to help you install and get started on ARM and the Raspberry Pi. Official SliTaz Raspberry Pi website[262]

7.8.7 Packages

Among the 4400 packages available in SliTaz 5.0 you will find anything you need to transform your machine to an office suite with LibreOffice, graphics studio with The Gimp or Inkscape, a video editor with Kino or a complete graphical desktop (razor-qt). You can experience the world wide web with instant messaging, VOIP, email and of course through a web browser. Packages can be found through the search function of Tazpkg, Tazpanel or via the website: http://pkgs.slitaz.org/

Iptables functions as the firewall and Rsync can be installed for incremental backup. SliTaz can of course also provide a complete development environment with the GCC 4.6.3 compiler, Geany IDE, Mercurial Repostitories and all development libraries.

SliTaz is designed to function as a powerful web server, using the stable LightTPD/PHP package (not installed by default), supporting CGI, Perl and Python. Apache, Squid and Privoxy are also available. The ability to browse the web securely using the Tor network is also supported. Packages are also checked by Cookutils to ensure the FHS is followed and packages are now built automatically by the SliTaz Build Bot: http://cook.slitaz.org/

[261] http://arm.slitaz.org/codex/pibook.html
[262] http://arm.slitaz.org/rpi/

7.8.8 Administration

The tool to configure the system on SliTaz 5.0 is TazPanel. It is a CGI/web interface with a themable user interface — which was updated for 5.0 — from where you can control your entire system such as networking, package management, adding or removing users, managing hardware, creating Live systems and much more. Each page provides a small description to help you manage your SliTaz system. To access the panel you can use the menu entry in “System Tools” or this url: http://tazpanel:82

7.8.9 Core Desktop

By default, the SliTaz LiveCD uses the very light and stable Openbox window manager. Openbox is widely themeable and configurable using the ObConf utility. The integration of the taskbar “LXpanel” makes it possible to dynamically provide a menu based on the Freedesktop standards. The principle is to have a small menu accessible via a screen click with the favorites, windows effects, LiveCD and LiveUSB tools, Openbox configuration and system actions made available. Applications can also be accessed through the menu supplied by LXpanel. The managment of the Desktop and icons are entrusted to the file manager PCManFM.

Through the support of a LiveCD flavor or an installed system you can install the Enlightenment (e17) desktop environment or the window managers XFCE, Pekwm, JWM, DWM and Razor-qt. The different sessions can be selected via the `F1` key when using the “Slim” login window. To change the default session you can use tazx; or manually edit the `~/.Xinitrc` file.

7.8.10 From 4.0 to 5.0

A SliTaz GNU/Linux installer offers an update function allowing you to upgrade from a 4.0 to 5.0 version. To upgrade the system you first need to boot the Stable LiveCD, launch the installer from the command line or Tazpanel, select upgrade and then specify the partition or configuration file containing the system that you want to update. The installer will then clean out the system and reinstall all the packages not present on the CD from the mirror. When this has finished you can reboot with your new version of SliTaz.

To upgrade a 4.0 to 5.0 it is also possible to use the package manager Tazpkg via the set-release function, but beware this is not yet proven and may require some manual intervention.

7.8.11 People of the Project

SliTaz is proud to be an international community project. The people of the project are the ones who develop the distribution, correct the website, develop the HG repositories and write the official documentation. Passing through Switzerland, France, Brazil, Quebec, China, India, Russia, Ukraine, England, and the U.S.

7.9 Boot scripts

author jozee, linea

The startup and shutdown scripts with their configuration files.

7.9.1 SliTaz and startup

SliTaz does not use a level of execution (runlevel), the system is initialized via a primary script and its main configuration file. This script itself launches some other smaller scripts which deal with the internationalization or any commands placed for the system to start.

7.9.2 `/etc/init.d/*` — Directory of scripts/daemons

The directory `/etc/init.d` contains all of the rc scripts, scripts finishing with '.sh' are simple shell scripts and daemons such as `dropbear` or `lighttpd` are scripts that launch a service. The daemon scripts can start, stop or restart through the command:

```
# /etc/init.d/daemon [start|stop|restart]
```

On SliTaz you will find a `/etc/init.d/README` describing the basic function of rc scripts. Also note that all startup scripts and daemons can call upon the `/etc/init.d/rc.functions` file. This file makes it possible to include various functions in rc scripts. For example, SliTaz uses a function `status` to check whether the previous command has succeeded (0) or not.

7.9.3 `/etc/init.d/rcS` — Primary initialization script

The `/etc/init.d/rcS` script configures all the basic services and initializes the base system. It begins by mounting the filesystems and starts services like `syslogd`, `klogd`, `mdev` and cleans up the system and so on. It uses the configuration file `/etc/rcS.conf` to locate which daemons and scripts to launch at startup. You can browse the script to learn which commands are executed:

```
# nano rootfs/etc/init.d/rcS
```

7.9.4 Specific scripts and daemons

`bootopts.sh` — LiveCD mode options

This script is used to configure the LiveCD options passed at boot time and is readable via the `/proc/cmdline` file. This is the script that enables you to use a USB key or external hard disk `/home` partition with the option `home=usb` or `home=sda[1-9]`. Note that it can also directly specify the language and keyboard parameters.

network.sh — Initializing the network

This script searches the network.sh configuration file /etc/network.conf for the network interface to use; if one wants to launch the DHCP client (or not) or if you want to use a fixed (static) IP. On SliTaz the /etc/init.d/network.sh script configures the network interfaces to start using the information contained in /etc/network.conf. If the variable $DHCP is equal to yes, then the /etc/init.d/network.sh script launches the DHCP client on the $INTERFACE interface.

i18n.sh — Internationalization

SliTaz backs up the configuration of the default locale in /etc/locale.conf which is read by /etc/profile at each login. The /etc/locale.conf is generated during boot time thanks to the /etc/i18n.sh script. This script launches the **tazlocale** application if /etc/locale.conf doesn't exist. We use the same process for the keyboard layout using **tazkmap** and the /etc/kmap.conf configuration file. Both applications are installed and located in /sbin and use **dialog** and the **ncurses** library. The script also checks whether the configuration file for the time zone /etc/TZ exists, otherwise it creates one relying on the keyboard configuration.

local.sh — Local commands

The /etc/init.d/local.sh script allows the system administrator to add local commands to be executed at boot. Example:

```
#!/bin/sh
# /etc/init.d/local.sh: Local startup commands.
# All commands here will be executed at boot time.
#
. /etc/init.d/rc.functions

echo "Starting local startup commands... "
```

wpa_action.sh — Wireless network

This script is applied by network.sh to start/restart the DHCP server if you use a dynamic IP.

rc.shutdown

This script is invoked by /etc/inittab during system shutdown. It also stops all daemons via the variable RUN_DAEMONS in the primary /etc/rcS.conf configuration file.

7.9.5 /etc/inittab — Configuration file init

The first file read by the Kernel at boot. It defines the initialization script (/etc/init.d/rcS), virtual terminals (ttys) and actions in the event of a reboot or disruption. You will find a complete example with accompanying notes in *SliTaz Tools* (page 380):

```
# /etc/inittab: init configuration for SliTaz GNU/Linux.
# Boot-time system configuration/initialization script.
#
::sysinit:/etc/init.d/rcS

# /sbin/getty respawn shell invocations for selected ttys.
tty1::respawn:/sbin/getty 38400 tty1
tty2::respawn:/sbin/getty 38400 tty2
tty3::respawn:/sbin/getty 38400 tty3
tty4::respawn:/sbin/getty 38400 tty4
tty5::respawn:/sbin/getty 38400 tty5
tty6::respawn:/sbin/getty 38400 tty6

# Stuff to do when restarting the init
# process, or before rebooting.
::restart:/etc/init.d/rc.shutdown
::restart:/sbin/init
::ctrlaltdel:/sbin/reboot
::shutdown:/etc/init.d/rc.shutdown
```

7.10 Rootcd

author jozee, linea

Descriptions of files contained on the CD-ROM.

7.10.1 Syslinux/isolinux

`Syslinux` and the (SliTaz) main bootloader — we use the **`isolinux`** version to start the system contained on the CD-ROM. Simple effective and configurable, **`isolinux`** was installed during the creation of the base system. The binary is named `isolinux.bin` and uses the configuration file `isolinux.cfg`. Here's an example of an `isolinux.cfg` using `isolinux.msg` to post the splash image and displayable help files via `F1`, `F2`, `F3` and `F4`. You will find the files `help.txt`, `options.txt`, etc in *SliTaz Tools* (page 380).

```
display isolinux.msg
default slitaz
label slitaz
      kernel /boot/bzImage
      append initrd=/boot/rootfs.gz rw root=/dev/null vga=788
implicit 0
prompt 1
timeout 80
F1 help.txt
F2 options.txt
F3 isolinux.msg
F4 display.txt
```

7.10.2 Isolinux boot splash image

We can configure **`isolinux`** to display a splash image when booting SliTaz or any other operating system. This image has a particular format `.lss`, suitable for **`Syslinux`** and must be indexed using the 16 color mode. You can use the official logo, **ppmforge**, **imagemagick**, **GIMP** or other tools to create your image.

The **Syslinux** file (`sample/syslogo.lss`) provides an official logo which you can directly use by copying to the root of the CD-ROM. SliTaz provides a logo (`rootcd/boot/isolinux/splash.lss`) which you can find in *SliTaz Tools* (page 380). To display a splash image when booting, it's necessary that the 'display' option calls the `isolinux.msg` file which loads the `*.lss` format image. Note that the `isolinux.msg` file uses 24 ASCII characters. Example using **echo** and an `isolinux.msg` file incorporating a `.lss` splash image:

```
# echo -e "\24isplash.lss\n" > isolinux.msg
```

You can also add a text message underneath the splash image by modifying the file with your favorite text editor, **echo** or **cat** and so on.

7.10.3 ISO bootable with isolinux

To create a bootable ISO image using **isolinux** and **genisoimage**:

```
# genisoimage -R -o slitaz-test.iso -b boot/isolinux/isolinux.bin \
     -c boot/isolinux/boot.cat -no-emul-boot -boot-load-size 4 \
     -V "SliTaz" -input-charset iso8859-1 -boot-info-table rootcd
```

7.10.4 GRUB

GRUB (GRand Unified Bootloader) is a bootloader distributed by the GNU project. This is used during installation to a hard drive; it can boot Linux, BSD, HURD and Window$. GRUB provides `stage2_eltorito` to start the ISO images. To find `stage2_eltorito` on your system, you need to have the GRUB package installed. Finally you copy `stage2_eltorito` to the root of the cdrom. Note that SliTaz provides a (`.tazpkg`) package **grub-0.97** that you can find on the mirrors or you can rebuild **grub-0.97** from sources. Example using a `stage2_eltorito` image from a Debian system or SliTaz:

```
# mkdir -p rootcd/boot/grub
# cp /usr/lib/grub/i386-pc/stage2_eltorito \
     rootcd/boot/grub
```

The GRUB configuration file is called `menu.lst` and can be edited with your favorite text editor. Example:

```
# By default, boot the first entry.
default 0

# Boot automatically after 20 secs.
timeout 20

# Change the colors.
color yellow/brown white/black

title  SliTaz GNU/Linux 1.0 (vga 800x600) (Kernel 2.6.20)
       kernel /boot/bzImage root=/dev/null vga=788
       initrd /boot/rootfs.gz
```

(continues on next page)

(continued from previous page)

```
title  SliTaz GNU/Linux 1.0 (vga 1024x768) (Kernel 2.6.20)
       kernel /boot/bzImage root=/dev/null vga=771
       initrd /boot/rootfs.gz
```

7.10.5 ISO bootable with GRUB

To create a bootable ISO image using **GRUB** and **genisoimage** or **mkisofs**:

```
# genisoimage -R -o slitaz-test.iso -b boot/grub/stage2_eltorito \
     -no-emul-boot -V "SliTaz" -boot-load-size 4 -input-charset iso8859-1 \
     -boot-info-table rootcd
```

7.10.6 Memtest86

The application **memtest86** is a tool to test random access memory (RAM). We download the utility into the src directory, decompress the archive, and copy the (precompiled) binary:

```
# mkdir -v -p src
# cd src
# wget http://www.memtest86.com/memtest86-3.2.tar.gz
# tar xzfv memtest86-3.2.tar.gz
# cd memtest86-3.2
(# more README)
# cp precomp.bin ../../rootcd/boot/memtest
# cd ../..
```

Once installed, you can add the label for the `memtest86` file to `isolinux.cfg`, specifing the path to the utility:

```
label memtest
      kernel /boot/memtest
```

Or if you want to use **GRUB**, here's the line to launch **memtest86**:

```
title          Memtest86 (Test system memory)
kernel         /boot/memtest
```

Once the lines are added, you can then create a new ISO and test.

7.11 Toolchain

author pankso, linea, brianperry

The Toolchain is the set of packages used to build the entire system. It consists of **Binutils**, **GCC**, **Linux API headers** and the **GNU libc** aka **Glibc**. SliTaz has an annual development cycle, so the toolchain gets a huge update once a year just after a stable release. When the toolchain changes we must rebuild all packages to ensure quality and consistency.

On SliTaz you have 2 ways to rebuild a toolchain from scratch: **Cookutils** and **Tazwok**. **Tazwok** is historically the first packages builder for SliTaz and over time has evolved into a powerful tool that can rebuild the full system from scratch. More info: *The new tazwok illustrated!* (page 258). (Tazwok has been deprecated since SliTaz v4).

Cookutils are the new tools, written from scratch as before and for SliTaz 4.0. The build tools needed a huge improvement since they were first written when SliTaz had 400-500 packages, when **Cookutils** was started SliTaz had 2960 packages, so the way to handle all these packages is even more complex than at the beginning of the project. Also the **Cookutils** have been written with simplicity and speed in mind and are used on the official SliTaz build host aka Tank.

To rebuild the official toolchain with **Cookutils** you must use the package **slitaz-toolchain**. This package is used to install the toolchain in a development environment and builds the **Toolchain** in the correct order with 2 passes for **binutils** and **GCC**. You will find some info in the **slitaz-toolchain** receipt itself and after building we keep a note in `/usr/share/doc/slitaz/toolchain.txt`

Recooking the full toolchain can be slow:

```
# cook slitaz-toolchain
```

7.12 Cross compilation

author pankso, linea

After SliTaz got new build tools for 4.0, we had to rebuild all packages and used this opportunity to cross compile as much as possible in the wok. There are many reasons: lots of user requests about Slitaz on ARM, prepare for the future and cross compile by default on our main build host aka Tank. Tank is a i686 machine and SliTaz targets i486 which is a more generic architecture and works on almost all X86 systems, so we enable cross compilation from a i686 build system for a i486 host system. This way we take advantage of Tank's power while we produce i486 binaries. If you look at the packages build log on Tank you will see:

```
checking build system type... i686-slitaz-linux-gnu
checking host system type... i486-slitaz-linux-gnu
checking whether we are cross compiling... yes
```

--build, --host and --target

Build and host. These options are for cross-compiling. If you specify both options and `BUILD_SYSTEM` is different from `HOST_SYSTEM`, **configure** will prepare to cross-compile from `BUILD_SYSTEM` to be used on `HOST_SYSTEM`.

To make it work we have 2 variables in `cook.conf`: `BUILD_SYSTEM` and `HOST_SYSTEM`. The *build system* is auto-detected by **uname -m** but can be changed to the native i486-slitaz-linux machine type tools. The *host system* is set with the `ARCH` variable also used in `CFLAGS`. In `CFLAGS` we have the **GCC** compiler options and by default we use `-march=$ARCH` but it must be changed for some architectures as explained below.

The `--target` option is only used for building cross-compilers.

Current state and packages

While all main packages cross compile well, we still have some receipts to improve. But it may take some time, so for now we use a native i486 build for packages that don't compile. Instead of `$CONFIGURE_ARGS` we use: `--build=$HOST_SYSTEM --host=$HOST_SYSTEM` so it doesn't cross compile but builds with the native i486 toolchain, this doesn't screw anything since it is the default architecture, but we can't cross compile these packages for another arch like ARM.

Cross compiler and toolchain

If we want to be able to cook packages from a X86 machine for another architecture like ARM we need a cross compilation toolchain. A basic cross toolchain is: **`binutils`** and **`gcc`**. We need to make packages for cross compilers and use the `--target` option in **`configure`**. Accordingly the two example/test packages: `cross-arm-binutils` and `cross-arm-gcc` are a first attempt to provide a basic C compiler. After installation you can check the machine name with:

```
$ arm-slitaz-linux-gcc -dumpmachine
```

Note: Actually the packages were committed to let others try them and make them work properly since we are still in a building stage. Also if you do change the receipt keep the `CFLAGS` unset since this should not be set when building a cross compiler.

ARM Note

Listing 1: cook.conf

```
ARCH="arm"
CFLAGS="-O2"
```

X86_64 Notes

Notes about building natively on X86_64 and cross compiling for X86_64 machine.

Listing 2: cook.conf

```
ARCH="X86_64"
CFLAGS="-O2 -march=generic"
```

7.13 Contributors

author bellard

(Work in progress…)

7.13.1 How to contribute

- Documentation
- Translation
- Web design
- Art work
- Improve usability / simplicity
- Package set selection
- Package development

7.13.2 SliTaz network

- Restricted web acces

```
$ htpasswd -nb YourUsername YourPassword
```

- unix accounts

```
# grep ^YourUsername /etc/shadow
```

CHAPTER 8

Scratchbook

author jozee, linea, domcox

Index of documents, step by step construction of a mini GNU/LINUX LiveCD and installation instructions.

8.1 Introduction

author jozee, linea, domcox

The scratchbook allows you to track the creation of the first public release of SliTaz and make a trip to the heart of GNU/Linux. You'll be able to customize your new system or create your own autonomous distro running in system memory (RAM) that's fully installable on a hard drive or USB key. Once started you'll be able to remove the CD-ROM and still have SliTaz working. SliTaz can also be used as an environment in which we can chroot or use the CD-ROM for multitasking. The only prerequisite is a host distribution in which you can store libraries, use a compiler and development tools, etc. The host system can be a chrooted development environment, a minimal distro, SliTaz installed on a hard drive or a 'general' distro such as Debian, Slackware, Fedora, Gentoo, Mandriva, Arch, etc. Note that nothing is installed in the host system by our commands.

SliTaz uses the 'Swiss Army Knife' BusyBox as the basis of the system and the Linux Kernel, it runs embedded using a small memory footprint and provides many files. BusyBox is our main source of information and it's a utility of the Debian project which we use and cherish.

SliTaz uses the Syslinux bootloader and an archived initramfs compressed with cpio. This archive is then decompressed in memory at boot by the kernel into a system of no fixed size, retaining control over init. At the time of compilation or copying of applications, we use strip to clean the repositoiries. The system commands **`genisoimage`** or **`mkisofs`** are used to create the ISO images. To finish, you can test the ISO image with **`Qemu`** or engrave the generated ISO on to a rewritable CD-RW.

8.2 Organize a working directory

To create SliTaz, we need a working directory and several subdirectories whether you have a chrooted environment for developing or a host system, we advise to use a directory named `distro/` in which to work. The `distro/` directory can be a simple folder or a partition, but you are obviously free to put all of this elsewhere.

`distro/`

Contents of a working directory:

`rootfs/`: The root filesystem — this is the root system, designed to operate in RAM, it is used to generate the initramfs image.

`rootfs.gz`: The initramfs image of our system — a cpio archive compressed with gzip.

`rootcd/`: The rootcd. This is the root of the CD-ROM files.

`src/`: The sources, Kernel, Syslinux, Busybox, Dropbear, etc (it can also be a symbolic link).

Thereafter, the initramfs and bootable ISO image (`slitaz-cooking.iso`) will be created in the root directory of our work named `SliTaz/`.

Option: rootfs.ext2 — using a virtual hard drive

Option: rootfs.ext2 (root filesystem in ext2) is a virtual hard disk formatted with ext2 and mounted on a (rootfs) loop. A device loop allows a file to be used as a standard device (hard drive, floppy, etc) to build a filesystem inside. This file can be any number of megabytes, we propose 20,480, which corresponds to 20MB:

```
# dd if=/dev/zero of=rootfs.ext2 bs=1k count=20480
```

Create a *ext2* filesystem named `rootfs.ext2`, the option `-F` formats the file. Note that the `-m 0` option doesn't allocate any space for the user root — by default it occupies approximately 5% and the `-t` option defines the type of filesystem to be used, such as ext2 or ext3:

```
# mkfs -t ext2 -F -m 0 rootfs.ext2
```

We can now assemble `rootfs.ext2` with a loop, thanks to the `-o loop option` provided by the mount utility in the `rootfs/` directory. You can check if the assembly went well with the **`df-h`** command:

```
# mkdir rootfs
# mount -o loop rootfs.ext2 rootfs
# df -h
```

At the end of the session, you can dismount the volume with umount:

```
# umount rootfs
```

Now we can proceed to the construction of the *Base System* (page 399).

8.3 Base System

author domcox, lebardix

Build a SliTaz GNU/Linux distro running in RAM and using BusyBox.

8.3.1 About

This document describes the construction of the SliTaz base system and why we use a Linux Kernel, BusyBox and Syslinux to boot the system. SliTaz uses an initramfs archive unpacked in RAM by the kernel at boot. We will create a box to hold a root of 3 to 4MB and use strip on the libraries and binaries to save space.

The scripts and configuration files are created with GNU nano, using the keystroke `Ctrl+X` to save and exit. But of course you are free to replace with your own text editor.

This document is based on a howto found in the archive of BusyBox, which is itself based on a paper presented by Erik Anderson in the Embedded Systems Conference in 2001.

8.3.2 Wget src

Create a `src` directory for downloading and compiling:

```
# mkdir -p src
# cd src
```

- Linux Kernel 2.6.20 (http://www.kernel.org/).

  ```
  # wget ftp://ftp.kernel.org/pub/linux/kernel/v2.6/linux-2.6.20.tar.bz2
  ```

- Busybox 1.2.2 (http://www.busybox.net/).

  ```
  # wget http://www.busybox.net/downloads/busybox-1.2.2.tar.bz2
  ```

- Syslinux 3.35 (http://syslinux.zytor.com/).

  ```
  # wget ftp://ftp.kernel.org/pub/linux/boot/syslinux/3.xx/syslinux-3.
  ↪35.tar.gz
  ```

- SliTaz tools 1.1. Download SliTaz tools, unpack, save the file in `src/` and that's it:

  ```
  # wget http://download.tuxfamily.org/slitaz/sources/tools/slitaz-
  ↪tools-1.1.tar.gz
  # tar xzf slitaz-tools-1.1.tar.gz
  ```

8.3.3 Unpack and prepare the Linux Kernel

We will begin by compiling a Linux kernel, which may take a little time.

8.3.3.1 Linux Kernel

Your kernel must support the intramfs filesystem, otherwise the CD-ROM will not start. You can also install the modules in a directory so as not to touch the host system. The configuration of the Linux kernel sources is done by **`make menuconfig`** using ncurses or graphically with **`make gconfig`** or **`make xconfig`** using GTK development packages and/or QT respectively. You can find in *SliTaz Tools* (page 380), Makefiles for the various 2.6.xx kernels.

A feature of the 2.6 kernels is that if we **`make menuconfig`**, **`xconfig`** or **`config`** for the first time, the setup menu is displayed based on the configuration of our current kernel.

The options depend on your needs, you can install module-init-tools to support compressed modules or for a minimal install, you can select only the vital options.

We start by changing into the sources, **`make mrproper`** to put things in order, then we start a configuration interface: gconfig, xconfig, menuconfig or oldconfig:

```
# tar xjf linux-2.6.20.tar.bz2
# cd linux-2.6.20
# make mrproper
# cp ../slitaz-tools-1.1/Makefiles/linux-2.6.20-slitaz.config .config
# make oldconfig
(# make menuconfig)
# make bzImage
# make modules
# make INSTALL_MOD_PATH=$PWD/_pkg modules_install
# cd ..
```

If you want more info on compiling kernels, there are many textbooks. Note that you can install the kernel and after rebooting, you can compile your own kernel following the same instructions.

8.3.4 Creation of the root system (rootfs)

The next step will create a file named 'rootfs' — Root File System, in the working directory `SliTaz/`:

```
# mkdir ../rootfs
```

8.3.4.1 Install BusyBox

BusyBox (www.busybox.net[263]) is a single executable offering versions of the main tools necessary to use a Linux kernel. It is (mainly) intended to be used embedded and can do almost anything. As well as proposing (coreutils) shell commands and a daemons system, it also provides a websever and client/server (DHCP, udhcpc).

```
# tar xjf busybox-1.2.2.tar.bz2
```

Configure and compile, remembering the dumpkmap options, init, etc — you can find help in the Makefile in SliTaz Busybox tools. Make install creates a `_install` directory in the current directory:

```
# cd busybox-1.2.2
# cp ../slitaz-tools-1.1/Makefiles/busybox-1.2.2.config .config
```

(continues on next page)

[263] http://www.busybox.net/

(continued from previous page)

```
# make oldconfig
(# make menuconfig)
# make
# make install
# chmod 4755 _install/bin/busybox
```

Copy files compiled by BusyBox in the directory `_install` to the root file system (rootfs):

```
# cp -a _install/* ../../rootfs
```

The `linuxrc` link pointing to `/bin/busybox`, folders `/bin`, `/lib` and `/sbin` were added to the directory `/rootfs` — you can check this. It may be that the link isn't there if you didn't select the option initrd support in Busybox. We'll delete the `linuxrc` link and create a link for `init` that points to `/bin/busybox`:

```
# cd ../../rootfs
# ls -CF
bin/  linuxrc@  sbin/  usr/

# rm linuxrc
# ln -s bin/busybox init
```

8.3.4.2 `ldd` on `BusyBox`

The **`ldd`** command can show any libraries used by a program. Libraries used by **`Busybox`** may differ depending on the host system. On Debian for example, copying the libraries in `/lib/tls`. The following commands are given using 'v' for verbose mode. To eliminate the symbols of executable binaries and shared libraries we can utilize **`strip`**. Note you may also use the mklibs or uClibc libraries.

```
# mkdir lib
```

SliTaz or another:

```
# cp /lib/{libcrypt.so.1,libm.so.6,libc.so.6} lib
# cp /lib/ld-linux.so.2 lib
```

Example on Debian Etch:

```
# cp /lib/tls/{libcrypt.so.1,libm.so.6,libc.so.6} lib
# cp /lib/ld-linux.so.2 lib
```

Cleanup libraries with **`strip`**:

```
# strip -v lib/*
```

8.3.4.3 Linux tree and configuration

Make some directories for a classic Linux branch SliTaz installation. `/dev` for devices, `/etc`, `/home`, `/usr`, `/proc`, `/root` and co. To learn more about the hierarchy of a file system and its contents, there is a File System Hierarchy Standard available in various formats at www.pathname.com/fhs/[264].

[264] http://www.pathname.com/fhs/

You are free to create your own directory tree. In traditional Unix systems, /usr usually contains files from the distribution, /dev contains devices (devices), /etc contains configuration files, /lib libraries, /home for home users and /var for variable data. Note that we do not create /lib, /bin or /sbin — these are created when BusyBox is installed.

```
# mkdir -p dev etc root home proc media mnt sys tmp var
# mkdir -p usr/{lib,local,games,share} \
  var/{cache,lib,lock,log,games,run,spool} \
  media/{cdrom,flash,usbdisk}
```

Change permissions on the /tmp directory:

```
# chmod 1777 tmp
```

Setting up glibc — note /etc/ld.so.conf and /etc/rpc are not essential for a micro system:

```
# touch etc/ld.so.conf
# cp /etc/rpc etc
```

8.3.4.4 Create the devices in /dev

This can be done with the script mkdevs.sh found in BusyBox, or with our script mktazdevs.sh in SliTaz tools. If you want more details, read the scripts. If you used the BusyBox version, we must still create the pts directory:

```
# cp ../src/slitaz-tools-1.1/utils/mktazdevs.sh bin
# ./bin/mktazdevs.sh dev
```

or:

```
# cp ../src/busybox-1.2.2/examples/bootfloppy/mkdevs.sh bin
# ./bin/mkdevs.sh dev
# mkdir -p dev/{pts,input,shm,net,usb}
```

Note that we start mdev-s with the rcS script to create devices dynamically at boot.

8.3.4.5 Support for the resolution of hostnames (DNS)

Copy the libraries libnss_* of the host system into our SliTaz system. These libraries are used for name resolution and are cleaned with strip:

```
# cp /lib/{libnss_dns.so.2,libnss_files.so.2} lib
# cp /lib/libresolv.so.2 lib
# strip -v lib/*.so*
```

8.3.5 Configuration of your box

Create the necessary files in /etc. For more info, just look at the contents of the files. We start by creating some files relevant to the core operating system.

8.3.5.1 Configure network

Create basic files used to configure the network:

```
# echo "127.0.0.1      localhost" > etc/hosts
# echo "localnet    127.0.0.1" > etc/networks
# echo "slitaz" > etc/hostname
# echo "order hosts,bind" > etc/host.conf
# echo "multi on" >> etc/host.conf
```

8.3.5.2 `/etc/nsswitch.conf`

Configuration files used to resolve names:

```
# nano etc/nsswitch.conf
```

```
# /etc/nsswitch.conf: GNU Name Service Switch config.
#

passwd:     files
group:      files
shadow:     files

hosts:      files dns
networks:   files
```

8.3.5.3 `/etc/securetty`

`/etc/securetty` lists terminals that can connect to root:

```
# nano etc/securetty
```

```
# /etc/securetty: List of terminals on which root is allowed to login.
#
console

# For people with serial port consoles
ttyS0

# Standard consoles
tty1
tty2
tty3
tty4
tty5
tty6
tty7
```

8.3.5.4 `/etc/shells`

`/etc/shells`, a shells list of valid connections. This file is used by the SSH server (**`Dropbear`**):

```
# nano etc/shells
```

```
# /etc/shells: valid login shells.
/bin/sh
/bin/ash
/bin/hush
```

8.3.5.5 `/etc/issue` and `/etc/motd`

`/etc/issue` is displayed at the end of boot and the *message of the day* is displayed after logging in:

```
# echo "SliTaz GNU/Linux 1.0 Kernel \r \l" > etc/issue
# echo "" >> etc/issue
# nano etc/motd
```

```
 (°-  { Get documentation in: /usr/share/doc.
 //\    Use: 'less' or 'more' to read files, 'su' to be root. }
 v_/_

SliTaz is distributed in the hope that it will be useful, but
with ABSOLUTELY NO WARRANTY.
```

8.3.5.6 `/etc/busybox.conf`

The configuration file for BusyBox, it can set duties on BusyBox applications. For more information, you can read the *SliTaz and System Security* (page 344) page in the Handbook. `BusyBox.conf` file:

```
# nano etc/busybox.conf
```

```
# /etc/busybox.conf: SliTaz GNU/linux Busybox configuration.
#

[SUID]
# Allow command to be run by anyone.
su = ssx root.root
passwd = ssx root.root
loadkmap = ssx root.root
mount = ssx root.root
reboot = ssx root.root
halt = ssx root.root
```

For added security, change the permissions on the file:

```
# chmod 600 etc/busybox.conf
```

8.3.5.7 `/etc/inittab`

Minimal configuration file for init. It helps to have a root console without having to go through the login and a console on tty2.

```
# nano etc/inittab
```

```
# /etc/inittab: init configuration for SliTaz GNU/Linux.

::sysinit:/etc/init.d/rcS
::respawn:-/bin/sh
tty2::askfirst:-/bin/sh
::ctrlaltdel:/bin/umount -a -r
::ctrlaltdel:/sbin/reboot
```

You will also find a wider example of an `inittab` file in the archive of BusyBox.

8.3.5.8 `/etc/profile`

This file is read at each login and affects all users. We must use the `~/.profile` config file for each individual user:

```
# nano etc/profile
```

```
# /etc/profile: system-wide .profile file for the Bourne shells

PATH="/usr/sbin:/usr/bin:/sbin:/bin:/usr/games"
LD_LIBRARY_PATH="/usr/lib:/lib"

if [ "`id -u`" -eq 0 ]; then
  PS1='\e[1m\u@\h:\w\#\e[m '
else
  PS1='\e[1m\u@\h:\w\$\e[m '
fi

DISPLAY=:0.0

export PATH LD_LIBRARY_PATH PS1 DISPLAY ignoreeof
umask 022
```

8.3.5.9 Users, groups and passwords

Create configuration files of users, groups and passwords in `/etc/` *passwd*, *shadow*, *group*, *gshadow*, and adjust permissions:

```
# echo "root:x:0:0:root:/root:/bin/sh" > etc/passwd
# echo "root::13525:0:99999:7:::" > etc/shadow
# echo "root:x:0:" > etc/group
# echo "root:*::" > etc/gshadow
# chmod 640 etc/shadow
# chmod 640 etc/gshadow
```

You can add other users, like hacker is used by the LiveCD mode. You can also configure a password for the root user with the **passwd** command. To add an existing user to an existing group, you must edit `/etc/group` and `/etc/gshadow` because the applet **adduser** provided by BusyBox doesn't offer all of the options provided by the original program.

8.3.5.10 `/etc/fstab` or `/etc/mtab`

List filesystems to be mounted:

```
# nano etc/fstab
```

```
# /etc/fstab: information about static file system.
#
proc            /proc        proc     defaults          0        0
sysfs           /sys         sysfs    defaults          0        0
devpts          /dev/pts     devpts   defaults          0        0
tmpfs           /dev/shm     tmpfs    defaults          0        0
```

`/etc/mtab` is used by other `mkfs*`, for listing the mounted partitions. It needs `/proc` because there is a link on `/proc/mounts`:

```
# chroot . /bin/ash
/# ln -s /proc/mounts /etc/mtab
```

8.3.5.11 Keyboard

You can create a `kmap` file specific to your keyboard with the **`dumpkmap`** command provided by BusyBox. You can find some `kmap` files in SliTaz tools. To create a fr_CH kmap file:

```
/# mkdir /usr/share/kmap
/# /bin/busybox dumpkmap > /usr/share/kmap/fr_CH.kmap
/# exit
```

Once this is done, you can automatically load your keyboard with loadkmap in a `/etc/init.d/rcS` script:

8.3.5.12 `/usr/share/doc`

You can also add various documents, such as a SliTaz user manual, which you can download as a `tar.gz` from the website:

```
# mkdir -p usr/share/doc
```

8.3.5.13 Installing the `udhcpc` script

Udhcpc DHCP client supplied by Busybox is fast and stable, but is developed independently. Web site: http://udhcp.busybox.net/. You can use the default script found in the archive of BusyBox. This script goes into `/usr/share/udhcpc/default.script`, but this can be changed via the command line. On SliTaz, the client is started at boot by the script `/etc/init.d/network.sh` via the configuration file `/etc/network.conf`:

```
# mkdir usr/share/udhcpc
# cp ../src/busybox-1.2.2/examples/udhcp/simple.script \
  usr/share/udhcpc/default.script
# chmod +x usr/share/udhcpc/default.script
```

8.3.5.14 /etc/init.d/rcS

To finish off this draft, you must create the init script `/etc/init.d/rcS` to mount the filesystems and run some commands. For more information, you can look at the *Boot scripts* (page 388) page. You can change the value of the variable `KMAP=` for the keyboard:

```
# mkdir etc/init.d
# nano etc/init.d/rcS
```

```
#! /bin/sh
# /etc/init.d/rcS: rcS initial script.
#

KMAP=fr_CH

echo "Processing /etc/init.d/rcS... "

/bin/mount proc
/bin/mount -a
/bin/hostname -F /etc/hostname
/sbin/ifconfig lo 127.0.0.1 up
/sbin/loadkmap < /usr/share/kmap/$KMAP.kmap
```

```
# chmod +x etc/init.d/rcS
```

8.3.5.15 Note

Note that you can still install the Tazpkg package manager (10 kb) that we created, you will find information to install in the source tarball. You can also install various files from SliTaz tools, such as the licence.

8.3.6 Build an initramfs cpio archive

The initramfs is a "cpio" archive generated from the root of the system, it is decompressed in RAM by the Linux kernel at boot to create the filesystem (also in RAM). To generate an initramfs archive, using the root directory of system files (rootfs), we facilitate a search with **`find`** and add some pipes **`|`**. Then we create a cpio archive using **`gzip`** which we put in the working directory.

The SliTaz initramfs `rootfs.gz` is the root system, but with a `.gz` extension. If you want to change the name, you need to edit the configuration file for isolinux: `isolinux.cfg` or the `menu.lst` for GRUB.

Generation of the initramfs:

```
# find . -print | cpio -o -H newc | gzip -9 > ../rootfs.gz
```

You should now have a file `rootfs.gz` about 1 to 2MB in the working directory `SliTaz/`.

For a new image, when making changes in rootfs, simply copy the new `rootfs.gz` archive to `rootcd/boot` and create a new image with **`genisoimage`** or **`mkisofs`**. For this you can also use **`mktaziso`** within SliTaz tools. This script will check if the directories are present, create a new compressed cpio archive and generate a new bootable ISO image.

8.3.7 Make rootcd files

The following steps will help you create the root of the bootable CD-ROM. We begin by creating the `rootcd`, `boot` and `isolinux` directories for the CD-ROM files:

```
# cd ..
# mkdir -p rootcd/boot/isolinux
```

Optionally, you can create some other directories in which to place various data, such as HTML documents or packages.

8.3.7.1 Copy the kernel

Just copy the kernel previously compiled to `rootcd/boot`:

```
# cp src/linux-2.6.20/arch/i386/boot/bzImage rootcd/boot
```

8.3.7.2 Copy the initramfs into `rootcd/boot`

Copy the `rootfs.gz` to `rootcd/boot`. We must not forget to generate a new initramfs archive for any changes made to the rootfs (root file system):

```
# cp rootfs.gz rootcd/boot
```

8.3.7.3 Install the isolinux bootloader

The bootloader isolinux — simply copy the `isolinux.bin` from the source archive of Syslinux:

```
# cd src
# tar xzf syslinux-3.35.tar.gz
# cp syslinux-3.35/isolinux.bin ../rootcd/boot/isolinux
# cd ..
```

8.3.7.4 `isolinux.cfg` — Configure isolinux

Here is an example of an `isolinux.cfg` file that should work well. You can change it if you wish:

```
# nano rootcd/boot/isolinux/isolinux.cfg
```

```
display display.txt
default slitaz
label slitaz
    kernel /boot/bzImage
    append initrd=/boot/rootfs.gz rw root=/dev/null vga=788
implicit 0
prompt 1
timeout 80
```

Here are some changes that you might like to make in `isolinux.cfg`:

- The `timeout` value is the number of seconds to wait before booting. You can make it 0 or delete the line to start instantly, or choose to wait as long as 10s.
- `prompt` can be set to 0 to disable the `boot:` prompt.
- You can add more lines to view the contents of several text files when the user presses `F1`, `F2`, `F3`, etc.

8.3.7.5 `display.txt`

A small welcome note, powered by isolinux, you can modify this file if you wish:

```
# nano rootcd/boot/isolinux/display.txt
```

```
/*       _\|/_
         (o o)
 +----oOO-{_}-OOo---------------------------------------------------+

     ____  _ _ _____
    / ___|| (_)_   _|_ _ ____
    \___ \| | | | |/ _` |_  /
     ___) | | | | | (_| |/ /
    |____/|_|_| |_|\__,_/___|

 SliTaz GNU/Linux - Temporary Autonomous Zone

     <ENTER> to boot.

                                                                   */
```

8.3.8 Create an ISO image with `genisoimage` or `mkisofs`

```
# genisoimage -R -o slitaz-cooking.iso -b boot/isolinux/isolinux.bin \
  -c boot/isolinux/boot.cat -no-emul-boot -boot-load-size 4 \
  -V "SliTaz" -input-charset iso8859-1 -boot-info-table rootcd
```

For each change in the root of the box, you must create a new ISO image.

You can create a small script that will generate a new compressed cpio archive and a new image, or use **`mktaziso`** within SliTaz tools. Note that you can also use GRUB to boot the box.

8.3.9 Burn or test ISO image with `Qemu`

You can burn the ISO image with **`Graveman`**, **`k3b`** or **`wodim`** and boot it. Simple burning command using **`wodim`** (also valid for **`cdrecord`**), with a 2.6.XX. kernel:

```
# wodim -v -speed=24 -data slitaz-cooking.iso
```

8.3.9.1 Qemu

Note that you can test the ISO image with the software emulator **`Qemu`** (On Debian **`# aptitude install qemu`**). To emulate the newly created ISO image, simply type:

```
# qemu -cdrom slitaz-cooking.iso
```

8.3.9.2 Following chapter

The next chapter *Base Applications* (page 410) provides all the instructions to install and configure the basic applications and libraries.

8.4 Base Applications

author domcox

Install and configure libraries and basic applications.

8.4.1 About

This chapter describes the facilities libraries and basic text mode applications supplied with SliTaz.

Assign an environment variable (`$fs`)

An environmental variable can't specify the path to the directory, just the name of the directory. We will affect a variable `$fs` to indicate the path to the root filesystem (rootfs). To do this, we venture into the working directory `SliTaz/`, and type:

```
# export fs=$PWD/rootfs
```

To check:

```
# echo $fs
```

8.4.2 bc-1.06 — Text mode calculator

The application **bc** (www.gnu.org/software/bc/[265]) provides a small calculator. When compiling the utility, **dc** is also built, but not installed by SliTaz. Note that **dc** is also available with **BusyBox**. If you decide to copy **dc**, you need to delete the link to **BusyBox** (if it exists). We use a directory `_pkg` (package) for installation, use **strip** to clean the executables and copy the utilities:

```
# cd src
# wget http://ftp.gnu.org/pub/gnu/bc/bc-1.06.tar.gz
# tar xzfv bc-1.06.tar.gz
# cd bc-1.06
# ./configure --prefix=/usr --infodir=/usr/share/info \
  --mandir=/usr/share/man
# make
# make DESTDIR=$PWD/_pkg install
# strip -vs _pkg/usr/bin/*
# cp -avi _pkg/usr/bin/bc $fs/usr/bin
```

[265] http://www.gnu.org/software/bc/

libs

A small **ldd** on bc should produce:

```
libc.so.6 => /lib/libc.so.6 (0x40029000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

8.4.3 zlib-1.2.3 — Compression libraries

The **zlib** (http://www.zlib.net/) package provides compression and decompression functions used by among others, the SSH server **Dropbear** and the X server:

```
# cd ..
# wget http://www.gzip.org/zlib/zlib-1.2.3.tar.bz2
# tar xjfv zlib-1.2.3.tar.bz2
# cd zlib-1.2.3
# ./configure --shared --prefix=/usr
# make
# strip -vs libz.so*
# cp -av libz.so* $fs/usr/lib
```

8.4.4 pcre-7.4 — Perl-compatible regular expressions

The package **pcre** (http://www.pcre.org/) provides libraries of functions for Perl compatible regular expressions used by among others, the web server **Lighttpd**:

```
# cd ..
# wget ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/pcre-7.4.tar.
↪gz
# tar xzfv pcre-7.4.tar.gz
# cd pcre-7.4
# ./configure --prefix=/usr
# make
# make DESTDIR=$PWD/_pkg install
# strip -vs _pkg/usr/bin/*
# strip -vs _pkg/usr/lib/*
# cp -av _pkg/usr/bin/* $fs/usr/bin
# cp -av _pkg/usr/lib/*.so* $fs/usr/lib
```

8.4.5 e2fsprogs-1.39 — Filesystem management utilities

The **e3fsprogs** (http://e2fsprogs.sourceforge.net/) provides utilities for handling ext2 and ext3 filesystems. We will not take all of them because we need the space. It should be noted that we use **fsck** of **BusyBox**:

```
# cd ..
# wget http://puzzle.dl.sourceforge.net/sourceforge/e2fsprogs/e2fsprogs-1.
↪39.tar.gz
# tar xzf e2fsprogs-1.39.tar.gz
# cd e2fsprogs-1.39
# ./configure --prefix=/usr --with-root-prefix="" \
```

(continues on next page)

(continued from previous page)

```
  --enable-elf-shlibs --disable-evms --sysconfdir=/etc \
  --infodir=/usr/share/info --mandir=/usr/share/man
# make
# make DESTDIR=$PWD/_pkg install
# strip -vs _pkg/sbin/*
# strip -vs _pkg/lib/*
# strip -vs _pkg/usr/bin/*
# strip -vs _pkg/usr/sbin/*
# strip -vs _pkg/usr/lib/*
```

Install the utilities, configuration files and libraries in the rootfs of SliTaz. Be careful if you used **fsck**, that you didn't destroy the link to **BusyBox**:

```
# cp -i _pkg/sbin/{badblocks,blkid,dumpe2fs,e2fsck,e2image} $fs/sbin
# cp -i _pkg/sbin/{e2label,findfs,logsave,mke2fs,mkfs.*} $fs/sbin
# cp -i _pkg/sbin/{resize2fs,tune2fs} $fs/sbin
# cp -a _pkg/lib/* $fs/lib
# rm -rf $fs/lib/libss*
# cp -a _pkg/etc/* $fs/etc
# cp -a _pkg/usr/bin/* $fs/usr/bin
# cp -a _pkg/usr/sbin/* $fs/usr/sbin
# cp -ad _pkg/usr/lib/*.so $fs/usr/lib
# rm -rf $fs/usr/lib/libss*
```

You can also copy files from the French locale:

```
# mkdir $fs/usr/share/locale
# cp -a _pkg/usr/share/locale/fr $fs/usr/share/locale
```

8.4.6 Dropbear-0.50 — Lightweight SSH client and server

Dropbear (http://matt.ucc.asn.au/dropbear/dropbear.html) is a small secure client/server supporting SSH 2. **Dropbear** is compatible with **OpenSSH** and uses `~/.ssh/authorized_keys` for the management of public keys. **Dropbear** also provides a version of **scp**, which must be compiled with **make scp**:

```
# cd ..
# wget http://matt.ucc.asn.au/dropbear/releases/dropbear-0.50.tar.gz
# tar xzf dropbear-0.50.tar.gz
# cd dropbear-0.50
# ./configure --prefix=/usr
# make
# make scp
# make DESTDIR=$PWD/_pkg install
# strip -v scp
# strip -v _pkg/usr/bin/*
# strip -v _pkg/usr/sbin/*
```

Install the client and tools in `/usr/bin`, and the server in `/usr/sbin`:

```
# cp scp $fs/usr/bin
# cp -a _pkg/usr/bin/* $fs/usr/bin
# cp -a _pkg/usr/sbin/* $fs/usr/sbin
```

libs

```
libutil.so.1 => /lib/libutil.so.1 (0x40025000)
libz.so.1 => /usr/lib/libz.so.1 (0x40028000)
libcrypt.so.1 => /lib/libcrypt.so.1 (0x4003b000)
libc.so.6 => /lib/libc.so.6 (0x40068000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

Copy the library `libutil.so.1` to `$fs/lib`, if this is not already the case. Other libraries should be present following the construction of the base system:

```
# cp -a /lib/libutil* $fs/lib
```

Configure Dropbear

The user configuration files `authorized_keys` and `known_hosts` are in `~/.ssh`. This directory and the file `known_hosts` are created automatically the first time the user launches **`dbclient`**. The system configuration files for the **`Dropbear`** server are in `/etc/dropbear`:

```
# mkdir $fs/etc/dropbear
```

You must generate the secure keys before starting the **`Dropbear`** server on SliTaz. You can use **`dropbearkey`** with the following commands:

```
# dropbearkey -t rsa -f /etc/dropbear/dropbear_rsa_host_key
# dropbearkey -t dss -f /etc/dropbear/dropbear_dss_host_key
```

On SliTaz, you can start the SSH server with the command:

```
# /etc/init.d/dropbear start
```

8.4.7 lighttpd-1.4.18 — HTTP Web server

`Lighttpd` (www.lighttpd.net[266]) is a light, secure and powerful web server. The project is very active and the server's configuration simple. It supports virtual hosts, CGI scripts, and allows intelligent management of the CPU:

```
# cd ..
# wget http://www.lighttpd.net/download/lighttpd-1.4.18.tar.gz
# tar xzf lighttpd-1.4.18.tar.gz
# cd lighttpd-1.4.18
# ./configure -enable-shared --disable-ipv6 --prefix=/usr \
  --libdir=/usr/lib/lighttpd --mandir=/usr/share/man
# make
# make DESTDIR=$PWD/_pkg install
# strip -vs _pkg/usr/bin/*
# strip -vs _pkg/usr/sbin/*
# strip -vs _pkg/usr/lib/lighttpd/*
```

Install the server and generated libraries. We will then copy some of the modules (9):

[266] http://www.lighttpd.net/

```
# cp _pkg/usr/bin/* $fs/usr/bin
# cp _pkg/usr/sbin/* $fs/usr/sbin
# mkdir $fs/usr/lib/lighttpd
Modules :
# cp _pkg/usr/lib/lighttpd/mod_access.so $fs/usr/lib/lighttpd
# cp _pkg/usr/lib/lighttpd/mod_accesslog.so $fs/usr/lib/lighttpd
# cp _pkg/usr/lib/lighttpd/mod_alias.so $fs/usr/lib/lighttpd
# cp _pkg/usr/lib/lighttpd/mod_auth.so $fs/usr/lib/lighttpd
# cp _pkg/usr/lib/lighttpd/mod_cgi.so $fs/usr/lib/lighttpd
# cp _pkg/usr/lib/lighttpd/mod_compress.so $fs/usr/lib/lighttpd
# cp _pkg/usr/lib/lighttpd/mod_rewrite.so $fs/usr/lib/lighttpd
# cp _pkg/usr/lib/lighttpd/mod_status.so $fs/usr/lib/lighttpd
# cp _pkg/usr/lib/lighttpd/mod_userdir.so $fs/usr/lib/lighttpd
```

libs

There should be a `libdl.so.2` library; if missing, we can copy:

```
# cp -a /lib/libdl* $fs/lib
```

`/var/www` — root of documents served

`/var/www` is the root directory of documents served by default. You can access this via the URL http://localhost/. This directory contains an `index.html` automatically displayed by a query. We will create the directory `/var/www`, to see what's placed inside:

```
# mkdir -p $fs/var/www
```

`lighttpd.conf` — Lighttpd configuration file

The **`Lighttpd`** main configuration file is located at `/etc/lighttpd` and is called `lighttpd.conf`. The configuration file SliTaz provides is self-explanatary, just browse. You can find other examples on the **`Lighttpd`** website and as well as an example configuration in `/doc` in the **`Lighttpd`** archive:

```
# cp -a ../slitaz-tools-1.1/etc/lighttpd $fs/etc
```

Creating the directory containing the log files:

```
# mkdir $fs/var/log/lighttpd
```

User and group www

We will add a user and a group for the web server, it adds security and there is no reason for it to be run a root. The default user on SliTaz is 'www', but you can change this in the configuration file `lighttpd.conf`. The **`BusyBox`** application **`adduser`** has some limitations, so we add user 'www' manually. We also change permissions on the directory of web server logs:

```
# echo "www:x:80:80:www:/var/www:/bin/sh" >> $fs/etc/passwd
# echo "www:*:13509:0:99999:7:::" >> $fs/etc/shadow
# echo "www:*:13509:0:99999:7:::" >> $fs/etc/shadow-
# chroot $fs /bin/ash
/# addgroup -g 80 www
/# chown www.www /var/log/lighttpd
# exit
```

To start the web server, you can use script `/etc/init.d/lighttpd` provided by SliTaz tools, by typing: **`/etc/init.d/lighttpd start`**. You can also automate its launch at boot with a link `/etc/init.d/lighttpd pointing` to `/etc/rc.d/60lighttpd`.

8.4.8 iptables-1.3.7 — Netfilter, Linux firewall

`Netfilter` (www.netfilter.org[267]) is the module which provides the Linux kernel firewall functions, shared internet connections (NAT) and the archiving of network traffic. The **`iptables`** command allows you to configure **`Netfilter`** using **`iptables-restore`** and **`iptable-save`**, to save and restore the **`Netfilter`** configuration:

```
# cd ..
# wget http://www.netfilter.org/projects/iptables/files/iptables-1.3.7.tar.
↪bz2
# tar xjf iptables-1.3.7.tar.bz2
# cd iptables-1.3.7
# make KERNEL_DIR=../linux-2.6.20 BINDIR=/sbin \
  LIBDIR=/lib MANDIR=/usr/share/man
# make KERNEL_DIR=../linux-2.6.20 BINDIR=/sbin \
  LIBDIR=/lib MANDIR=/usr/share/man \
  DESTDIR=$PWD/_pkg install
# strip  _pkg/sbin/*
# strip  _pkg/lib/iptables/*
```

Installing the **`iptables*`** applications and libraries sufficient for a basic firewall:

```
# cp -a _pkg/sbin/iptables* $fs/sbin
# mkdir $fs/lib/iptables
# cp -a _pkg/lib/iptables/{libipt_standard.so,libipt_conntrack.so} \
  $fs/lib/iptables
# cp -a _pkg/lib/iptables/{libipt_tcp.so,libipt_udp.so} $fs/lib/iptables
```

To satisfy the **`iptables`** dependencies, you must copy the `libnsl*` library:

```
# cp -va /lib/libnsl* $fs/lib/tls
# strip $fs/lib/libnsl*
```

8.4.9 sqlite-3.5.1 — Small SQL database engine

This package provides `sqlite3` (www.sqlite.org[268]) and `sqlite3.so*` libraries. **`SQLite`** is fast and efficient and integrates directly to programs using database files:

[267] http://www.netfilter.org/
[268] http://www.sqlite.org/

```
# cd ..
# wget http://www.sqlite.org/sqlite-3.5.1.tar.gz
# tar xzf sqlite-3.5.1.tar.gz
# cd sqlite-3.5.1
# ./configure --prefix=/usr --disable-tcl
# make
# make DESTDIR=$PWD/_pkg install
# strip _pkg/usr/lib/*.so*
# strip _pkg/usr/bin/*
```

Installing the `sqlite3` utility and libraries in the rootfs of SliTaz:

```
# cp -a _pkg/usr/lib/*.so* $fs/usr/lib
# cp -a _pkg/usr/bin/* $fs/usr/bin
```

8.4.10 cdrkit-1.1.5 — Tools for manipulating cdrom and ISO images

cdrkit (www.cdrkit.org[269]) provides tools for manipulating CD-ROMs. SliTaz installs by default **wodim** for burning and **genisoimage** to create an ISO image. The compilation is a bit different (**cmake**), but shouldn't pose any problems:

```
# cd ..
# wget http://cdrkit.org/releases/cdrkit-1.1.5.tar.gz
# tar xzf cdrkit-1.1.5.tar.gz
# cd cdrkit-1.1.5
# make
# make install PREFIX=$PWD/_pkg/usr
# strip -v _pkg/usr/bin/*
# strip -v _pkg/usr/sbin/*
# cp _pkg/usr/bin/genisoimage $fs/usr/bin
# cp _pkg/usr/bin/wodim $fs/usr/bin
```

Copy the library `libcap.so.1` required by `wodim`:

```
# cp -a /lib/libcap.so* $fs/lib
```

8.4.11 cpio-2.8 — Archiver

"cpio" (http://www.gnu.org/software/cpio/) provides tools for manipulating cpio archives. The archive format is used for packages and the SliTaz initramfs image of the CD-ROM. Note that **BusyBox** provides a version of cpio that only unpacks archives:

```
# cd ..
# wget ftp://sunsite.cnlab-switch.ch/mirror/gnu/cpio/cpio-2.8.tar.gz
# tar xzf cpio-2.8.tar.gz
# cd cpio-2.8
# ./configure --prefix=/usr --bindir=/bin \
  --libexecdir=/usr/bin --mandir=/usr/share/man \
  --infodir=/usr/share/info
# make
# make DESTDIR=$PWD/_pkg install
```

(continues on next page)

[269] http://www.cdrkit.org/

(continued from previous page)

```
# strip -v _pkg/bin/*
# strip -v _pkg/usr/bin/*
```

Installing `cpio` in `/bin` and `rmt` in `/usr/bin`. You can also install the French locale files:

```
# cp -a _pkg/bin/* $fs/bin
# cp -a _pkg/usr/bin/* $fs/usr/bin
# cp -a _pkg/usr/share/locale/fr $fs/usr/share/locale
```

8.4.12 microperl-5.8.8 — A tiny Perl

Microperl is a tiny implementation of Perl using the most basic functions of the language. You can find more info in the source archive and the file `README.micro`. We use a small **sed** on the configuration file that searches for microperl modules in `/usr/lib/perl5`. We also create a link to the `#! /usr/bin/perl` script:

```
# wget http://ftp.funet.fi/pub/CPAN/src/perl-5.8.8.tar.gz
# tar xzf perl-5.8.8.tar.gz
# cd perl-5.8.8
# sed -i s/'usr\/local'/'usr'/ uconfig.sh
# sed -i s/'perl5\/5.9'/'perl5'/ uconfig.sh
# sed -i s/'unknown'/'i486-pc-linux-gnu'/ uconfig.sh
# make -f Makefile.micro regen_uconfig
# make -f Makefile.micro
# strip microperl
# cp microperl $fs/usr/bin
# chroot $fs /bin/ash
/# cd /usr/bin
/# ln -s microperl perl
/# exit
```

8.4.13 module-init-tools-3.2 — Utilities for manipulating kernel modules

The module-init-tools[270] from kernel.org: **modprobe**, **insmod**, **rmmod** and **lsmod**. We have chosen to use these because we can compile modutils/modprobe to support compressed (.gz) modules to save space. To do this we use the option `--enable-zlib`, we then clean and copy the binaries. We do not take everything that has been created, only what we need: `depmod`, `insmod`, `modinfo`, `modprobe` and `rmmod` in `/sbin` and `lsmod` in `/bin`:

```
# cd ..
# wget http://ftp.kernel.org/pub/linux/utils/kernel/module-init-tools/
↪module-init-tools-3.2.tar.bz2
# tar xjf module-init-tools-3.2.tar.bz2
# cd module-init-tools-3.2
# ./configure --enable-zlib --prefix=/usr --sbindir=/sbin --bindir=/bin \
  --sysconfdir=/etc --infodir=/usr/share/info --mandir=/usr/share/man
# make
# make DESTDIR=$PWD/_pkg install
# strip -v _pkg/sbin/{depmod,insmod,modinfo,modprobe,rmmod}
# strip -v _pkg/bin/lsmod
```

(continues on next page)

270 http://ftp.kernel.org/pub/linux/utils/kernel/module-init-tools/

(continued from previous page)

```
# cp -i _pkg/sbin/{depmod,insmod,modinfo,modprobe,rmmod} $fs/sbin
# cp -i _pkg/bin/lsmod $fs/bin
# cd ..
```

8.4.14 Copy kernel modules

Copy files from `linux-2.6.20/_pkg`:

```
# cp -a linux-2.6.20/_pkg/lib/* $fs/lib
```

Compress kernel modules

Compress modules, this step will gain us back around 50% of available space. We begin by moving into the rootfs, then we search for all files with the ".ko" extension, and compress them. You can also do this with the `gzmodtaz.sh` script found in SliTaz tools:

```
# cd $fs
```

With `gztazmod.sh`:

```
# cp -v ../src/slitaz-tools-1.1/utils/gztazmod.sh sbin
# ./sbin/gztazmod.sh lib/modules/2.6.20-slitaz
```

Or by hand:

```
# cd lib/modules/2.6.20-slitaz
# find . -name "*.ko" -exec gzip '{}' \;
# sed 's/\.ko/.ko.gz/g' modules.dep > tmp.dep
# rm modules.dep
# mv tmp.dep modules.dep
```

8.4.15 Generate the initramfs and an ISO image

To create a new ISO image, you can use **`mktaziso`** in *SliTaz tools* (page 380). Or you can create a new initramfs image, copy it to `/boot` in the root of the CD-ROM (rootcd) and finally generate an ISO image with **`genisoimage`**:

```
# cd $fs
# find . -print | cpio -o -H newc | gzip -9 > ../rootfs.gz
# cd ..
# cp rootfs.gz rootcd/boot
# genisoimage -R -o slitaz-test.iso -b boot/isolinux/isolinux.bin \
  -c boot/isolinux/boot.cat -no-emul-boot -boot-load-size 4 \
  -V "SliTaz" -input-charset iso8859-1 -boot-info-table rootcd
```

Test ISO image:

```
# qemu -cdrom slitaz-test.iso
```

Following chapter

The next chapter is called *Ncurses libraries and applications* (page 419). It covers the installation and configuration of the ncurses libraries and applications.

8.5 Ncurses libraries and applications

author domcox

Installation and configuration of ncurses libraries and applications.

8.5.1 About

This chapter describes the construction and installation of some ncurses applications and libraries in SliTaz. The procedure consists of moving into the `/src` directory, downloading the sources for the application in question, unpacking, reading the README or INSTALL file(s), compiling and installing the binary in SliTaz. Once the applications are installed, we can create a new initramfs, copy it to the root of the CD-ROM and generate a new ISO image. For this you can also use **`mktaziso`** in *SliTaz Tools* (page 380).

8.5.1.1 Assign an environment variable (`$fs`)

An environmental variable can't specify the path to the directory, just the name of the directory. We will affect a variable `$fs` to indicate the path to the root filesystem (rootfs). To do this, we venture into the working directory `SliTaz/`, and type:

```
# export fs=$PWD/rootfs
```

To check:

```
# echo $fs
```

8.5.2 ncurses-5.6 — Terminal utilities and libraries

`ncurses` (dickey.his.com/ncurses/[271]) contains functions to display text in different ways on the screen of a Linux terminal and also provides the terminfo file. Ncurses libraries are used among others by **`retawq`**, **`nano`** and some games. We install the libraries in `/lib` and the rest in `/usr/bin` with a small **`strip`** to clean the executables:

```
# cd src
# wget ftp://invisible-island.net/ncurses/ncurses-5.6.tar.gz
# tar xzf ncurses-5.6.tar.gz
# cd ncurses-5.6
# ./configure --prefix=/usr \
  --libdir=/lib --sysconfdir=/etc \
  --infodir=/usr/share/info --mandir=/usr/share/man \
  --with-shared --without-debug --without-ada
# make
```

(continues on next page)

[271] http://dickey.his.com/ncurses/

(continued from previous page)

```
# make DESTDIR=$PWD/_pkg install
# strip -v _pkg/lib/*
# strip -v _pkg/usr/bin/*
```

Copy the `libncurses` library and some applications in SliTaz. The `reset` utility is used at the end of the rcS initialization script to remove the Linux logo. If you wish, you can copy the associated utilities (`tic`, `tack`, `toe`, etc), being careful not to delete the link to **BusyBox**:

```
# cp -a _pkg/lib/libncurses.so* $fs/lib
# cp -ia _pkg/usr/bin/{clear,ncurses5-config,tset,reset} \
  $fs/usr/bin
```

Copy `terminfo` files, we only use a few files. If you want more, you can copy:

```
# mkdir -v $fs/usr/share/terminfo
# mkdir -v $fs/usr/share/terminfo/{a,l,r,v,x}
# cp _pkg/usr/share/terminfo/a/ansi \
  $fs/usr/share/terminfo/a
# cp _pkg/usr/share/terminfo/l/linux \
  $fs/usr/share/terminfo/l
# cp _pkg/usr/share/terminfo/r/rxvt \
  $fs/usr/share/terminfo/r
# cp _pkg/usr/share/terminfo/x/{xterm,xterm-color,xterm-new,xterm-vt220} \
  $fs/usr/share/terminfo/x
# cp _pkg/usr/share/terminfo/v/{vt100,vt102*} \
  $fs/usr/share/terminfo/v
```

Copy the `tabset` files:

```
# cp -a _pkg/usr/share/tabset $fs/usr/share
```

8.5.3 clex-3.16 — File Manager

CLEX (http://www.clex.sk/) is a small ncurses file manager (160 KB). The configuration file (rc) is `~/clexrc`; `~/.clexbm` is used for bookmarks:

```
# cd ..
# wget http://www.clex.sk/download/clex-3.16.tar.gz
# tar xzf clex-3.16.tar.gz
# cd clex-3.16
# ./configure --prefix=/usr --infodir=/usr/share/info \
  --mandir=/usr/share/man
# make
# make DESTDIR=$PWD/_pkg install
# strip -v _pkg/usr/bin/clex
```

Install the `clex` binary in the rootfs of SliTaz:

```
# cp _pkg/usr/bin/clex $fs/usr/bin
```

libs

If we execute the **ldd** command on `clex`, the following dependancies should be displayed:

```
libncurses.so.5 => /lib/libncurses.so.5 (0x40025000)
libc.so.6 => /lib/libc.so.6 (0x40064000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

8.5.4 nano-2.0.6 — Advanced Text Editor

GNU **nano** (www.nano-editor.org[272]) is a well known, fast, effective GNU/Linux text editor that supports colored syntax. This is the default text editor in SliTaz:

```
# cd ..
# wget http://www.nano-editor.org/dist/v2.0/nano-2.0.6.tar.gz
# tar xzf nano-2.0.6.tar.gz
# cd nano-2.0.6
# ./configure --enable-all --enable-extra --prefix=/usr \
  --infodir=/usr/share/info --mandir=/usr/share/man \
  --sysconfdir=/etc
# make
# make DESTDIR=$PWD/_pkg install
# strip -v _pkg/usr/bin/nano
```

Copy the `nano` binary and the `rnano` link in SliTaz:

```
# cp -a _pkg/usr/bin/* $fs/usr/bin
```

Copy the configuration files in `_pkg/usr/share/nano` to our rootfs:

```
# cp -a _pkg/usr/share/nano $fs/usr/share
```

libs

If we execute the **ldd** command on `nano`, the following dependancies should be displayed:

```
libncurses.so.5 => /lib/libncurses.so.5 (0x40025000)
libc.so.6 => /lib/libc.so.6 (0x40064000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

locale

About the language, you can copy the `.mo` files created when installing **nano** from `/usr/share/locale/(fr,en,de,es,etc)/LC_MESSAGES` to the rootfs. Example for the French language:

```
# cp -a _pkg/usr/share/locale/fr/LC_MESSAGES $fs/usr/share/locale/fr
```

Customize nano

You can customize **nano** via `/etc/nanorc` or `~/.nanorc` for each user of the system. It's in this file that you can define the colors used by **nano** through the files in `/usr/share/nano`. You will find a broad example of this file in the archive of **nano** and *SliTaz Tools* (page 380).

[272] http://www.nano-editor.org/

For a system configuration file, you can copy the file in SliTaz tools to `/etc` in the rootfs:

```
# cd ..
# cp -a slitaz-tools-1.1/etc/nanorc $fs/etc
```

8.5.5 retawq-0.2.6c — Text mode Web browser

retawq (retawq.sourceforge.net[273]) is a small text-only web browser. We only flag a few useful options when configuring, **retawq** needs terminfo files, libncurses libraries and libthread:

```
# wget http://switch.dl.sourceforge.net/sourceforge/retawq/retawq-0.2.6c.
↪tar.gz
# tar xzf retawq-0.2.6c.tar.gz
# cd retawq-0.2.6c
# ./configure --enable-i18n --enable-local-cgi --path-prefix=/usr \
  --path-doc=/usr/share/doc/retawq --path-man=/usr/share/man
# make
# strip -v retawq
```

Copy the `retawq` binary in SliTaz:

```
# cp retawq $fs/usr/bin
```

libs

```
libncurses.so.5 => /lib/libncurses.so.5 (0x40025000)
libpthread.so.0 => /lib/libpthread.so.0 (0x40064000)
libc.so.6 => /lib/libc.so.6 (0x40074000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

You can copy the lipthread library from your host system or use the minimum package glibc-2.3.6 distributed by SliTaz:

```
# cp -a /lib/libpthread* $fs/lib
# strip --strip-unneeded $fs/lib/*
```

locale

For language, you can copy the `.mo` files in `/i18n` of the **retawq** archive to `/usr/share/locale/(fr,en,es,etc)/LC_MESSAGES`. Example for the French language, renaming the file to `retawq.mo`:

```
# cp -v i18n/fr.mo $fs/usr/share/locale/fr/LC_MESSAGES/retawq.mo
```

Customize retawq

To personalize **retawq**, you can use a `~/.retawq` directory containing a config file. You can also save bookmarks (html) in the root directory of the user. You will find an `examples/` in the archive of

[273] http://retawq.sourceforge.net/

retawq (or SliTaz tools) containing a `bookmarks.html` page with a list of favorite web sites. You can also copy the docs (`/documents`) from **retawq** to `/usr/share/doc/retawq`.

8.5.6 htop-6.0.5 — System process viewer

htop (htop.sourceforge.net/[274]) is software that displays system processes using ncurses.

Returning to the `/src` directory, download, unpack, configure, compile and clean (with **strip**):

```
# cd ..
# wget http://switch.dl.sourceforge.net/sourceforge/htop/htop-0.6.5.tar.gz
# tar xzf htop-0.6.5.tar.gz
# cd htop-0.6.5
# ./configure --prefix=/usr --mandir=/usr/share/man
# make
# make DESTDIR=$PWD/_pkg install
# strip -v _pkg/usr/bin/htop
```

Copy the `htop` binary in SliTaz:

```
# cp _pkg/usr/bin/htop $fs/usr/bin
```

You can still copy the **htop** icon found in: `_pkg/usr/share/pixmaps`.

libs

```
libm.so.6 => /lib/libm.so.6 (0xb7f97000)
libncurses.so.5 => /lib/libncurses.so.5 (0xb7f55000)
libc.so.6 => /lib/libc.so.6 (0xb7e20000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0xb7fc9000)
```

8.5.7 dialog-1.1.20070409 — GUI shell scripts

dialog (invisible-island.net/dialog/dialog.html[275]), is a utility to build GUI-based consoles:

```
# cd ..
# wget ftp://invisible-island.net/dialog/dialog.tar.gz
# tar xzf dialog.tar.gz
# cd dialog-1.1-20070409
# ./configure --enable-nls --with-ncurses --prefix=/usr \
  --sysconfdir=/etc --mandir=/usr/share/man
# make
# make DESTDIR=$PWD/_pkg install
# strip -v _pkg/usr/bin/dialog
```

Copy `dialog` binary in SliTaz:

```
# cp _pkg/usr/bin/dialog $fs/usr/bin
```

[274] http://htop.sourceforge.net/
[275] http://invisible-island.net/dialog/dialog.html

libs

```
libncurses.so.5 => /lib/libncurses.so.5 (0x40027000)
libm.so.6 => /lib/libm.so.6 (0x40066000)
libc.so.6 => /lib/libc.so.6 (0x40089000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

locale

You can install locale files if you wish:

```
# cp -a _pkg/usr/share/locale/fr $fs/usr/share/locale
```

The **dialog** configuration file is `/etc/dialogrc` and/or `~/.dialogrc` for each user. There are also full examples of scripts in the `/sample` directory in the sources of **dialog**.

8.5.8 Ninvaders-0.1.1 — Space Invaders clone

ninvaders (http://ninvaders.sourceforge.net/) is a clone of the popular **Space Invaders** game (46 KB). We begin by placing ourselves in the `/src` directory, then we download, untar, compile, clean using **strip** and copy the `nInvaders` binary in `/usr/games` of SliTaz:

```
# cd ..
# wget http://ovh.dl.sourceforge.net/sourceforge/ninvaders/ninvaders-0.1.1.
↪tar.gz
# tar xzf ninvaders-0.1.1.tar.gz
# cd ninvaders-0.1.1
# make
# strip -v nInvaders
# cp nInvaders $fs/usr/games
```

8.5.9 bastet-0.41 — Bastard Tetris clone

A game of Tetris (17 KB):

```
# wget http://fph.altervista.org/prog/bastet-0.41.tgz
# tar xzf bastet-0.41.tgz
# cd bastet-0.41
# make
# strip bastet
# cp bastet $fs/usr/games
# mkdir -p $fs/var/games
# touch $fs/var/games/bastet.scores
# chmod 666 $fs/var/games/bastet.scores
```

8.5.10 rhapsody-0.28b — IRC chat client

Rhapsody (http://rhapsody.sourceforge.net/) is a fast and lightweight chat client supporting the IRC protocol. It provides a menu for managing servers, channels and configuration. It is therefore easy to use:

```
# cd ..
# wget http://switch.dl.sourceforge.net/sourceforge/rhapsody/rhapsody_0.
↪28b.tgz
# tar xzf rhapsody_0.28b.tgz
# cd rhapsody-0.28b
# ./configure -i /usr/bin -d /usr/share/doc/rhapsody
# make
# strip -v rhapsody
```

Install the binary and help files in SliTaz. We must adjust permissions on these files so that everyone can read:

```
# cp rhapsody $fs/usr/bin
# mkdir $fs/usr/share/doc/rhapsody
# cp -a help $fs/usr/share/doc/rhapsody/help
# chmod 644 $fs/usr/share/doc/rhapsody/help/*
```

libs

Rhapsody uses the following libraries:

```
libncurses.so.5 => /lib/libncurses.so.5 (0x40026000)
libc.so.6 => /lib/libc.so.6 (0x40066000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000))
```

Rhapsody use the following libraries:

You can customize **rhapsody** via `~/.rhapsodyrc` or use `Ctrl+T` for options:

8.5.11 Generate the initramfs and an ISO image

To create a new ISO image, you can use **mktaziso** in SliTaz tools. Or you can create a new initramfs image, copy it to `/boot` in the root of the CD-ROM (rootcd) and finally generate an ISO image with **genisoimage**:

```
# cd $fs
# find . -print | cpio -o -H newc | gzip -9 > ../rootfs.gz
# cd ..
# cp rootfs.gz rootcd/boot
# genisoimage -R -o slitaz-cooking.iso -b boot/isolinux/isolinux.bin \
  -c boot/isolinux/boot.cat -no-emul-boot -boot-load-size 4 \
  -V "SliTaz" -input-charset iso8859-1 -boot-info-table rootcd
```

Following chapter

The next chapter describes the installation of the *locales* (page 426) and i18n.

8.6 Locale & i18n

author domcox

Installation and configuration of locales.

8.6.1 Locale installation

This chapter describes the installation of locales in SliTaz GNU/Linux from a SliTaz GNU/Linux host system. The installation of locales contained in the X server are described in the chapter *X window system* (page 430). The various files copied in this chapter come from the compiliation package glibc-2.3.6 forming part of the *toolchain*.

Various file directories

We begin by creating the directories that contain libraries and files relevant to the different locales. The directory `LC_MESSAGES` contains the files for the translated messages (`.mo`), if they exist:

```
# mkdir -p rootfs/usr/share/{i18n,locale}
# mkdir -p rootfs/usr/lib/{locale,gconv}
# mkdir -p rootfs/usr/share/i18n/{charmaps,locales}
# mkdir -p rootfs/usr/share/locale/fr/LC_MESSAGES
```

Copy the localization files for French, Swiss-French and Swiss-German in `/usr/share/i18n/locales`:

```
# cp -a /usr/share/i18n/locales/{de_CH,fr_CH,fr_FR,i18n,iso14651_t1} \
  rootfs/usr/share/i18n/locales
```

Copy the `translit_*` files in `/usr/share/i18n/locales`:

```
# cp -a /usr/share/i18n/locales/{translit_circle,translit_cjk_compat} \
  rootfs/usr/share/i18n/locales
# cp -a /usr/share/i18n/locales/{translit_combining,translit_compat} \
  rootfs/usr/share/i18n/locales
# cp -a /usr/share/i18n/locales/{translit_font,translit_fraction} \
  rootfs/usr/share/i18n/locales
# cp -a /usr/share/i18n/locales/{translit_narrow,translit_neutral} \
  rootfs/usr/share/i18n/locales
# cp -a /usr/share/i18n/locales/{translit_small,translit_wide} \
  rootfs/usr/share/i18n/locales
```

Copy the `charmaps` files in `/usr/share/i18n/charmaps`:

```
# cp -a /usr/share/i18n/charmaps/ANSI_X3.* rootfs/usr/share/i18n/charmaps
# cp -a /usr/share/i18n/charmaps/{ISO-8859-1.gz,ISO-8859-2.gz,ISO-8859-15.
↪gz} \
  rootfs/usr/share/i18n/charmaps
```

Copy the `gconv` libraries in `/usr/lib/gconv` to rootfs of SliTaz:

```
# cp /usr/lib/gconv/{ANSI_X3.110.so,gconv-modules,UNICODE.so} \
  rootfs/usr/lib/gconv
```

(continues on next page)

(continued from previous page)

```
# cp /usr/lib/gconv/{ISO8859-1.so,ISO8859-2.so,ISO8859-15.so} \
  rootfs/usr/lib/gconv
# strip -v rootfs/usr/lib/gconv/*.so
```

Copy the `locale` utility:

```
# cp /usr/bin/locale rootfs/usr/bin
```

It's necessary that the file `/usr/lib/locale/locale-archive` is generated, for that we use the **`localedef`** utility while chrooted in SliTaz:

```
# cp /usr/bin/localedef rootfs/usr/bin
# chroot rootfs /bin/ash
```

Use of **`localedef`** for French-speaking Switzerland and France:

```
# localedef -i fr_CH -f ISO-8859-1 fr_CH
# localedef -i fr_FR -f ISO-8859-1 fr_FR
# exit
```

You can delete the `localedef` binary to gain some space:

```
# rm rootfs/usr/bin/localedef
```

8.6.2 Config and use of locale

To use a language in a session, you can create a script launched at boot, or add 2 lines to the `~/.profile` specific to each user with:

```
export LANG=fr_CH
export LC_ALL=fr_CH
```

Voilà, the French language should now function. If you installed **`retawq`** or **`nano`**, you can check the performance of locales by copying the `.mo` files in the sources of **`Retawq`** or **`Nano`** to `/usr/share/locale/fr/LC_MESSAGES`.

Following chapter

SliTaz uses the `/etc/init.d/i18n.sh` script and the `/etc/locale.conf` configuration file to manage the system locale. This is detailed in the next chapter *Boot scripts* (page 427). On a working system, just modify `/etc/locale.conf` with a text editor or launch **`tazlocale`** to change the default system locale Or to specify the language as a boot option: `lang=xx`.

8.7 Boot scripts

author domcox

The startup and shutdown scripts with their configuration files.

8.7.1 SliTaz and startup

SliTaz does not use a level of execution (runlevel), the system is initialized via a primary script and its main configuration file. This script itself launches some other smaller scripts which deal with the internationalization or the commands placed for the system to start.

8.7.2 `/etc/init.d/*` — Directory of scripts and daemons

The directory `/etc/init.d` contains all of the rc scripts, scripts finishing with `.sh` are simple shell scripts and daemons such as `dropbear` or `lighttpd` are scripts that launch a service. The daemon scripts can start, stop or restart through the command:

```
# /etc/init.d/daemon [start|stop|restart]
```

On SliTaz you will find a `/etc/init.d/README` describing the basic function of rc scripts. Also note that all startup scripts and daemons can call upon the `/etc/init.d/rc.functions` file. This file makes it possible to include various functions in rc scripts. SliTaz uses a function `status` to check whether the previous command has succeeded (0) or not.

8.7.3 `/etc/init.d/rcS` — Primary initialization script

The `/etc/init.d/rcS` script configures all the basic services and initializes the base system. It begins by mounting the filesystems and starts services like `syslogd`, `klogd`, `mdev` and cleans up the system and so on. It uses the configuration file `/etc/rcS.conf` to locate which daemons and scripts to launch at startup. You can browse the script to know which commands are executed:

```
# nano rootfs/etc/init.d/rcS
```

8.7.4 Specific scripts and daemons

`bootopts.sh` — LiveCD mode options

This script is used to configure the LiveCD options passed at boot time and is readable via the `/proc/cmdline` file. This is the script that allows you to use a USB key or external hard disk `/home` partition with the option `home=usb` or `home=sda[1-9]` or directly specify the language and keyboard parameters.

`network.sh` — Initializing the network

This script searches the `network.sh` configuration file `/etc/network.conf` for the network interface to use; if one wants to launch the DHCP client (or not) or if you want to use a fixed (static) IP. On SliTaz the `/etc/init.d/network.sh` script configures the network interfaces to start using the information contained in `/etc/network.conf`. If the variable `$DHCP` is equal to `yes`, then the `/etc/init.d/network.sh` script launches the DHCP client on the `$INTERFACE` interface.

`i18n.sh` — Internationalization

SliTaz backs up the configuration of the default locale in `/etc/locale.conf` which is read by `/etc/profile` at each login. The `/etc/locale.conf` is generated during boot time thanks to the `/etc/i18n.sh` script. This script launches the **`tazlocale`** application if `/etc/locale.conf` doesn't exist. We use the same process for the keyboard layout using **`tazkmap`** and the `/etc/kmap.conf` configuration file. Both applications are installed and located in `/sbin` and use **`dialog`** and the **`ncurses`** library. The script also checks whether the configuration file for the time zone `/etc/TZ` exists, otherwise it creates one relying on the keyboard configuration.

`local.sh` — Local commands

The `/etc/init.d/local.sh` script allows the system administrator to add local commands to be executed at boot. Example:

```
#!/bin/sh
# /etc/init.d/local.sh: Local startup commands.
# All commands here will be executed at boot time.
#
. /etc/init.d/rc.functions

echo "Starting local startup commands... "
```

file:*rc.shutdown*

This script is invoked by `/etc/inittab` during system shutdown. It also stops all daemons via the variable `RUN_DAEMONS` in the primary `/etc/rcS.conf` configuration file.

8.7.5 `/etc/inittab` — Configuration file init

The first file read by the Kernel at boot. It defines the initialization script (`/etc/init.d/rcS`), shells (ttys) and actions in the event of a reboot or disruption. You will find a complete example with accompanying notes in *SliTaz Tools* (page 380):

```
# /etc/inittab: init configuration for SliTaz GNU/Linux.
# Boot-time system configuration/initialization script.
#
::sysinit:/etc/init.d/rcS

# /sbin/getty respawn shell invocations for selected ttys.
tty1::respawn:/sbin/getty 38400 tty1
tty2::respawn:/sbin/getty 38400 tty2
tty3::respawn:/sbin/getty 38400 tty3
tty4::respawn:/sbin/getty 38400 tty4
tty5::respawn:/sbin/getty 38400 tty5
tty6::respawn:/sbin/getty 38400 tty6

# Stuff to do when restarting the init
# process, or before rebooting.
::restart:/etc/init.d/rc.shutdown
::restart:/sbin/init
```

(continues on next page)

(continued from previous page)

```
::ctrlaltdel:/sbin/reboot
::shutdown:/etc/init.d/rc.shutdown
```

Following chapter

The next chapter continues on with the *X window system* (page 430).

8.8 X window system

author domcox

Installation and basic configuration of the X window system.

8.8.1 About

This chapter describes the installation and configuration of the X window system on SliTaz. We will install libraries for expat, XML, fonts, a graphical server (Xvesa), a terminal emulator (xterm), various small tools and a window manager (JWM). We'll also install the JPEG libraries and Links web browser.

Environmental variable (`$fs`)

If you do not specify any path to the rootfs directory, export the environmental variable:

```
# export fs=$PWD/rootfs
```

To check:

```
# echo $fs
```

8.8.2 expat-2.0.0 — XML parser library

Expat (http://expat.sourceforge.net/) contains the XML parsing libraries:

```
# cd ..
# wget http://switch.dl.sourceforge.net/sourceforge/expat/expat-2.0.0.tar.
↪gz
# tar xzf expat-2.0.0.tar.gz
# cd expat-2.0.0
# ./configure --sysconfdir=/etc --prefix=/usr \
  --mandir=/usr/share/man
# make
# make DESTDIR=$PWD/_pkg install
# strip -v _pkg/usr/lib/*
# strip -v _pkg/usr/bin/*
```

Thereafter, we will install 'xterm' which needs libexpat.so.0, simply create a symbolic link and voilà. Then you can install the 'xmlwf' application and libraries in the rootfs:

```
# cd _pkg/usr/lib
# ln -s libexpat.so.1.5.0 libexpat.so.0
# cp -a *.so* $fs/usr/lib
# cd ..
# cp -a bin/* $fs/usr/bin
# cd ../..
```

libs

Libraries used by xmlwf:

```
libexpat.so.1 => /usr/lib/libexpat.so.1 (0x40021000)
libc.so.6 => /lib/tls/libc.so.6 (0x40041000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

8.8.3 freetype-2.3.1 — System font libraries

The package freetype (http://www.freetype.org/) contains libraries used by X for configuring the system fonts:

```
# cd ..
# wget http://download.savannah.gnu.org/releases/freetype/freetype-2.3.1.
↪tar.bz2
# tar xjf freetype-2.3.1.tar.bz2
# cd freetype-2.3.1
# ./configure --sysconfdir=/etc --prefix=/usr \
  --mandir=/usr/share/man
# make
# make DESTDIR=$PWD/_pkg install
# strip -vs _pkg/usr/lib/*
# cp -a _pkg/usr/bin/* $fs/usr/bin
# cp -a _pkg/usr/lib/*.so* $fs/usr/lib
```

8.8.4 fontconfig-2.4.2 — Manage system fonts

The fontconfig package (www.fontconfig.org/wiki/[276]) provides the libfontconfig library used by many programs under X. Note XFree86 also provides these utilities. We chose the original package because it works better with JWM:

```
# cd ..
# wget http://fontconfig.org/release/fontconfig-2.4.2.tar.gz
# tar xzf fontconfig-2.4.2.tar.gz
# cd fontconfig-2.4.2
# ./configure --sysconfdir=/etc --prefix=/usr \
  --mandir=/usr/share/man --localstatedir=/var
# make
# make DESTDIR=$PWD/_pkg install
# strip -v _pkg/usr/bin/*
# strip -v _pkg/usr/lib/*
```

(continues on next page)

[276] http://www.fontconfig.org/wiki/

(continued from previous page)

```
# cp -a _pkg/usr/bin/* $fs/usr/bin
# cp -a _pkg/usr/lib/*.so* $fs/usr/lib
# cp -a _pkg/etc $fs
# cp -a _pkg/var $fs
```

libs

A 'ldd' on fc-cache gives the libraries below. You can also use libfreetype of XFree86:

```
libfreetype.so.6 => /usr/lib/libfreetype.so.6 (0xb7f12000)
libz.so.1 => /usr/lib/libz.so.1 (0xb7eff000)
libexpat.so.1 => /usr/lib/libexpat.so.1 (0xb7edf000)
libfontconfig.so.1 => /usr/lib/libfontconfig.so.1 (0xb7eb0000)
libc.so.6 => /lib/tls/libc.so.6 (0xb7d7b000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0xb7f8c000)
```

8.8.5 Xserver — Graphical Xvesa server of Xfree86

We will use the binary versions of Xvesa server (www.xfree86.org/[277]) and fonts distributed by Xfree86.org. We could also copy Xorg libraries from the host system that would be used for compiling X applications. Xvesa works well like this and simplifies things. You can also rebuild *Xorg packages* (page 446) on your development system.

Xtinyx server — Xvesa

The Xvesa server is very light and uses tiny libraries; it is contained in the `Xtinyx.tgz` archive. Download and install in `/usr/bin` of SliTaz rootfs:

```
# cd ..
# mkdir -p XFree86-4.6.0 && cd XFree86-4.6.0
# wget http://ftp.xfree86.org/pub/XFree86/4.6.0/binaries/Linux-ix86-
↪glibc23/Xtinyx.tgz
# tar xzf Xtinyx.tgz
# cp bin/Xvesa $fs/usr/bin
# strip $fs/usr/bin/Xvesa
# chmod 4711 $fs/usr/bin/Xvesa
```

libs for Xvesa

```
libz.so.1 => /usr/lib/libz.so.1 (0xb7ed6000)
libm.so.6 => /lib/tls/libm.so.6 (0xb7eb1000)
libc.so.6 => /lib/tls/libc.so.6 (0xb7d7e000)
/lib/ld-linux.so.2 (0xb7ef8000)
```

[277] http://www.xfree86.org/

rgb.txt — RGB colors in X

The colors configuration file used by the X server is called: `rgb.txt`; we suggest that you copy it to the host system. The library `libX11.so` will seek the configuration files in `/usr/share/X11`, and the Xvesa server in `/usr/X11R6/lib/X11`; we create a link in `/usr/share/X11` to enable this:

```
# mkdir -p $fs/usr/share/X11
# cp /usr/share/X11/rgb.txt $fs/usr/share/X11
# chroot $fs /bin/ash
# mkdir -p /usr/X11R6/lib/X11/
# ln -s /usr/share/X11/rgb.txt /usr/X11R6/lib/X11/rgb.txt
# exit
```

Xfnts — Fonts

To operate the server, we need the basic fonts; you can download them from Xfree86.org and then compile packages from Xorg, or copy them from your host system. The system fonts can be put into different folders and the cache updated with **`lc-cache`**. Attention, fonts take pride of place and you can only install the minimum. `/usr/share/fonts` contains the TrueType fonts such as bitstream-vera:

```
# wget http://ftp.xfree86.org/pub/XFree86/4.6.0/binaries/Linux-ix86-
↪glibc23/Xfnts.tgz
# tar xzf Xfnts.tgz
# mkdir -p $fs/usr/X11R6/lib/X11/fonts
# mkdir -p $fs/usr/share/fonts/truetype
```

Copy the fonts...

```
# cp -a lib/X11/fonts/* $fs/usr/X11R6/lib/X11/fonts)
# cp -a /usr/share/fonts/truetype/* $fs/usr/share/fonts/truetype
```

Then regenerate the `fonts.dir` file, you must run **`mkfontdir`** on the directory in question:

```
# mkfontdir $fs/usr/X11R6/lib/X11/fonts/75dpi
```

Fontconfig configuration files can be found in `/etc/fonts` provided by the **`fontconfig`** package. Now you can run **`fc-cache`** to update the cache, and **`fc-list`** for a list of fonts. You do this by chrooting into the rootfs:

```
# chroot $fs /bin/ash
# fc-cache -v
# fc-list
# exit
```

Xlib locale — Localization files

On SliTaz, we installed 4 locales: `C`, `iso8859-1`, `iso8859-15` and `iso8859-2` from the *compilation of Xorg* (page 446). You can copy these files from the host system or use the files distributed by XFree86. Sample copy of all the locales from the host system:

```
# mkdir -p $fs/usr/share/X11/locale
# cp -a /usr/share/X11/locale/* $fs/usr/share/X11/locale
```

Using X

Note that you can already use Xvesa as a X terminal if you have a machine on the network accepting XDMCP connections. For this, you can start the server with the `-query` option. For example:

```
# Xvesa -ac -shadow -screen 1024x768x24 -query 192.168.0.2
```

8.8.6 xterm — Terminal Emulator

The **`xterm`** package (invisible-island.net/xterm/[278]) provides a terminal emulator for X:

```
# wget ftp://invisible-island.net/xterm/xterm-223.tgz
# tar xzf xterm-223.tgz
# cd xterm-223
# ./configure --prefix=/usr --sysconfdir=/etc \
  --mandir=/usr/share/man --localstatedir=/var \
  --with-app-defaults=/usr/share/X11/app-defaults \
  --build=i486-pc-linux-gnu --host=i486-pc-linux-gnu
# make
# make DESTDIR=$PWD/_pkg install
# strip _pkg/usr/bin/*
# cp _pkg/usr/bin/* $fs/usr/bin
# cp -a _pkg/usr/share/X11/* $fs/usr/share/X11
```

libs

A **`ldd`** on XTerm, we copy (and **`strip`**) the missing libraries from the host system:

```
libXft.so.2 => /usr/lib/libXft.so.2 (0xb7f09000)
libXrender.so.1 => /usr/lib/libXrender.so.1 (0xb7f00000)
libfontconfig.so.1 => /usr/lib/libfontconfig.so.1 (0xb7ed5000)
libfreetype.so.6 => /usr/lib/libfreetype.so.6 (0xb7e68000)
libz.so.1 => /usr/lib/libz.so.1 (0xb7e54000)
libX11.so.6 => /usr/lib/libX11.so.6 (0xb7d68000)
libXaw.so.7 => /usr/lib/libXaw.so.7 (0xb7d0f000)
libXmu.so.6 => /usr/lib/libXmu.so.6 (0xb7cfa000)
libXext.so.6 => /usr/lib/libXext.so.6 (0xb7cec000)
libXt.so.6 => /usr/lib/libXt.so.6 (0xb7c9e000)
libSM.so.6 => /usr/lib/libSM.so.6 (0xb7c96000)
libICE.so.6 => /usr/lib/libICE.so.6 (0xb7c7f000)
libncurses.so.5 => /lib/libncurses.so.5 (0xb7c3c000)
libc.so.6 => /lib/libc.so.6 (0xb7b2c000)
libexpat.so.1 => /usr/lib/libexpat.so.1 (0xb7b0b000)
libXau.so.6 => /usr/lib/libXau.so.6 (0xb7b08000)
libXdmcp.so.6 => /usr/lib/libXdmcp.so.6 (0xb7b03000)
libdl.so.2 => /lib/libdl.so.2 (0xb7aff000)
libXpm.so.4 => /usr/lib/libXpm.so.4 (0xb7aee000)
```

8.8.7 libpng-1.2.18 — PNG Libraries

PNG libraries (http://libpng.org/pub/png/libpng.html) are used to manipulate and format PNG images:

[278] http://invisible-island.net/xterm/

```
# wget http://puzzle.dl.sourceforge.net/sourceforge/libpng/libpng-1.2.18.
↪tar.bz2
# tar xjf libpng-1.2.18.tar.bz2
# cd libpng-1.2.18
# ./configure --enable-shared --prefix=/usr \
  --mandir=/usr/share/man
# make
# make DESTDIR=$PWD/_pkg install
# strip _pkg/usr/lib/*.so*
# cp -a _pkg/usr/lib/libpng12.so* $fs/usr/lib
# cp -a _pkg/usr/bin/libpng12* $fs/usr/bin
```

8.8.8 jwm-2.0 — Window manager

Joe's Window Manager (http://www.joewing.net/programs/jwm/) is an ultra light and friendly window manager. This is the default SliTaz window manager. The main configuration file `/etc/jwm/system.jwmrc` includes the style and config menu:

```
# cd ..
# wget http://www.joewing.net/programs/jwm/releases/jwm-2.0.tar.bz2
# tar xjf jwm-2.0.tar.bz2
# cd jwm-2.0
# ./configure --prefix=/usr --mandir=/usr/share/man \
  --sysconfdir=/etc/jwm --disable-xinerama
# make
# strip src/jwm
# cp src/jwm $fs/usr/bin
# mkdir $fs/etc/jwm
# cp example.jwmrc $fs/etc/jwm/system.jwmrc
```

libs

ldd libraries that we have provided:

```
libX11.so.6 => /usr/lib/libX11.so.6 (0xb7e35000)
libpng12.so.0 => /usr/lib/libpng12.so.0 (0xb7e12000)
libXft.so.2 => /usr/lib/libXft.so.2 (0xb7e00000)
libXrender.so.1 => /usr/lib/libXrender.so.1 (0xb7df7000)
libfontconfig.so.1 => /usr/lib/libfontconfig.so.1 (0xb7dcc000)
libfreetype.so.6 => /usr/lib/libfreetype.so.6 (0xb7d5f000)
libz.so.1 => /usr/lib/libz.so.1 (0xb7d4a000)
libXpm.so.4 => /usr/lib/libXpm.so.4 (0xb7d3a000)
libXext.so.6 => /usr/lib/libXext.so.6 (0xb7d2c000)
libc.so.6 => /lib/libc.so.6 (0xb7c1c000)
libXau.so.6 => /usr/lib/libXau.so.6 (0xb7c19000)
libXdmcp.so.6 => /usr/lib/libXdmcp.so.6 (0xb7c14000)
libdl.so.2 => /lib/libdl.so.2 (0xb7c0f000)
libm.so.6 => /lib/libm.so.6 (0xb7bea000)
libexpat.so.1 => /usr/lib/libexpat.so.1 (0xb7bc9000)
```

You can start the X server and JWM with the command below or create a script in `/usr/bin/startx` with the content:

```
Xvesa -ac -shadow -screen 1024x768x24 & exec jwm
```

On SliTaz

SliTaz uses the `~/.Xsession` file to start a graphical session. The **`startx`** command checks whether the file exists or it runs **`tazx`** to configure the X system. The user guide on X window is located in `/usr/share/doc/slitaz/user-guide/x-window.html` or is on the website:

We chose to use the Tango icons theme http://tango.freedesktop.org/, that isn't compiled. We only use the minimum: images in 16×16 format that we put in `/usr/share/icons`.

To test JWM with a cooking ISO:

```
# Xvesa -ac -shadow -screen 800x600x24 & exec jwm
```

8.8.9 jpeg-6b — JPEG Libraries

Libraries handling JPEG images, and some small utilities:

```
# wget http://www.ijg.org/files/jpegsrc.v6b.tar.gz
# tar xzf jpegsrc.v6b.tar.gz
# cd jpeg-6b
# ./configure --enable-shared --prefix=/usr \
  --mandir=/usr/share/man
# make
# strip .libs/*
# cp -a .libs/*.so* $fs/usr/lib
# cp .libs/{cjpeg,djpeg,jpegtran} $fs/usr/bin
```

8.8.10 tiff-3.8.2 — TIFF Libraries and Utilities

Libraries handling TIFF images and some small optional utilities:

```
# wget ftp://ftp.remotesensing.org/pub/libtiff/tiff-3.8.2.tar.gz
# tar xzf tiff-3.8.2.tar.gz
# cd tiff-3.8.2
# ./configure  --prefix=/usr --mandir=/usr/share/man
# make
# make DESTDIR=$PWD/_pkg install
# strip _pkg/usr/bin/*
# strip _pkg/usr/lib/*.so*
# cp -a _pkg/usr/lib/*.so* $fs/usr/lib
```

You can install the utilities you want.

8.8.11 links-2.1pre29 — Graphical and text mode web browser

Links (links.twibright.com[279]) is a web browser offering graphical and text modes. It is translated into multiple languages, including French:

[279] http://links.twibright.com/

```
# cd ..
# wget http://links.twibright.com/download/links-2.1pre28.tar.gz
# tar xzf links-2.1pre28.tar.gz
# cd links-2.1pre28
# ./configure --prefix=/usr --sysconfdir=/etc --mandir=/usr/share/man \
  --without-directfb --without-ssl --enable-graphics --enable-javascript
# make
# make DESTDIR=$PWD/_pkg install
# strip -v _pkg/usr/bin/*
# cp -v _pkg/usr/bin/* $fs/usr/bin
```

libs

```
libtiff.so.3 => /usr/lib/libtiff.so.3
libjpeg.so.62 => /usr/lib/libjpeg.so.62 (0xb7ede000)
libpng12.so.0 => /usr/lib/libpng12.so.0 (0xb7eba000)
libz.so.1 => /usr/lib/libz.so.1 (0xb7ea7000)
libX11.so.6 => /usr/lib/libX11.so.6 (0xb7dbb000)
libdl.so.2 => /lib/tls/libdl.so.2 (0xb7db7000)
libpcre.so.0 => /usr/lib/libpcre.so.0 (0xb7d96000)
libm.so.6 => /lib/tls/libm.so.6 (0xb7d70000)
libc.so.6 => /lib/tls/libc.so.6 (0xb7c3e000)
libXau.so.6 => /usr/lib/libXau.so.6 (0xb7c3b000)
libXdmcp.so.6 => /usr/lib/libXdmcp.so.6 (0xb7c36000)
/lib/ld-linux.so.2 (0xb7f5d000)
```

8.8.12 Generate the initramfs and an ISO image

To create a new ISO image, you can use **mktaziso** in ref:*cookbook slitaztools*. Or you can create a new initramfs image, copy it to /boot in the root of the CD-ROM (rootcd) and finally generate an ISO image with **genisoimage**:

```
# cd $fs
# find . -print | cpio -o -H newc | gzip -9 > ../rootfs.gz
# cd ..
# cp rootfs.gz rootcd/boot
# genisoimage -R -o slitaz-cooking.iso -b boot/isolinux/isolinux.bin \
  -c boot/isolinux/boot.cat -no-emul-boot -boot-load-size 4 \
  -V "SliTaz" -boot-info-table rootcd
```

Following chapter

The next chapter *GTK+ Libraries* (page 437) describes the installation of GTK libraries.

8.9 GTK+ Libraries

author domcox

Compilation and installation of GTK+ packages and libraries.

8.9.1 About

This chapter describes the installation and configuration of GTK libraries on SliTaz used by lots of free software. Note that you can simply compile and create a SliTaz package that you can install on demand with **tazpkg**.

The compilation of GTK applications can take quite some time and you must meet many dependencies. You will find a guide in English: gtk-building.html[280] on developer.gnome.org. This document sets out the need to compile things in order: Glib, Pango, ATK and GTK+, etc. Before commencing you must verify that the dependencies are properly installed on your host system. Glib, Pango, ATK and GTK+ form a group of packages and are distributed by the team of GTK developers.

Environmental variable (`$fs`)

If you do not specify any path to the rootfs directory, export the environmental variable:

```
# export fs=$PWD/rootfs
```

To check:

```
# echo $fs
```

8.9.2 cairo-1.2.6 — 2D graphics library

We begin with libcairo (http://www.cairographics.org/) used to compile pango:

```
# cd src
# wget http://cairographics.org/releases/cairo-1.2.6.tar.gz
# tar xzf cairo-1.2.6.tar.gz
# cd cairo-1.2.6
# ./configure --prefix=/usr --mandir=/usr/share/man \
  --with-html-dir=/usr/share/doc
# make
# make DESTDIR=$PWD/_pkg install
# strip -v _pkg/usr/lib/*.so*
```

Install in rootfs

```
# cp -av _pkg/usr/lib/*.so* $fs/usr/lib
```

8.9.3 glib-2.12.4 — C routines

```
# cd ..
# wget ftp://ftp.gtk.org/pub/glib/2.12/glib-2.12.4.tar.bz2
# tar xjf glib-2.12.4.tar.bz2
# cd glib-2.12.4
# ./configure --prefix=/usr --sysconfdir=/etc \
  --mandir=/usr/share/man --with-html-dir=/usr/share/doc
```

(continues on next page)

280 http://developer.gnome.org/gtk/2.22/gtk-building.html

(continued from previous page)

```
# make
# make DESTDIR=$PWD/_pkg install
# strip -v _pkg/usr/bin/*
# strip -v _pkg/usr/lib/*.so*
```

Install in rootfs

Option: the utilities glib-genmarshal and gobject-query need the `/lib/tls/librt.so.1`:

```
# cp -a _pkg/usr/lib/*.so* $fs/usr/lib
# cp -a _pkg/usr/share/locale/fr $fs/usr/share/locale
```

The binaries and options:

```
# cp -a _pkg/usr/bin/* $fs/usr/bin
```

8.9.4 pango-1.14.8 — Library for layout and rendering of text

```
# cd ..
# wget ftp://ftp.gtk.org/pub/pango/1.14/pango-1.14.8.tar.bz2
# tar xjf pango-1.14.8.tar.bz2
# cd pango-1.14.8
# ./configure --prefix=/usr --sysconfdir=/etc \
  --mandir=/usr/share/man --with-html-dir=/usr/share/doc
# make
# make DESTDIR=$PWD/_pkg install
# strip -v _pkg/usr/bin/*
# strip -v _pkg/usr/lib/*.so*
# strip -v _pkg/usr/lib/pango/1.5.0/modules/*
```

Install in rootfs

```
# cp -a _pkg/usr/bin/* $fs/usr/bin
# cp -a _pkg/usr/lib/*.so* $fs/usr/lib
# cp -a _pkg/usr/lib/pango $fs/usr/lib
# rm -rf $fs/usr/lib/pango/1.5.0/modules/*.la
# cp -a _pkg/etc $fs
```

Create `/etc/pango.modules` via chroot in the rootfs (pango-querymodules uses `librt.so.1`):

```
# chroot $fs /bin/ash
/# pango-querymodules > /etc/pango/pango.modules
# exit
```

8.9.5 atk-1.12.4 — Accessibility toolkit

```
# cd ..
# wget http://ftp.gnome.org/pub/gnome/sources/atk/1.12/atk-1.12.4.tar.bz2
# tar xjf atk-1.12.4.tar.bz2
# cd atk-1.12.4
# ./configure --prefix=/usr --mandir=/usr/share/man \
  --with-html-dir=/usr/share/doc
# make
# make DESTDIR=$PWD/_pkg install
# strip -v _pkg/usr/lib/*.so*
```

Install in rootfs

```
# cp -a _pkg/usr/lib/*.so* $fs/usr/lib
# cp -a _pkg/usr/share/locale/fr $fs/usr/share/locale
```

8.9.6 gtk+-2.8.20 — The GIMP Toolkit

```
# cd ..
# wget ftp://ftp.gtk.org/pub/gtk/v2.8/gtk+-2.8.20.tar.bz2
# tar xjf gtk+-2.8.20.tar.bz2
# cd gtk+-2.8.20
# ./configure --prefix=/usr --sysconfdir=/etc \
  --mandir=/usr/share/man --with-html-dir=/usr/share/doc
# make
# make DESTDIR=$PWD/_pkg install
# strip -v _pkg/usr/bin/*
# strip -v _pkg/usr/lib/*.so*
# strip -v --strip-unneeded \
  _pkg/usr/lib/gtk-2.0/2.4.0/*/*
```

Install in rootfs

```
# cp -a _pkg/usr/lib/*.so* $fs/usr/lib
# mkdir $fs/usr/lib/gtk-2.0
# cp -a _pkg/usr/lib/gtk-2.0/2.4.0 $fs/usr/lib/gtk-2.0
# rm -rf $fs/usr/lib/gtk-2.0/2.4.0/*/*.la
```

Locale and themes:

```
# cp -a _pkg/usr/share/locale/fr $fs/usr/share/locale
# cp -a _pkg/usr/share/themes $fs/usr/share
```

The applications:

```
# cp  _pkg/usr/bin/gtk-query-immodules-2.0 $fs/usr/bin
# cp  _pkg/usr/bin/gtk-update-icon-cache $fs/usr/bin
# cp  _pkg/usr/bin/gdk-pixbuf-csource $fs/usr/bin
# cp  _pkg/usr/bin/gdk-pixbuf-query-loaders $fs/usr/bin
...
```

For the gtk-demo application:

```
# cp -a _pkg/usr/bin/gtk-demo $fs/usr/bin
# cp -a _pkg/usr/share/gtk-2.0 $fs/usr/share
```

Create files `/etc/gtk-2.0/gtk.immodules` and `gdk-pixbuf.loaders` via a chroot in the rootfs:

```
# chroot $fs /bin/ash
/# mkdir /etc/gtk-2.0
/# gtk-query-immodules-2.0 > /etc/gtk-2.0/gtk.immodules
/# gdk-pixbuf-query-loaders > /etc/gtk-2.0/gdk-pixbuf.loaders
# exit
```

At this stage you can test GTK+ with the **gtk-demo** application by creating an ISO and using **qemu**. You can also compile a small GTK application such as **LeafPad** and test! The compiliation and installation of GTK+ applications distributed by default with SliTaz are described in the next chapter *GTK+ Applications* (page 441).

8.9.7 Generate the initramfs and an ISO image

To create a new ISO image, you can use **mktaziso** in *SliTaz Tools* (page 380). Or you can create a new initramfs image, copy it to `/boot` in the root of the CD-ROM (rootcd) and finally generate an ISO image with **genisoimage**:

```
# cd $fs
# find . -print | cpio -o -H newc | gzip -9 > ../rootfs.gz
# cd ..
# cp rootfs.gz rootcd/boot
# genisoimage -R -o slitaz-cooking.iso -b boot/isolinux/isolinux.bin \
  -c boot/isolinux/boot.cat -no-emul-boot -boot-load-size 4 \
  -V "SliTaz" -boot-info-table rootcd
```

Following chapter

After the libraries, the *GTK+ Applications* (page 441).

8.10 GTK+ Applications

author domcox

Compiliation and installation of applications using GTK+.

8.10.1 About

This chapter describes the commands for the compilation and installation of GTK+ applications distributed by default on the SliTaz LiveCD. The installation of GTK+ libraries are described in the *GTK+ Libraries* (page 437) chapter.

Environmental variable (`$fs`)

If you do not specify any path to the rootfs directory, export the environmental variable:

```
# export fs=$PWD/rootfs
```

To check:

```
# echo $fs
```

8.10.2 leafpad-0.8.10 — Simple text editor

Website: http://tarot.freeshell.org/leafpad/

```
# wget http://savannah.nongnu.org/download/leafpad/leafpad-0.8.10.tar.gz
# tar xzf leafpad-0.8.10.tar.gz
# cd leafpad-0.8.10
# ./configure --prefix=/usr
# make
# make DESTDIR=$PWD/_pkg install
# strip _pkg/usr/bin/*
```

Install in rootfs

```
# cp _pkg/usr/bin/* $fs/usr/bin
# cp _pkg/usr/share/pixmaps/leafpad.png $fs/usr/share/pixmaps
# cp -a _pkg/usr/share/locale/fr $fs/usr/share/locale
```

8.10.3 gitmail-0.4 — Ghost In The Mail, mail client

Ghost in the mail allows users to quickly and easily send mail via SMTP.

Website: http://gitmail.sourceforge.net/

```
# wget http://switch.dl.sourceforge.net/sourceforge/gitmail/gitmail-0.4.
↪tar.gz
# tar xzf gitmail-0.4.tar.gz
# cd GhostInTheMail-0.4
# ./configure --prefix=/usr
# make
# make DESTDIR=$PWD/_pkg \
  gitmaildocdir=/usr/share/doc/GhostInTheMail \
  install
# strip _pkg/usr/bin/*
```

Install in rootfs

```
# cp _pkg/usr/bin/* $fs/usr/bin
```

8.10.4 gqview-2.0.4 — Images Manager

Website: http://gqview.sourceforge.net/

```
# wget http://belnet.dl.sourceforge.net/sourceforge/gqview/gqview-2.0.4.
↪tar.gz
# tar xzf gqview-2.0.4.tar.gz
# cd gqview-2.0.4
# ./configure --prefix=/usr --mandir=/usr/share/man
# make
# make DESTDIR=$PWD/_pkg install
# strip _pkg/usr/bin/*
```

Install in rootfs

```
# cp _pkg/usr/bin/* $fs/usr/bin
# cp _pkg/usr/share/pixmaps/* $fs/usr/share/pixmaps
# cp -a _pkg/usr/share/locale/fr $fs/usr/share/locale
```

8.10.5 mtpaint-3.11 — Image creation and processing

Website: http://mtpaint.sourceforge.net/

```
# wget http://switch.dl.sourceforge.net/sourceforge/mtpaint/mtpaint-3.11.
↪tar.bz2
# tar xjf mtpaint-3.11.tar.bz2
# cd mtpaint-3.11
# ./configure --cpu=i486 --prefix=/usr intl
# make
# strip src/mtpaint
```

Install in rootfs

```
# cp src/mtpaint $fs/usr/bin
# cp po/fr.mo $fs/usr/share/locale/fr/LC_MESSAGES/mtpaint.mo
# cp src/icons1/icon.xpm $fs/usr/share/pixmaps/mtpaint.xpm
```

8.10.6 Transmission-0.72 — Lightweight BitTorrent client

Tranmission BitTorrent client is fast, lightweight and easy to use. The compiled package provides the command line client (transmissioncli) and a GTK+ client (transmission-gtk). We install the GTK+ client, the command line client is distributed as a separate SliTaz package (`*.tazpkg`).

Website: http://transmission.m0k.org/

```
# wget http://download.m0k.org/transmission/files/Transmission-0.72.tar.gz
# tar xzf Transmission-0.72.tar.gz
```

The archived version 0.72 is wrong:

```
# mv "Transmission .72" Transmission-0.72

# cd Transmission-0.72
# ./configure --prefix=/usr  --disable-openssl
# make
# strip gtk/transmission-gtk
# strip cli/transmissioncli
```

Install in rootfs

```
# cp gtk/transmission-gtk $fs/usr/bin
# cp gtk/transmission.png $fs/usr/share/pixmaps
# cp gtk/po/fr.mo $fs/usr/share/locale/fr/LC_MESSAGES/transmission-gtk.mo
```

8.10.7 emelfm2-0.3.5 — File Manager

The emelFM2 application is a file manager providing lots of useful functions, such as the mounting of devices, a text viewer, opening a terminal in the current directory and so on.

Website: http://emelfm2.net/

```
# cd ..
# wget http://emelfm2.net/rel/emelfm2-0.3.5.tar.gz
# tar xzf emelfm2-0.3.5.tar.gz
# cd emelfm2-0.3.5
# make PREFIX=/usr
# make i18n PREFIX=/usr
# make install PREFIX=$PWD/_pkg/usr
# make install_i18n PREFIX=$PWD/_pkg/usr
# strip -v _pkg/usr/bin/*
# strip -v _pkg/usr/lib/emelfm2/plugins/*
```

Install in rootfs

```
# cp _pkg/usr/bin/* $fs/usr/bin
# cp -a _pkg/usr/lib/* $fs/usr/lib
# cp -a _pkg/usr/share/pixmaps $fs/usr/share
# cp -a _pkg/usr/share/locale/fr $fs/usr/share/locale
```

8.10.8 geany-0.11 — Integrated Development Environment

Geany is a simple, fast and light IDE offering colored syntax, tabs, autocompletion, aids to scripts and much more.

Website: http://geany.uvena.de/

To compile and run geany on SliTaz, you must have the libstdc++ and libgcc1 libraries, both provided by gcc (we recompiled with gcc-4.1.1), but you can copy the libraries from the host system.

Note: The force is with you, if you activate it via the option `--enable-the-force`.

```
# wget http://mesh.dl.sourceforge.net/sourceforge/geany/geany-0.11.tar.gz
# tar xzf geany-0.11.tar.gz
# cd geany-0.11
# ./configure --prefix=/usr --mandir=/usr/share/man \
  --disable-vte --enable-the-force
# make
# make DESTDIR=$PWD/_pkg install
# strip -v _pkg/usr/bin/*
```

Install in rootfs

```
# cp _pkg/usr/bin/* $fs/usr/bin
# cp -a _pkg/usr/share/geany $fs/usr/share
# cp _pkg/usr/share/pixmaps/geany.png $fs/usr/share/pixmaps
# cp -a _pkg/usr/share/locale/fr $fs/usr/share/locale
```

8.10.9 gftp-2.0.18 — Fast and simple FTP client

The gFTP application is a fast and efficient FTP client with a GTK+ graphical interface. Note that we compile without support for a text interface and SSL support. Get, untar, configure, compile and install.

Website: http://www.gftp.org/

```
# wget http://www.gftp.org/gftp-2.0.18.tar.gz
# tar xzf gftp-2.0.18.tar.gz
# cd gftp-2.0.18
# ./configure --prefix=/usr --mandir=/usr/share/man \
  --disable-ssl --disable-textport \
  --build=i486-pc-linux-gnu --host=i486-pc-linux-gnu
# make
# make DESTDIR=$PWD/_pkg install
# strip _pkg/usr/bin/*
```

Install in rootfs

SliTaz provides only the GTK+ client on the CD. Note that “gftp” is just a small script that detects the environment (console or X) and launches the right interface:

```
# cp _pkg/usr/bin/gftp $fs/usr/bin
# cp _pkg/usr/bin/gftp-gtk $fs/usr/bin
# cp -a _pkg/usr/share/gftp $fs/usr/share
# cp -a _pkg/usr/share/pixmaps $fs/usr/share
# cp -a _pkg/usr/share/locale/fr $fs/usr/share/locale
```

To save a little space and avoid duplication, you can delete `COPYING` (17 KB) included in `/usr/share/gftp`. The GNU licence is already present in `/usr/share/licence`, if you want to create a symbolic link.

8.10.10 xpad-2.12 — Mini note taking application

The Xpad application can quickly take notes via various customizable (GTK+) windows.

Website: http://xpad.sourceforge.net/

```
# wget http://surfnet.dl.sourceforge.net/sourceforge/xpad/xpad-2.12.tar.bz2
# tar xjf xpad-2.12.tar.bz2
# cd xpad-2.12
# ./configure --prefix=/usr --mandir=/usr/share/man \
  --build=i486-pc-linux-gnu --host=i486-pc-linux-gnu
# make
# make DESTDIR=$PWD/_pkg install
# strip _pkg/usr/bin/*
```

Install in rootfs

```
# cp _pkg/usr/bin/xpad $fs/usr/bin
# cp -a _pkg/usr/share/pixmaps $fs/usr/share
# cp -a _pkg/usr/share/locale/fr $fs/usr/share/locale
```

8.11 How-to Xorg — Modular graphical server

author domcox

Note SliTaz uses the Xvesa server provided by XFree86 and Xorg libraries, this page describes the compilation of Xorg libraries used by SliTaz. This document is primarily aimed at developers and contributors to the project, but it may be useful to all those seeking to rebuild Xorg and Xlib libraries from source generating a minimum of dependencies.

8.11.1 Build Xorg automatically with Tazwok

On SliTaz, if you have Tazwok installed, you can rebuild Xorg with a few commands. The wok contains a package called "xorg" and another named "xorg-dev", these can compile/cook all the packages used by Xorg on SliTaz. To compile, you must have most of the development packages installed; if this is not the case:

```
# tazpkg get-install slitaz-dev-pkgs
```

Then you can start to cook (if everything is ready, wok and development packages, etc), starting with the protos' (xproto, etc):

```
# tazwok cook xorg-dev-proto
# tazwok cook xorg
# tazwok cook xorg-dev
```

8.11.2 Download Xorg (7.2) using wget

Xorg is distributed in the form of modules, which is handy because you can only install what you want, but it takes a lot of downloads. To help, we have created a small script that downloads the minimum required for SliTaz; you can find the script "getXorg.sh" in *SliTaz Tools* (page 380) (1.1). This script is no longer updated, developers use the *Wok & Tools* (page 374). To use the script, it must be placed in the directory where you want to download Xorg:

```
# cd ..
# mkdir Xorg && cd Xorg
# cp slitaz-tools-1.1/utils/getXorg-7.2.sh .
# ./getXorg-7.2.sh
```

8.11.3 Compile Xorg by hand

Compiling Xorg can take a long time, there are many packages. To commence you need to compile the downloaded proto packages. You can use the command **`make DESTDIR=$PWD/_pkg install`** to install the package in a given directory. Example:

```
# cd proto
# tar xzf xproto-X11R7.2-7.0.10.tar.gz
# cd xproto-X11R7.2-7.0.10
# ./configure --prefix=/usr --sysconfdir=/etc \
  --mandir=/usr/share/man --localstatedir=/var \
  --build=i486-pc-linux-gnu --host=i486-pc-linux-gnu
 # make
 # make install
```

Compile libraries by taking again the options used by proto. Example using the package to compile xtrans, remember to run “ldconfig” if you install the package on the development machine:

```
# cd .. && cd lib
# tar xzf xtrans-X11R7.2-1.0.3.tar.gz
# cd xtrans-X11R7.2-1.0.3
# ./configure --prefix=/usr --sysconfdir=/etc \
  --mandir=/usr/share/man --localstatedir=/var \
  --build=i486-pc-linux-gnu --host=i486-pc-linux-gnu
# make
# make install
# ldconfig
```

Once all the packaged libraries are compiled, you can begin to compile X applications such as the graphical terminal Xterm. Note: SliTaz uses the RGB package containing the `/usr/share/X11/rgt.text` file for defining colors. Example using the “xsetroot” application that permits you to change the background color of the screen (modify $VERSION for the version that you want downloaded):

```
# cd .. && cd app
# tar xzf xsetroot-$VERSION.tar.gz
# cd xsetroot-$VERSION
# ./configure --prefix=/usr --sysconfdir=/etc \\
  --mandir=/usr/share/man --localstatedir=/var \\
  --build=i486-pc-linux-gnu --host=i486-pc-linux-gnu
# make && make install
```

CHAPTER 9

Manuals

9.1 Tazinst Manual

9.1.1 NAME

Tazinst — Tiny autonomous zone installer manager

9.1.2 SYNTAX

```
tazinst [command] <setting> <value> <file>
```

9.1.3 DESCRIPTION

Tazinst is a lightweight SliTaz HDD installer. It installs SliTaz to a hard drive from any local media such as a Live-CD, a LiveUSB key, an ISO image located on one of your disks, or from the web by automatically downloading a SliTaz image.

Tazinst can format the target partition to ext2, ext3 or ext4. The home partition can be installed on another partition and if need be, formatted before installation into any one of the available formats. **Tazinst** may upon request install a bootloader on the target disk. A dual-boot with an existing Windows™ partition is possible — finding the Windows™ partition can either be done automatically or manually specified.

Tazinst can also update SliTaz installed on a hard disk partition which is handy in case of version changes. In this case, SliTaz is updated, any data in `/home` is preserved and additional packages are reinstalled on to the new version.

Tazinst was created independently for the needs of the SliTaz GNU/Linux mini distribution.

Tazinst is written from scratch in shell script and is compatible with Busybox Ash and Bash. **Tazinst** is licensed under the GNU Free GPL v3.

9.1.3.1 Known limitations

`Tazinst` doesn't allow SliTaz to boot on (U)EFI systems (mostly Windows™ 8 systems), except in BIOS compatibility mode.

9.1.4 SETTINGS

`Tazinst` installer is able to perform an installation automatically based on a few settings.

mode Installation mode that will be performed by **`tazinst`**. Type **`tazinst help mode`** in order to have a list of supported modes.

media The media containing the SliTaz source files, either **cdrom** (SliTaz LiveCD), **usb** (SliTaz LiveUSB), **iso** (ISO image of SliTaz), or **web** (ISO image on the Web).

source The name of the source file containing SliTaz. It depends on the type of **media**:

cdrom:	*unused*
usb:	name of the partition on the host USB device. Type **`tazinst list usb`** to list USB partitions.
iso:	name of the ISO file, example: `~/slitaz-rolling.iso`. Type **`tazinst list iso`** to list ISO files on your disks.
web:	name of the image on the web, example: `stable cooking rolling base core gtkonly justx`, for a full list type **`tazinst list web`**, or enter the full URL of the image, example: `http://mirror.slitaz.org/iso/cooking/slitaz-cooking.iso`.

root_uuid The name of the target partition SliTaz will install to. Type **`tazinst list uuid`** to list partitions on your disks.

root_format Optional. If this setting is used, the target partition will be formatted in the file system specified, otherwise the partition will be cleaned and `/home` will be preserved. Type **`tazinst help format`** to get the list of all supported filesystems, and **`tazinst list root_format`** to see filesystems already installed on your system.

home_uuid Optional. A separate home partition may be created if needed. This setting indicates if need be, the name of the partition to receive the `/home` directory.

home_format Optional. If this setting is used, a separate `/home` partition will be created, this partition will be formatted in the file system specified.

hostname Optional. Hostname of the system, `slitaz` by default.

root_pwd Optional. Superuser [root] password, `root`, by default.

user_login Optional. First user name, `tux` by default.

user_pwd Optional. First user password, `tux` by default.

bootloader Optional. Install a bootloader. Usually you should set it to `auto` unless you want to use an already installed bootloader on your system, install a bootloader by hand yourself or install a specific bootloader. In this case type **`tazinst help bootloader`** to list supported bootloaders.

winboot Optional. If a bootloader is installed, this setting indicates the partition containing Windows™ to implement a dual-boot. It can also be set to `auto`, in this case the dual-boot will be on the first Windows™ partition. Type **`tazinst list winboot`** to see values **`tazinst`** automatically detects.

9.1.5 COMMANDS

new Generates a new self-documenting install file containing settings which, when set up as required by the user, will allow **`tazinst`** to execute an unattended installation. The default file is `./tazinst.rc`, but an optional file name may be given as a parameter.

Examples:

```
tazinst new
tazinst new /var/lib/tazinst.conf
```

set Assign a new value to a given setting.

Examples:

```
tazinst set mode install
tazinst set mode install /var/lib/tazinst.conf
```

unset Unset, clears a setting.

Examples:

```
tazinst unset mode
tazinst unset mode /var/lib/tazinst.conf
```

get Get the value of a setting. Without a parameter, gets the values of all settings.

Examples:

```
tazinst get
tazinst get mode
tazinst get mode /var/lib/tazinst.conf
```

check Check a setting for errors. Without a parameter, checks all settings.

Examples:

```
tazinst check
tazinst check mode
tazinst check mode /var/lib/tazinst.conf
```

list List the system resources. Resources are:

mode	Available modes of install
media	Available media to install from. Example: cdrom is not listed on systems with no cdrom drive
usb	Partitions of USB disks
iso	ISO images located on local drives, in `/root`, and in all user's home and first subdirectory
web	Predefined names of ISO images to download automatically from the Internet
format	Installed filesystems
bootloader	Available bootloaders
partition_table	Partition table schemes of local disks
winboot	Bootable Windows™ partitions

Examples:

```
tazinst list
tazinst list media
```

execute Performs a SliTaz install on a HDD based on data in the install file. If you selected to format your HDD, all data will be lost. If you do not, all data except for any existing `/home` directory will be removed, (the home directory will be kept as is).

Examples:

```
tazinst execute
tazinst execute /var/lib/tazinst.conf
```

clean Remove installation and log files.

Examples:

```
tazinst clean
tazinst clean /var/lib/tazinst.conf
```

log Display the last log file contents and exit.

Example:

```
tazinst log
```

version Print the version information and exit.

Example:

```
tazinst version
```

usage Print a short help and exit.

Example:

```
tazinst usage
```

help Print a short help for a given setting and exit. Without an argument, print a short help for all settings.

Example:

```
tazinst help mode
```

9.1.6 EXAMPLES

9.1.6.1 Install

How to Install SliTaz on a partition of your hard disk drive. The root partition is not formatted, all data except for any existing `/home` directory will be removed, (the `home` directory will be kept as is).

1. Create an install file:

```
# tazinst new
```

2. Set the mode as install:

```
# tazinst set mode install
```

3. Use a CD-ROM as source:

```
# tazinst set media cdrom
```

4. Select the partition to install SliTaz on:

```
# tazinst set root_uuid /dev/hda1
```

5. Install a bootloader:

```
# tazinst set bootloader auto
```

6. Execute an installation:

```
# tazinst execute
```

9.1.6.2 Complex Install

How to Install SliTaz on your hard disk drive with a separate home partition and a Windows™ dual-boot. The `/home` and root partitions are both formatted, (all existing data will be lost).

1. Create an install file:

```
# tazinst new
```

2. Set the mode as install:

```
# tazinst set mode install
```

3. Use a Live USB as source:

```
# tazinst set media usb
```

4. Select a partition on the Live USB:

```
# tazinst set source /dev/sda1
```

5. Select the partition to install SliTaz on:

```
# tazinst set root_uuid /dev/hda1
```

6. Format / as ext4:

```
# tazinst set root_format ext4
```

7. Use a separate `/home` partition:

```
# tazinst set home_uuid /dev/hda2
```

8. Format `/home` as ext2:

```
# tazinst set home_format ext2
```

9. Install a bootloader:

```
# tazinst set bootloader auto
```

10. Set up a Windows™ dual-boot:

```
# tazinst set winboot auto
```

11. Execute an installation:

```
# tazinst execute
```

12. Remove any traces behind:

```
# tazinst clean
```

9.1.6.3 Upgrade

How to upgrade an already installed SliTaz system on your hard disk drive. Your `/home` `/etc` `/var/www` directories will be kept, all other directories will be removed. Any additional packages added to your old SliTaz system will be updated as long you have an active internet connection.

1. Create an install file:

```
# tazinst new
```

2. Set the mode as upgrade:

```
# tazinst set mode upgrade
```

3. Use web as source:

```
# tazinst set media web
```

4. Select the stable image:

```
# tazinst set source stable
```

5. Select the partition containing SliTaz to upgrade:

```
# tazinst set root_uuid /dev/hda1
```

6. Install a bootloader:

```
# tazinst set bootloader auto
```

7. Execute an installation:

```
# tazinst execute
```

9.1.6.4 Tips

1. Not all settings are used depending on the mode of install. List all settings to see which you need to edit:

```
# tazinst get
```

2. Check your settings before executing install:

```
# tazinst check
```

9.1.7 FILES

9.1.7.1 INSTALL FILE

Settings are saved in the install file, then used by **tazinst** to execute an unattended installation. The default file is `./tazinst.rc`, but an optional file name may be given as a parameter. The install file is self-documented. The `clean` command erases this file.

9.1.7.2 SYSTEM FILE

The `/etc/slitaz/tazinst.conf` configuration file allows you to change the default settings of **tazinst** in case you want to use default custom values.

All settings are customisable, if a particular setting is missing, just add the name of the setting in caps and enter the new value.

Example:

If you intend to always install SliTaz from the same ISO on the web, you just have to modify or add the following values:

```
MEDIA="web"
SOURCE="stable"
```

9.1.7.3 LOG FILE

The file `/var/log/tazinst.log` contains a log of the install process. The `clean` command erases this file.

9.1.8 MAINTAINERS

- Christophe Lincoln <pankso@slitaz.org>
- Dominique Corbex <domcox@slitaz.org>

9.2 TazPkg Manual

9.2.1 Name

TazPkg — Tiny autonomous zone package manager.

9.2.2 Syntax

```
tazpkg [command] [options...]
```

9.2.3 Description

TazPkg is a lightweight package manager to install, list, download, update or remove precompiled packages on a GNU/Linux system. **TazPkg** offers commands for searching and creating packages and was created independently for the project. The format of the packages using the `*.tazpkg` extension is a cpio archive containing a filesystem compressed with lzma, a receipt and an optional description. **TazPkg** also manages dependencies based on package receipts. Each receipt contains all the information about a package and can also include pre- and post-installation functions. The same receipt is used by *Cookutils* (page 480) to compile sources and generate a `.tazpkg` package.

TazPkg is entirely built from scratch using Shell script, compatible with Bash; it runs under Ash — part of the Busybox project. **TazPkg** is distributed under the free GNU license GPL V3.

9.2.4 Environment

TazPkg uses some environment variables:

LANGUAGE defines the language of output and user confirmations.

Note, `LANG` and `LC_ALL` environment variables also affect output language

LC_TIME defines the date format in the *activity* (page 461) command

root if defined, points to the root of a file system where **TazPkg** should work. Note, a value defined using `--root=` option has precedence over this environment variable

9.2.5 Files

9.2.5.1 Configuration files

- `/etc/slitaz/slitaz.conf`
- `/etc/slitaz/tazpkg.conf`

9.2.5.2 Package database files

Default placement of the package database is `/var/lib/tazpkg`.

`ID` (deprecated): identifier of the current SliTaz repository state.

The value changes when new or updated packages appear in the repository.

`IDs`: identifier of the current SliTaz repository state and the UNIX time stamp.

The ID value changes when new or updated packages appear in the repository. Time stamp allows you to track how long a change is made in the repository and to track the freshness of repository mirrors. (To convert UNIX time stamp to the date: **`date -d@timestamp`**)

`mirror`: URL of the current repository mirror in use.

URL points to the remote folder containing packages and database files.

`mirrors`: list of URLs of available repository mirrors.

Note the difference between URLs from these two files; you should append URL from this file by `packages/`*`cooking`*`/` (for cooking-based SliTaz version).

`packages.list` (deprecated): list of package names with version numbers available in the repository.

`packages.desc` (deprecated): list contained package name, version, short description, category and upstream URL.

`packages.txt` (deprecated): list containing package name, version, short description and two package sizes (first — traffic to download package, second — HDD size for installed package).

`packages.md5` (deprecated): list containing MD5 checksum with package file name.

`packages.info`: list was built to replace and extend above lists.

For every package available in the repository it contains: package name, version, category, short description, upstream URL, tags, package sizes, depends and MD5 checksum. Development continues, and the list can be extended by other fields, if necessary.

`packages.equiv`: list of equivalent packages available in the repository.

Format of the list item: `package = rule rule...`

Format of the rule: `alternative:newname` — if `alternative` is installed then `newname` will be installed instead of the `package`.

In the second format of the rule `alternative:` part is omitted — in this case `newname` *can be* installed instead of the `package` (in the user choice).

`packages.diff`: list created when you run *recharge* (page 469) command.

List contains package names with versions of the new packages available in the repository with the last recharge.

`packages.up`: list created when you run *upgrade* (page 469) command.

List contains package names you can upgrade with the last recharge.

`descriptions.txt`: list of the "long" descriptions of the all packages available on the repository.

`packages-desc.`*`language`*: optional list of the packages translated short descriptions (see the package tazpkg-desc-ru).

`descriptions.`*`language`*`.txt`: optional list of the packages translated "long" descriptions (see the package tazpkg-desc-ru).

`packages.icons`: optional list of the packages icons for TazPanel.

`blocked-packages.list`: optional list of the packages blocked for update.

`extra.list`: list of the extra packages (non-free packages; free packages but not compiled from sources). List contains package name, short description, upstream URL, category, version, license.

`files.list.lzma`: very large compressed list which contains all the files of all the packages available in the repository (nearly 5,000 packages and nearly 0.7 million files now).

`files-list.md5`: MD5 checksum of the `files.list.lzma`.

`installed.info`: list of the installed packages with the exact format of the `packages.info` list.

Idea is just to copy list item from `packages.info` to the `installed.info` during package installation.

`installed.md5` (deprecated): list containing MD5 checksum with file names of all installed packages.

`priority`: optional list of the repositories priority.

One repository name per line. Undigest repositories are called by their names and main repository by "main". If priority list absent, then default priority is: main repository and all existing undigest repositories in alphabetical order.

All the deprecated files will be deleted after we verify that the programs do not use them (and modify the programs as necessary).

In addition to the common package database files there is also individual folders for all installed packages placed by default in the `/var/lib/tazpkg/installed/`*`package`*. Every folder here may contain the following files:

- `receipt` (mandatory) — the package recipe
- `files.list` (mandatory) — list of package files
- `md5sum` (mandatory) — checksums of package files (other checksum files can be specified in the settings: `cksum` (CRC32), `md5sum` (MD5), `sha1sum` (SHA1), `sha256sum` (SHA256), `sha512sum` (SHA512), `sha3sum` (SHA3-512))
- `description.txt` (optional) — "long" description
- `modifiers` (optional) — list of packages that have replaced some of the files of this package
- `volatile.cpio.gz` (optional) — archive of "official" configuration files

9.2.5.3 Cache

Default placement of the packages cache is `/var/cache/tazpkg` with sub-folders for the different repositories.

It is exactly `/var/cache/tazpkg/`*`cooking`*`/packages` for the cooking-based SliTaz version.

9.2.5.4 Misc files

File with default placement `/var/log/slitaz/tazpkg.log` stores the TazPkg activity log.

Tip: Log stores five types of actions: installing, uninstalling, blocking, unblocking, reconfiguring packages.

Shared MIME information which allows to “guess” SliTaz package files, package receipts and SliTaz flavor files.

Tip: File placed here: `/usr/share/mime/packages/tazpkg.xml`.

Plug-in for the TazPanel (SliTaz administration and configuration panel) `/var/www/tazpanel/pkgs.cgi` allows you to manage SliTaz packages in the web application[281].

TazPkg documentation is placed in the `/usr/share/doc/tazpkg` folder.

9.2.6 Commands

The first **`tazpkg`** parameter is a command followed by other mandatory and optional parameters as will be described hereinafter. Options begin with double dashes, you can arrange them in any order and in any place, even before the command. Unknown and inappropriate options are ignored. The following commands are equivalent:

```
$ tazpkg info nano --root=/mnt/sda6
$ tazpkg --root=/mnt/sda6 info nano
$ tazpkg info --root=/mnt/sda6 nano --color
```

You can add global option `--root=...` to any **`TazPkg`** command. This option allows you to work with other SliTaz installations and can point to the root of a mounted file system from another SliTaz installation.

Tip: By the way, using this option allows to install SliTaz to the other file system “from scratch” and upgrade SliTaz packages remotely.

- Service commands
  - *usage* (page 461): print short usage
  - *help* (page 461): show help on the TazPkg commands
  - *activity* (page 461): show TazPkg activity log

[281] http://127.0.0.1:82/pkgs.cgi

  - *clean-cache* (page 462): clean all packages downloaded in cache directory
  - *list-cache* (page 462): list all packages downloaded in cache directory
  - *shell* (page 462): run interactive TazPkg shell
- Working with lists
  - *list* (page 462): list installed packages on the system
  - *list-mirror* (page 462): list all available packages on the mirror
  - *list-config* (page 463): list the configuration files
- Search
  - *search* (page 463): search for a package by pattern or name
  - *search-pkgname* (page 463): search on mirror for package having a particular file
  - *search-file* (page 463): search for file in all installed packages files
- Installing and removing packages
  - *get* (page 463): download a package into the current directory
  - *install* (page 464): install a local package
  - *get-install* (page 464): download and install a package from the mirror
  - *get-list* (page 465): download a list of packages from the mirror
  - *install-list* (page 465): install all packages from a list of packages
  - *get-install-list* (page 465): download and install a list of packages from the mirror
  - *remove* (page 465): remove the specified package and all installed files
  - *reconfigure* (page 465): replay post install script from package
  - *link* (page 465): link a package from another SliTaz installation
  - *set-release* (page 465): change release and update packages
  - *add-flavor* (page 465): install the flavor list of packages
  - *install-flavor* (page 465): install the flavor list of packages and remove other ones
- Working with packages
  - *info* (page 466): print information about a package
  - *desc* (page 466): print description of a package
  - *list-files* (page 466): list the files installed with a package
  - *block, unblock* (page 466): block an installed package version or unblock it for upgrade
  - *check* (page 467): verify consistency of installed packages
  - *bugs* (page 467): show known bugs in packages
  - *depends* (page 467): display dependencies tree
  - *rdepends* (page 467): display reverse dependencies tree
  - *extract* (page 467): extract a (`*.tazpkg`) package into a directory

  - *pack* (page 467): pack an unpacked or prepared package tree
  - *repack* (page 468): create a package archive from an installed package
  - *repack-config* (page 468): create a package archive with configuration files
  - *recompress* (page 468): rebuild a package with a better compression ratio
  - *convert* (page 468): convert alien package to tazpkg
  - *list-suggested* (page 469): print list of suggested packages
- Working with repositories
  - *recharge* (page 469): recharge your packages database from the mirror
  - *upgrade* (page 469): check packages, list and install latest upgrades
  - *setup-mirror* (page 470): change the mirror URL configuration
  - *setup-undigest* (page 470): update an undigest mirror
  - *list-undigest* (page 470): list undigest mirrors
  - *add-undigest* (page 470): add an undigest mirror
  - *remove-undigest* (page 470): remove an undigest mirror
  - *mkdb* (page 470): make a TazPkg database for a folder with `*.tazpkg` packages

9.2.6.1 Service commands

9.2.6.1.1 usage

Show the full list of the TazPkg commands with a brief description.

```
$ tazpkg usage
```

9.2.6.1.2 help

Display help for the selected command (`help` or `-h`). You can enter a short name of the command, the full name or a part of the full name. You can ignore hyphens at the beginning of the short name. If the requested part of the full name match the several commands, you will be asked to clarify the request.

```
$ tazpkg help -gi
$ tazpkg -h us
```

9.2.6.1.3 activity

Display TazPkg activity log (`activity` or `log` or `-a`). Optional parameter `--nb=` lets you set number of displayed lines.

```
$ tazpkg activity
$ tazpkg -a --nb=20
```

9.2.6.1.4 clean-cache

Remove `*.tazpkg` packages downloaded to the cache (`clean-cache` or `-cc`). During installation, TazPkg keeps a copy of packages downloaded from the Web. This is done to save bandwidth in case of reinstallation, but you may want to free up space on the hard drive or re-download the packages.

```
# tazpkg clean-cache
# tazpkg -cc
```

9.2.6.1.5 list-cache

List `*.tazpkg` packages downloaded to the cache. Displays a list of file names and their sizes, as well as the total amount and size.

```
# tazpkg list-cache
```

9.2.6.1.6 shell

Run interactive TazPkg shell. Here you can enter all the TazPkg commands listed above.

```
$ tazpkg shell
# tazpkg shell
```

9.2.6.2 Working with lists

9.2.6.2.1 list

List packages installed on the system (`list` or `-l`). This command displays a column list of all installed packages. It also allows you to list the categories (`c` or `cat` or `categories`), packages based on category and packages placed on hold (`b` or `blocked`). You can also use the *search* (page 463) command for a list based on a term or package name.

```
$ tazpkg list
$ tazpkg list cat
$ tazpkg list games
$ tazpkg list blocked
```

9.2.6.2.2 list-mirror

List packages available on the mirror (`list-mirror` or `-lm`). This command will display the packages list recharged from the mirror. If it doesn't exist, you will be asked to launch `tazpkg` *recharge* (page 469) as administrator (root) for a list of available packages. The `--diff` option is used to display the differences between the last and current list of packages.

```
$ tazpkg list-mirror
$ tazpkg -lm --diff
```

9.2.6.2.3 list-config

Lists the system configuration files. The `--box` option displays in table form. You can specify package name to display configuration files only for this package.

```
$ tazpkg list-config
$ tazpkg list-config --box
$ tazpkg list-config slim
$ tazpkg list-config slim --box
```

9.2.6.3 Search

9.2.6.3.1 search

Search for packages by owner or package name (`search` or `-s`). This command will search for the term wanted in the installed packages (`-i` or `--installed`) and the list of available packages on the mirror (`-l` or `--list`).

To obtain the latest list of installable packages on the mirror, just run `tazpkg recharge` before conducting a search.

```
$ tazpkg search gcc
$ tazpkg search mt -i
$ tazpkg search bit -l
```

9.2.6.3.2 search-pkgname

Search for a file on mirror and output only the packages names (`search-pkgname` or `-sp`).

```
$ tazpkg search-pkgname libnss
$ tazpkg -sp /usr/share/fonts
```

9.2.6.3.3 search-file

Search for a file among the files installed by the packages (`search-file` or `-sf`). This command is very useful to find the full path to a file and determine if a file is present on the system. Option `--mirror` allows to search for a file among all the files available on the mirror.

```
$ tazpkg search-file libnss
$ tazpkg -sf /usr/share/fonts --mirror
```

9.2.6.4 Installing and removing packages

9.2.6.4.1 get

Get a package from the mirror (`get` or `-g`). The downloaded package is stored in the current directory. You will get regular package, or get-package, or extra-package (in that order). You can specify `--extra` option to get an extra-package only.

You can specify multiple packages on the command line or give tazpkg a list of the packages you want to download or use *get-list* (page 465) command.

```
# tazpkg get grub
# tazpkg get nano --root=/mnt/sda6
# tazpkg get palemoon --extra
# tazpkg -g nano mc
# tazpkg -g --list=/tmp/office
```

9.2.6.4.2 install

This command allows the installation of a local package with the `.tazpkg` extension (`install` or `-i`).

Option `--forced` allows you to update an already installed package. Option `--newconf` allows you to rewrite all user configuration files using the new files from a package. Option `--nodeps` allows you to install only a specified package without its dependencies.

When TazPkg installs package dependencies, it prefers local packages (i.e. dependent packages located in the same folder as installed packages) over mirrored/cached packages with the `--local` option. It is useful when you want to install a lot of already downloaded packages and their dependencies without the need of a network connection.

You can specify multiple packages on the command line or give tazpkg a list of the packages you want to install or use *install-list* (page 465) command.

See *get-install* (page 464) to install a package from the internet.

```
# tazpkg install package-1.0.tazpkg
# tazpkg -i path/to/package-1.0.tazpkg --forced
# tazpkg -i path/to/package-1.0.tazpkg --root=/mnt/rootfs
# tazpkg -i nano-2.4.0.tazpkg mc-4.8.14.tazpkg
# tazpkg -i --list=/tmp/development
# cd /home/boot/packages; tazpkg -i nano-2.4.0.tazpkg --local
```

9.2.6.4.3 get-install

Get and install a package from a mirror on the internet (`get-install` or `-gi`). Command begins by checking whether the package exists on the mirror and if it has been already downloaded.

Option `--forced` allows you to update an already installed package. Option `--newconf` allows you to rewrite all user configuration files using the new files from a package. Option `--nodeps` allows you to install only a specified package without its dependencies.

You can specify multiple packages on the command line or give tazpkg a list of the packages you want to get and install or use *get-install-list* (page 465) command.

For a list of packages on the mirror, you must use the *list-mirror* (page 462) command.

```
# tazpkg get-install grub
# tazpkg -gi grub --forced
# tazpkg -gi nano --root=/mnt/sda6
# tazpkg -gi nano mc
# tazpkg -gi --list=/tmp/multimedia
```

9.2.6.4.4 get-list, install-list, get-install-list

Get and/or install a set of packages listed in a file. This command allows you to work with the several packages with a single command.

All options are the same as for respective simple commands: *get* (page 463), *install* (page 464) and *get-install* (page 464).

```
# tazpkg install-list my-packages.list
# tazpkg get-install-list my-packages.list --forced
```

9.2.6.4.5 remove

Remove a package (`remove` or `-r`). You will be asked for confirmation (y/N) of removing the package, as well as for removing packages depending on this package, and for reinstalling packages modified by this package. This command will delete all files installed with the package.

Option `--auto` removes and reinstalls packages without your confirmation.

```
# tazpkg remove bc
# tazpkg -r gtk+-3 --root=/mnt/sda6
# tazpkg -r nano --auto
```

9.2.6.4.6 reconfigure

Replays the post-install script from the package.

```
# tazpkg reconfigure gcc
# tazpkg reconfigure gcc --root=/mnt/sda6
```

9.2.6.4.7 link

This command allows the installation of a package from another media device. The set up is done through symbolic links and consumes very little memory. It is generally used within the system RAM to install add-ons from a USB key.

```
# tazpkg link openoffice /media/usbdisk
```

9.2.6.4.8 set-release

This command changes the current version and upgrades all of the packages to the latest release.

```
# tazpkg set-release cooking
```

9.2.6.4.9 add-flavor, install-flavor

Install a set of packages from a flavor. In addition, `install-flavor` purges other installed packages.

```
# tazpkg add-flavor gtkonly
# tazpkg install-flavor justx
```

9.2.6.5 Working with packages

9.2.6.5.1 info

Show all the available information related to your package. You can specify the name of the installed package or a package that is not yet installed, but available in the repository. You can also get information about a local file package by entering an absolute or relative path to the file `.tazpkg`. The information contained in the TazPkg database and in the package recipe — its version, category, maintainer, Web site and all the dependencies (see also *Cookutils Documentation* (page 480) for more information on recipes).

```
$ tazpkg info busybox
$ tazpkg info minitube
$ tazpkg info packages/comix-4.0.4.tazpkg
```

9.2.6.5.2 desc

Description of the package (`desc` or `-d`).

```
$ tazpkg desc busybox
```

9.2.6.5.3 list-files

List all files installed with a package (`list-files` or `-lf`). This command will simply read and display the `files.list` of each package which is automatically generated when the package is created and is also used to remove files when uninstalling a package.

```
$ tazpkg list-files bc
```

9.2.6.5.4 block, unblock, chblock

The `block` (or `-b`) and `unblock` (or `-u`) commands permit you to block installed package versions so that they are not maintained by an *upgrade* (page 469). Command `chblock` changes the blocking state of the package. The list of packages on hold are contained in the `/var/lib/tazpkg/blocked-packages.list`. This file can also be edited by hand.

```
# tazpkg block grub
# tazpkg unblock grub
# tazpkg chblock grub
```

9.2.6.5.5 check

Check dependencies on installed packages and determine whether all the files needed for the repacking of packages are present. You can specify package name to check or check all installed packages. Option `--full` makes few more checks and need more time.

```
$ tazpkg check
$ tazpkg check --full
$ tazpkg check nano
$ tazpkg check sakura --full
```

9.2.6.5.6 bugs

Generates a list of known bugs in the packages. You can specify a single package to show bugs.

```
$ tazpkg bugs
$ tazpkg bugs nano
```

9.2.6.5.7 depends, rdepends

Displays a dependency tree or reverse dependency tree for a package.

For `depends` command: option `--mark` marks installed packages with plus sign and not installed with a minus/dash. Option `--total` calculates the number of displayed packages and their size. Also, with both options at a time you'll get the number and size of packages to be installed.

For `rdepends` command: *without* the `--all` option you'll get the list of only installed reverse dependency packages, *with* this option — a list of all available packages. Option `--mark` marks packages as in the `depends` command.

```
$ tazpkg depends mpd
$ tazpkg rdepends mpd
```

9.2.6.5.8 extract

Extract a package into a directory (`extract` or `-e`). If you do not specify the destination directory, the package will be extracted in the current directory using the name *`package-version`*.

```
$ tazpkg extract package.tazpkg
$ tazpkg extract package.tazpkg target/dir
```

9.2.6.5.9 pack

Create a package from a directory prepared in advance or from an unpacked package. It can also manually create a `.tazpkg` package (see the *Cookutils* (page 480) documentation for the automatic creation of packages).

```
# tazpkg pack package-version
```

9.2.6.5.10 repack

Recreate a package from the files on a system where it was previously installed.

```
# tazpkg repack <em>package</em>
```

9.2.6.5.11 repack-config

Recreate a package of the system configuration files (see *list-config* (page 463)). It is enough to install the package to find the current configuration.

```
# tazpkg repack-config
```

9.2.6.5.12 recompress

Recompress `.tazpkg` cpio archive with lzma.

```
# tazpkg recompress package.tazpkg
```

9.2.6.5.13 convert

Converts an “alien” package into a SliTaz package (`.tazpkg`) (`convert` or `-c`).

Supported packages formats:

- Debian packages[282] (`*.deb`, `*.udeb`)
- RPM packages[283] (`*.rpm`)
- Slax packages[284] (`*.sb`)
- Puppy packages[285] (`*.sfs`, `*.pet`)
- Slackware packages[286] (`*.tgz`)
- NuTyX packages[287] (`*.cards.tar.xz`)
- Arch Linux[288] / Alpine Linux packages[289] (`*.apk`, `*.pkg.tar.gz`, `*.pkg.tar.xz`)
- OpenWrt packages[290] (`*.ipk`, `*.opk`)

[282] https://packages.debian.org/search
[283] http://rpmfind.net/linux/rpm2html/search.php
[284] http://www.slax.org/
[285] http://puppylinux.org/
[286] http://www.slackware.com/packages/
[287] http://www.nutyx.org/
[288] https://www.archlinux.org/packages/
[289] http://pkgs.alpinelinux.org/packages
[290] http://wiki.openwrt.org/doc/packages

- 0Linux packages[291] (`*.spack`)
- paldo packages[292] (`*.tar.bz2`)
- Void packages[293] (`*.xbps`)
- Tinycore packages[294] (`*.tce`, `*.tcel`, `*.tcem`, `*.tcz`)

```
# tazpkg convert <em>alien-package-file</em>
```

9.2.6.5.14 list-suggested

List suggested packages for each of your installed packages. With option `--all` it shows all suggested packages, and without option it shows only non installed suggested packages.

```
$ tazpkg list-suggested
$ tazpkg list-suggested --all
```

9.2.6.6 Working with repositories

9.2.6.6.1 recharge

Recharge the list of available packages on the mirror. This command will download the most recent packages database of installable packages on the mirror and before starting will save the old database. Once the DB is updated, you can then use the *list* (page 462) and *search* (page 463) commands. To view and list the differences, you can use **`list-mirror --diff`**; and to view and update packages, you can simply *upgrade* (page 469).

Command without options will recharge databases of all your repositories. You can specify the repository to be recharged: “main” for main repo, or undigest repository name.

```
# tazpkg recharge
# tazpkg recharge main
# tazpkg recharge My_Undigest
# tazpkg recharge --root=/mnt/sda6
```

9.2.6.6.2 upgrade

Upgrade allows you to update all installed packages available on the current mirror (`upgrade` or `up`). Upgrading packages is an important part of system security, it helps to keep you secure with the latest updates and fixes. The SliTaz project, although tiny, provides regular updates on security and generally offers the latest versions of software. Note that this function is aimed at people with SliTaz installed on a hard drive. Updated packages in Live CD mode will be lost on system shutdown.

At the beginning the packages database is updated automatically (*recharge* (page 469)) in order to provide you with the current list of packages that you can update.

[291] http://0.tuxfamily.org/doku.php/paquets/start
[292] http://www.paldo.org/index-section-packages.html
[293] http://www.voidlinux.eu/packages/
[294] http://tinycorelinux.net/

Without options it runs in interactive mode and asks before install. You can specify one of the next options: `-c` or `--check` to check only for available upgrades; `-i` or `--install` to check for upgrades and install them all.

```
# tazpkg upgrade
# tazpkg up --check
# tazpkg up -i
```

9.2.6.6.3 setup-mirror

Setup the URL for the mirror (`setup-mirror` or `-sm`). Command will ask for the URL of the new mirror. Note that you can also modify the main `/var/lib/tazpkg/mirror` file. The URL must point to the directory containing the `packages.info` and packages.

```
# tazpkg setup-mirror
```

9.2.6.6.4 add-undigest, setup-undigest

Set the URL of an additional unofficial mirror to test packages that are not yet present on the official mirrors. Note, you can also manually edit the file in `/var/lib/tazpkg/undigest/`*`repository`*. The URL must point to the directory containing the packages and `packages.info`.

```
# tazpkg add-undigest public-repository http://my.home.org/slitaz
# tazpkg setup-undigest local-repository /home/slitaz/packages
```

9.2.6.6.5 list-undigest

Lists additional undigest mirrors. Option `--box` will output list in the table form.

```
$ tazpkg list-undigest
$ tazpkg list-undigest --box
```

9.2.6.6.6 remove-undigest

Removes the URL of an undigest mirror. You will be asked for confirmation.

```
# tazpkg remove-undigest my-repository
```

9.2.6.6.7 mkdb

Make a TazPkg database for a selected folder with `*.tazpkg` packages.

The following files describing packages will be created inside the selected folder: `packages.info`, `packages.equiv`, `descriptions.txt`, `files.list.lzma`, `IDs`. Do nothing if the database already exists; you can also force database files rebuilding with the `--forced` option.

```
# tazpkg mkdb /home/boot/packages
# tazpkg mkdb /home/boot/packages --forced
```

9.2.7 Maintainer

Christophe Lincoln <pankso@slitaz.org>

9.3 Tazlito Manual

9.3.1 NAME

Tazlito — SliTaz Live Tool.

9.3.2 SYNTAX

```
tazlito [command] [list|iso|flavor] [dir]
```

9.3.3 DESCRIPTION

Tazlito is a small utility to extract a LiveCD, rebuild the ISO image and regenerate the root filesystem of the LiveCD. **Tazlito** can also generate a distribution from a list of packages previously downloaded. To run, **Tazlito** uses the configuration file `/etc/tazlito/tazlito.conf` or an easily generated `tazlito.conf` found in the current directory. It specifies the name of the ISO, volume, maintainer and the paths of the packages to distribute and the generated ISO. **Tazlito** can also set up a directory containing additional files which will be copied to the LiveCD when generating the distribution.

Tazlito is distributed under the free GNU licence GPL v.3, installed by default on SliTaz and installed/successfully tested on Debian GNU/Linux. You will find additional information about creating a LiveCD in the Handbook.

9.3.4 COMMANDS

9.3.4.1 usage

The `usage` command displays a summary of available commands with a short description:

```
# tazlito usage
```

9.3.4.2 stats

`stats` displays the configuration variables, the paths to the various files and directories, and information on the ISO image:

```
# tazlito stats
```

9.3.4.3 gen-config

The `gen-config` command allows you to generate a configuration file ready to be edited. By default the file is created in the current directory, but can be in another directory if specified via the command line:

```
# tazlito gen-config
# tazlito gen-config /path/to/distro
```

9.3.4.4 configure

This command configures the system configuration file or one found in the current directory:

```
# tazlito configure
```

9.3.4.5 gen-iso

The `gen-iso` command can generate a new LiveCD image following modifications and additions to the root filesystem of the CD-ROM. To function, this command needs a directory containing the distro-tree of the Live system. This tree can easily be built with the *extract-distro* (page 474) command, modified and rebuilt via:

```
# tazlito gen-iso
```

9.3.4.6 gen-initiso

The `gen-initiso` command will do the same work as `gen-iso`, but it rebuilds the initramfs compressed system prior. The initramfs contains the root filesystem and must be rebuilt if modified:

```
# tazlito gen-initiso
```

9.3.4.7 list-flavors

The `list-flavors` command downloads (if necessary) and displays a list of the different flavors available. You can force the download with the `--recharge` option:

```
# tazlito list-flavors
# tazlito list-flavors --recharge
```

9.3.4.8 get-flavor

The `get-flavor` command downloads (if necessary) and prepares the files for *gen-distro* (page 474) to generate a flavor:

```
# tazlito get-flavor particular-flavor
```

9.3.4.9 show-flavor

The `show-flavor` command displays the description of the flavor and its size after regeneration. The options `--brief` and `--noheader` reduce the output displayed:

```
# tazlito show-flavor particular-flavor
# tazlito show-flavor particular-flavor --brief
# tazlito show-flavor particular-flavor --brief --noheader
```

9.3.4.10 gen-flavor

The `gen-flavor` command creates a description file of a new flavor from the results of generating a distro (*gen-distro* (page 474)). The `.flavor` file can then be sent to slitaz.org:

```
# tazlito gen-flavor new-flavor
```

9.3.4.11 gen-liveflavor

The `gen-liveflavor` command creates a description file of a new flavor from the results of generating a distro based on the current system. The `--help` option provides more information:

```
# tazlito gen-liveflavor
# tazlito gen-liveflavor --help
```

9.3.4.12 upgrade-flavor

The `upgrade-flavor` command refreshes a flavor file by updating the list of packages with the latest versions available:

```
# tazlito upgrade-flavor this-flavor
```

9.3.4.13 extract-flavor

The `extract-flavor` command converts a flavor into an easily modifiable tree structure in `/home/slitaz/`*`VERSION`*`/flavors` which can be managed with mercurial: http://hg.slitaz.org/flavors. For example on cooking you will have the work directory in `/home/slitaz/cooking`.

```
# tazlito extract-flavor this-flavor
```

9.3.4.14 pack-flavor

The `pack-flavor` command converts a tree structure in `/home/slitaz/`*`VERSION`*`/flavors` into a flavor file (`.flavor`). It is inverse of *extract-flavor* (page 473):

```
# tazlito pack-flavor this-flavor
```

9.3.4.15 extract-distro

The `extract-distro` command is used to extract an ISO image from the LiveCD to rebuild the structure of the root CD-ROM and system. It is then possible to make the desired changes or additions and rebuild the ISO image via *gen-iso* (page 472) or *gen-initiso* (page 472). Example of use:

```
# tazlito extract-distro slitaz-cooking.iso
```

9.3.4.16 gen-distro

The *Generate Distribution* command can generate the distro-tree and an ISO image via a list of packages. To function, this command needs a list of packages, a directory containing all the (`.tazpkg`) packages on the list, and a directory to generate the distribution. The list of packages can be extracted from a flavor with the *get-flavor* (page 472) command. If one uses the LiveCD, the options `--cdrom` and `--iso=` permit the regeneration of packages that place files in `/boot` without being obliged to download them and recovers the additional files of the LiveCD. The path to the various directories are configured in the configuration file and packages can be downloaded from the SliTaz mirrors or generated by Cookutils. To generate a distribution:

```
# tazlito gen-distro
# tazlito gen-distro --cdrom
# tazlito gen-distro --iso=slitaz.iso
# tazlito gen-distro package-list
```

9.3.4.17 clean-distro

Removes all files generated or extracts of the structure of the LiveCD:

```
# tazlito clean-distro
```

9.3.4.18 check-distro

This command simply verifies if files installed by the packages are present on the system:

```
# tazlito check-distro
```

9.3.4.19 writeiso

This command will write the current filesystem to a cpio archive (`rootfs.gz`) and then generate a bootable ISO image. Writeiso can be used in a HD install or in live mode and will also archive your current `/home` directory. This command lets you easily remaster and build your own LiveCD image, just boot, modify any files, and then:

```
# tazlito writeiso [gzip|lzma|none]
# tazlito writeiso gzip
# tazlito writeiso gzip image-name
```

9.3.4.20 check-list

Checks if the `distro-packages.list` is updated with the latest package versions:

```
# tazlito check-list
```

9.3.4.21 repack

Recompresses the rootfs with the best possible compression:

```
# tazlito repack slitaz.iso
```

9.3.4.22 merge

Combines several flavors like nested Russian dolls. Each rootfs is a subset of the previous. The first rootfs is extracted from the ISO image used in the third argument. The flavor will then be chosen to launch at startup according to the amount of RAM available:

```
# tazlito merge 160M slitaz-core.iso 96M rootfs-justx.gz 32M rootfs-base.gz
```

9.3.4.23 build-loram

Creates an ISO image flavor for low RAM systems from a SliTaz ISO image. You can build a flavor with / always in RAM or where / resides on the CD-ROM:

```
# tazlito build-loram slitaz.iso loram.iso
# tazlito build-loram slitaz.iso loram-cdrom.iso cdrom
```

9.3.4.24 emu-iso

The `emu-iso` command uses the Qemu emulator to start and run SliTaz. Qemu is used to test the newly built ISO image without burning to a CD-ROM or booting into frugal mode:

```
# tazlito emu-iso
# tazlito emu-iso path/to/image.iso
```

9.3.4.25 burn-iso

`burn-iso` will guess the CD-ROM device and its speed, and **`wodim`** (part of **`cdrkit`**) will begin to burn an ISO image. The default ISO image is the one located in the current configuration file, but it's possible to specify a different image via the command line:

```
# tazlito burn-iso
# tazlito burn-iso slitaz-hacked.iso
```

9.3.5 FLAVORS

A `.flavor` file contains just a few KB of information needed to (re)manufacture a custom LiveCD of SliTaz.

9.3.5.1 Manufacture a flavor

You can choose the flavor to (re)manufacture from among those available:

```
# tazlito list-flavors

List of flavors
===========================================================================
Name      ISO    Rootfs Description
===========================================================================
base        6.9M  13.1M Minimal set of packages to boot
core-3in1 31.5M 105.6M SliTaz core system with justX and base alternatives
core      31.5M 104.6M SliTaz core system
eeepc     31.2M 105.4M SliTaz eeepc system
justX     16.1M  51.2M SliTaz with a minimal X environment
```

We will start by remanufacturing the *eeepc* flavor which uses 105.4M of RAM and has a CD-ROM size of 31.2M:

```
# tazlito clean-distro
# tazlito get-flavor eeepc
# tazlito gen-distro
```

9.3.5.2 Create a flavor

To create a flavor, you must:

- Either create an ISO image with *gen-distro* (page 474) and then create a flavor file with *gen-flavor* (page 473)
- Either directly create the tree structure that defines the flavor (see *extract-flavor* (page 473)) and then create the flavor with *pack-flavor* (page 473)
- Either use the online builder[295]

9.3.5.3 Post a flavor

Because a `.flavor` file contains just a few KB, it can be easily sent via the mailing list[296].

[295] http://pizza.slitaz.org/
[296] http://www.slitaz.org/en/mailing-list.php

The results of *extract-flavor* (page 473) can also be put in mercurial[297]. This method is preferred because the tree will be directly visible with the mercurial web interface[298].

This tree includes:

- A `receipt` file describing the flavor thanks to the variables:
  - `FLAVOR`: The flavor name.
  - `SHORT_DESC`: Short description.
  - `VERSION`: Free format.
  - `MAINTAINER`: Email address of maintainer.
  - `FRUGAL_RAM`: Minimum RAM required (optional).
  - `ROOTFS_SIZE`: Size of `rootfs.gz` decompressed into RAM (optional).
  - `INITRAMFS_SIZE`: Size of `rootfs.gz` on the CD-ROM (optional).
  - `ISO_SIZE`: Size of CD-ROM (optional).
  - `ROOTFS_SELECTION`: Optional, see *Meta flavor* (page 478) below.
- The file `packages.list` containing the list of packages without specifying the version (**`tazlito`** uses the latest available). This file is missing if `ROOTFS_SELECTION` exists in the receipt.
- The optional `mirrors` file containing the list of unofficial mirrors (undigest) to be added to include personal packages.
- The optional directory `rootfs` containing the tree to add to the root filesystem `rootfs.gz` (configuration files usually).
- The optional directory `rootcd` containing the tree to add to the root of the CD-ROM.

9.3.5.4 Adapt a flavor

It is often easier to modify an existing flavor than to create one from scratch. To adapt the eeepc flavor for example:

```
# tazpkg get-install mercurial
# cd /home/slitaz
# hg clone http://hg.slitaz.org/flavors
# cd flavors
# cp -a eeepc myslitaz
```

Files in `myslitaz` can then be changed, and:

```
# tazlito pack-flavor myslitaz
```

Will simply create the new flavor.

Tip: You can skip **`mercurial`** installation by extracting a flavor. Using the previous example:

[297] http://hg.slitaz.org/flavors
[298] http://hg.slitaz.org/flavors/file/tip

```
# tazlito get-flavor eeepc
# tazlito extract-flavor eeepc.flavor
# cd /home/slitaz/flavors
# cp -a eeepc myslitaz
```

9.3.5.5 Meta flavor

A meta flavor contains several flavors like nested Russian dolls. The flavor will be launched at startup according to the amount of RAM available. The `ROOTFS_SELECTION` variable defines the minimum RAM and corresponding flavor parameters, example[299]:

```
ROOTFS_SELECTION="160M core 96M justX 32M base"
```

A meta flavor doesn't contain a list of packages (`packages.list`). SliTaz kernels prior to 2.6.30 do not support meta flavors.

9.3.6 MAINTAINER

- Christophe Lincoln <pankso@slitaz.org>
- Pascal Bellard <pascal.bellard@slitaz.org>

9.4 TazUSB Manual

9.4.1 NAME

TazUSB — SliTaz LiveUSB utility

9.4.2 SYNTAX

```
tazusb [command] [compression|device|file]
```

9.4.3 DESCRIPTION

TazUSB is a utility designed for installing SliTaz to a USB drive. Unlike a hard drive install, the filesystem is kept in a compressed `rootfs.gz` file. The filesystem is loaded entirely into memory upon boot. This should increase responsiveness, protect the filesystem against accidental corruption and reduce read/writes to the USB drive. Once setup, this utility can also rewrite the root filesystem with any changes you have made since booting up, giving the effective benefits of a hard drive install.

TazUSB supports FAT32/EXT3/EXT2 formatted drives using SYSLINUX and EXTLINUX respectively. `/home` is mounted on boot using the UUID of your particular flash drive. Unlike a device name, the UUID has the benefit of never changing from machine to machine.

[299] http://hg.slitaz.org/flavors/file/tip/core-4in1/receipt

9.4.4 COMMANDS

usage The command 'usage' will display a short summary of all available commands.

```
# tazusb usage
```

writefs The command 'writefs' will take the current memory resident filesystem and create a `rootfs.gz` file. If your flash drive is mounted as `/home` (as it should be), the new filesystem will be copied to the drive for you, otherwise it is left on the root of the drive. Your previous filesystem will be renamed to `previous.gz` and can be accessed on bootup by typing "previous" at the "boot:" prompt. All previous filesystems are renamed to `rootfs.gz.`*`unixtimestamp`*. These are not removed automatically, so you should periodically delete these to keep disk usage down.

Filesystem compression is supported in the form of lzma, gzip or none. Using no compression is very quick (under 5 seconds) and useful if you are experimenting with a lot of changes. By comparison, using lzma or gzip takes a few minutes but will dramatically reduce file size. This is recommended when committing permanent changes to the filesystem.

```
# tazusb writefs compression
```

Example:

```
# tazusb writefs lzma
```

format The command 'format' is used for formatting a device for use as a LiveUSB device. Currently, it supports formatting as EXT2, EXT3 and FAT32.

```
# tazusb format /dev/name
```

Example:

```
# tazusb format /dev/sda1
```

gen-liveusb "gen-liveusb" will install a fresh MBR, set your partition as bootable and install syslinux/extlinux depending on the detected filesystem. It will then copy the kernel and filesystem from the CD-ROM drive, and place this on the target USB drive. This will leave you with a bootable USB copy of SliTaz.

```
# tazusb gen-liveusb /dev/name
```

Example:

```
# tazusb gen-liveusb /dev/sda1
```

gen-swap The 'gen-swap' command (re)creates a virtual swap file and places it in the `/home` directory to be activated on each boot. This is useful for old systems with low memory.

```
# tazusb gen-swap
```

gen-iso2usb This command performs the same task as gen-liveusb, only copying the Kernel and filesystem from a downloaded ISO image instead of the CD-ROM drive.

```
# tazusb gen-iso2usb /path/to/iso
```

Example:

```
# tazusb gen-iso2usb /home/tux/slitaz.iso
```

clean "clean" removes old `rootfs.gz.`*`unixtimestamp`* filesystems (see writefs) to keep disk usage down.

```
# tazusb clean
```

9.4.5 MAINTAINER

Eric Joseph-Alexandre <erjo@slitaz.org>

9.5 Cookutils Documentation

9.5.1 SliTaz Cook & Cooker

The SliTaz Cookutils provide tools and utils to help build SliTaz packages. They are easy to use and learn, fast and light. You will be able to create SliTaz packages in a few commands. The cookutils provide the **`cook`** utility and *The Cooker* (page 483).

`Cook` lets you compile and create a package, provide a log file and check the receipt/package quality. The **`Cooker`** is a build bot with more automation and can be used as a frontend to **`cook`** since it provides a CGI/web interface which lets you view cook logs in a nice and colored way. **`Cook`** and the **`Cooker`** use the same DB files and *wok*, they both share *blocked* (page 484) and broken packages as well as any activity.

For technical information, for example the coding style, etc, please refer to the `README` found in the source tree or in `/usr/share/doc/cookutils`.

9.5.1.1 Cook usage

`Cook` provides a small built-in help usage that you can display with the command **`usage`**. It also has some options to perform special tasks on a package before *cooking* it or afterwards. To get help and usage:

```
# cook usage
```

9.5.1.2 Howto

The first thing you will have to do before building packages is setup your environment. The 2 recommended ways of working: *cook* directly on host or *cook* in chroot to protect your host. In the case you want to work in a chroot you can install and use **`Tazdev`** to create one and chroot into it:

```
# tazdev gen-chroot && tazdev chroot
```

By default **Tazdev** creates a chroot in /home/slitaz/cooking/chroot but you can specify a custom path in the argument. The chroot location is not important, when you will be in the chroot you will use standard SliTaz paths such as /home/slitaz/wok for the *wok* directory or /home/slitaz/log for all the cook logs. As usual you can display **tazdev** help usage with: **tazdev usage**.

When you use a chroot there are 2 special directories mounted with the bind option: src and packages. The sources for all packages are stored by default in /home/slitaz/src, this directory is mounted into the chroot so the utils can use them. This method lets you share sources between many chroots such as one for *cooking* and one for *stable*. The packages directory default location is: /home/slitaz/*version*/packages so they are not in the chroot and are safe in case the chroot is removed by error.

9.5.1.3 Getting started

So you have decided the way you want to work, so lets prepare the cook environment. **Cook** uses the cook.conf configuration file, if you want to use custom paths for SliTaz directories and files, you'll have to modify it. The setup will create some directories and files to keep trace of activity and errors, all files are pure plain text files that you can open in a text editor. To prepare your environment:

```
# cook setup
```

The setup command has a --wok option which lets you clone a SliTaz wok while setting up your cook environment. Even if you are not yet an official developer you can clone it and use existing packages as an example to create your own. To setup and clone the default *cooking wok* or the *undigest wok*:

```
# cook setup --wok
# cook setup --undigest
```

9.5.1.4 Test your environment

Cook provides a **test** command which will create a package and *cook* it. This lets you see if your environment is working and it provides an example package with a receipt. The dummy package is named 'cooktest' and can be removed after testing. To *cook* the test package:

```
# cook test
```

9.5.1.5 Create and cook

If your environment is setup correctly you can start creating and compiling SliTaz packages from your *wok*. To create a new package with an empty receipt (you can also create a receipt interactively):

```
# cook new pkgname
# cook new pkgname --interactive
```

If you have just created a new package, you'll have to edit the receipt with your favorite text editor. When the receipt is ready or if you have an existing package, you can *cook* it:

```
# cook pkgname
```

If all went well you will find your package in the `$SLITAZ/packages` directory and any produced files in `$SLITAZ/wok/`*`pkgname`*.

9.5.1.6 Cook and install

If you want to *cook* and install the package in one command:

```
# cook pkgname --install
```

9.5.1.7 Get sources

If you want or need to download only the source of a package without building it, you can use the option `--getsrc` as below:

```
# cook pkgname --getsrc
```

9.5.1.8 Clean packages

After compilation and packaging there are several files in the *wok* that take up disk space. To clean a single package:

```
# cook pkgname --clean
```

You can also clean the full *wok* at once or you can choose to keep SliTaz related files and just remove the source:

```
# cook clean-wok
# cook clean-src
```

9.5.1.9 Search

`Cook` provides a simple search function to quickly find a package in the *wok*. It uses **`grep`** and so supports regular expressions:

```
# cook search busybox
```

9.5.1.10 Receipt functions

Many packages provide the same kind of files such as `*-dev` packages with static libs, *pkgconfig* files and include headers. So cook provides a function to be used in the receipt:

```
get_dev_files      : Install /usr/lib/{lib.*a,pkgconfig} /usr/include
```

9.5.1.11 Packages DB list

`Cook` can list packages in the *wok* and also create a suitable packages list for **`Tazpkg`**. This lets you create a local packages repository quite easily and is used to create the official SliTaz packages list found on the *mirrors*. To list the current *wok* used by **`cook`** (you don't need to be *root*):

```
$ cook list-wok
```

When creating the packages DB, **cook** will check if you have a *flavors* repo in `/home/slitaz/flavors`, if so, it will pack all *flavors* using the latest packages list available. To create a packages list and the Live *flavors* files:

```
# cook pkgdb
```

9.5.1.12 The Cooker

The **Cooker** is a Build Bot, its first function is to check for commits in a *wok*, create an ordered cooklist and *cook* all modified packages. It can also be used as a frontend to **cook** since they both use the same files. The **Cooker** can also be used to *cook* a big list of packages at once such as all the packages in a *flavor*. The **Cooker** provides a nice CGI/Web interface that works by default on any SliTaz system since it provides CGI support via the **Busybox httpd** web server.

The **Cooker** provides a small built-in help usage and short command switch. For example to display usage you can use:

```
# cooker usage
# cooker -u
```

9.5.1.13 Cooker setup

Like **cook**, the **Cooker** needs a working environment before starting to use it. The main difference with the **cook** environment is that the **Cooker** needs 2 *woks*. One Hg and clean *wok* as a reference and one build *wok*. In this way it is easy to compare both *woks* and get modifications. If you already have a *cook* environment, you must move your *wok* before setting up the **Cooker** or it will complain. Setup will also install a set of development packages that can be configured in the `cook.conf` configuration file and the variable `SETUP_PKGS`. To setup your **cooker** environment:

```
# cooker setup
```

If all went well you now have 2 *woks*, base development packages installed and all needed files created. The default behavior is to check for commits, you can run a test:

```
# cooker
```

9.5.1.14 Cooker cook

Again, 2 ways to work now: make changes in the clean Hg *wok* and launch the **cooker** without any arguments or *cook* packages manually. The **cooker** lets you *cook* a single package or all packages of a category or a *flavor*. You can also try to build all unbuilt packages, but be aware the **Cooker** was not designed to handle thousands of packages.

To *cook* a single package which is the same as **cook pkgname** but with more logs:

```
# cooker pkg pkgname
```

To *cook* more than one package at once you have different kind of choices. You can use an existing package such as used for Live *flavors*, you can also use a custom list using the package names listed line by line. Finally you can build all packages of a category.

```
# cooker flavor [name]
# cooker list [/path/to/cooklist]
# cooker cat [category]
```

The **Cooker** lets you also *recook* a specific Hg revision. It's useful in production so that if the Build Bot was interrupted while *cooking* commits, you can then *cook* packages manually:

```
# cooker rev 9496
```

9.5.1.15 Blocked packages

Cook and the **Cooker** handle a file with a list of blocked package so they will not *cook* when commits happen or if a cooklist is used. This is very useful for a **Cooker** Build Bot in production. When you block or unblock a package you can add a note to the cooknotes. Blocking packages example:

```
# cook pkgname --block
# cooker block pkgname
# cooker -n "Blocked pkgname note"
```

The list of blocked packages are also displayed on the **Cooker** web interface. To unblock a package you have to use the **unblock** command or cook `--unblock` option:

```
# cook pkgname --unblock
# cooker unblock pkgname
```

9.5.1.16 Cooker CGI/Web

To let you view log files in a nice way, keep trace of activity and help find errors, you can use the **Cooker** Web interface located by default in the folder `/var/www/cooker`. If you don't use a chroot and the **Busybox httpd** web server is running, the web interface will work without configuration and should be reachable at: http://localhost/cooker/cooker.cgi

If you used a chroot environment, you should also install **cookutils** on your host and modify the `SLITAZ` path variable. A standard working way is to have a chroot in: `/home/slitaz/cooking/chroot`

With `/etc/slitaz/cook.conf` modified as below:

```
SLITAZ="/home/slitaz/cooking/chroot/home/slitaz"
```

Note: It's not obligatory to install the **cookutils** on your host to use the web interface. If you use **Lighttpd** you can also copy the `cooker.cgi` and `style.css` files for example into your `~/Public` directory and use a custom `cook.conf` with it. The advantage of installing **cookutils** on the host is to get regular updates via the **Tazpkg** packages manager. Say you have cloned or downloaded the **cookutils**:

```
$ cp -a cookutils/web ~/Public/cgi-bin/cooker
$ cp -f cookutils/cook.conf ~/Public/cgi-bin/cooker
```

Edit the configuration file: `~/Public/cgi-bin/cooker/cook.conf` to set your `SLITAZ` path and you're all done!

9.5.1.17 Cooknotes

The cooknotes feature lets you write small personal notes about packaging and is useful for collaboration. The cooknotes was coded to let the SliTaz **`Cooker`** bot maintainers share notes between themselves and other contributors. The **`Cooker`** can block a package's build or *recook* packages manually, for example it's nice to make a note if a package is blocked so that the maintainer knows why admin did that. Cooknotes are displayed on the web interface and can be checked from a cmdline:

```
# cooker note "Blocked pkgname due to heavy CPU load"
# cooker notes
```

9.5.1.18 Cooker as a Build Bot

The **`Cooker`** is designed to be a Built Bot for SliTaz, this means it monitors 2 *woks*, updates the Hg *wok*, gets the differences and *cooks* all packages that have been committed. The safer and cleaner way to run the **`Cooker`** as a Build Bot with **`cron`** is to use a chroot environment, but it can run directly on the host if you want.

To run The **`Cooker`** automatically you must use **`cron`** from the chroot and add a single line to root `crontabs` in `/var/spool/cron/crontabs`. Say you would like to run the **`Cooker`** every 2 hours:

```
* */2 * * * /usr/bin/cooker
```

9.5.1.19 Cooker BB started at boot

The **`Cooker`** environment and **`cron`** task can automatically be started at boot time. You must have the **`cookutils-daemon`** installed on the host and use a standard SliTaz installation to make it work properly (*cooking* goes in `/home/slitaz/cooking`). The daemon script will mount any virtual filesystems if needed as well as source and packages. Source files are in `/home/slitaz/src` and bound into the chroot so you can share package's sources between several versions (*stable*, *cooking*, *undigest*). If the package is not yet installed:

```
# tazpkg get-install cookutils-daemon
```

To start the daemon you must have a **`cron`** file definition for root in the chroot, the daemon script works like all other system daemons and can be handled with:

```
# /etc/init.d/cooker [start|stop|restart]
```

9.6 Burnbox Manual

9.6.1 About

Burnbox supports:

- ISO burning
- Backup of data and audio CDs
- Data CD/DVD burning (DVD burning requires an optional package **`dvd+rw-tools`**)
- Audio CD burning (MP3 requires an optional package **`mpg123`**)
- Video CD (VCD/SVCD) burning (requires packages **`vcdimager`** and **`ffmpeg`**)

9.6.2 General Device Settings

In the last tab, make sure the CD-ROM settings are correct (Device: `/dev/cdrom`; Speed (auto-detected): 48; Options:). Input is taken from the `DEVICE` settings.

9.6.3 Blank CD/DVD-RW

Use *Blank disk* to erase contents and prepare for burning.

9.6.4 ISO Burning

This is supported in the third tab.

- Press button *Browse* to specify the ISO file path
- Press button *Burn ISO*

You can also create or manipulate ISO images with the ISO Master utility.

9.6.5 Backup of data and audio CD

This is supported in the second tab. Optional package **`cdrkit-isoinfo`** may help to improve the burn speed.

- Insert CD
- Select backup option: "Save backup on Hard Disk Folder" (default) or "Backup on CD disc"
- If backup option is "Save backup on Hard Disk Folder" (default), then specify the folder to do a "CD/DVD backup"
- Press button *Backup DataCD*

Audio CD ripping is supported using the asunder package.

9.6.6 Data CD/DVD burning

This is supported in the First tab. One can specify the burn type before adding files. Depending upon the burn type, the files are decoded on the fly. One can remove the added files by simply double-clicking on the file in the tree view. Total track size shows the CD/DVD space occupied.

- Select data-cd in Burn Type
- Specify file/folder or *Browse*
- Press button *Add*
- Press button *Burn Disc*

9.6.7 Audio CD burning

This is supported in the first tab. WAV, OGG, MP3 files are supported. OGG and MP3 are automatically converted into an uncompressed WAV format for burning.

- Select audio-cd in Burn Type
- Specify file/folder or *Browse*
- Press button *Add* to auto-decode to uncompressed WAV (OGG, MP3 are decoded)
- Press button *Burn Disc*

9.6.8 Video CD (VCD/SVCD) burning

This is supported in the first tab. Only burning MPG video format is supported directly: MPEG-1 video for VCD and MPEG-2 for SVCD and DVD. AVI, MOV, FLV, WMV videos are automatically converted into a VCD/SVCD/DVD compatible MPG video format.

- Select video-dvd, vcd or svcd in Burn Type
- Specify file/folder or *Browse*
- Press button *Add* and select Enable decoding video for MPG files (AVI, FLV, MOV, WMV are decoded)
- Press button *Burn Disc*

9.6.9 Common Problems

- Burnbox works as root but not for a user: This happens when the user is not added to the "cdrom" group

  ```
  # addgroup tux cdrom
  ```

- CD-ROM not readable / writable: Make sure the user has proper permissions and is added to the "cdrom" group
- MP3 audio burn does not work: Check if you have installed **`mpg123`**
- VCD option does not work: Check if you have installed **`vcdimager`**
- AVI, FLV, WMV files are not decoded to MPG: Check if you have installed **`ffmpeg`**

- DVD burn does not work: Check if you have installed **`dvd+rw-tools`**

CHAPTER 10

Releases

10.1 SliTaz 4.0 Errata

author pankso, linea, bellard, domcox, seacat

- Xorg fails to start at the end of boot. Solution: Preselect your language at boot menu or from the command line with: `slitaz lang= kmap=`
- Upgrade from 3.0 to 4.0 can be a pain from the Package manager. Solution: Use **SliTaz Installer** from the command line or **TazPanel** to perform an upgrade.
- Desktop bottom panel is missing. This is a bug from **LXpanel**, only the main panel is copied to user config. Solution:

```
$ cp /etc/lxpanel/slitaz/panels/* $HOME/.config/lxpanel/slitaz/panels
```

- tazpanel liveCD loram conversion does not work. Solution 1: use tazlito, example:

```
# tazlito build-loram slitaz-4.0.iso slitaz-4.0-loram.iso ram
```

  Solution 2: stop tazpanel server with the **kill** command and start it in an xterm

```
# /etc/init.d/tazpanel start
```

- **tazlito build-loram** does not update the `/mdsum` file. The `media-check` option in boot menu reports unexpected broken files. slitaz-4.0-loram-cdrom media-check reports "177 files OK, 29 broken, 4488 not checked." Because `isolinux.bin` does not support *rockridge* and does not see most files.
- 'tazlito build-loram some.iso result.iso cdrom' does not work. Solution: use this fixed version tazlito[300]

[300] http://hg.slitaz.org/tazlito/raw-file/ad96fdd80b46/tazlito

- bootfloppybox does not retrieve the boot cmdline of `slitaz-4.0.iso`. Solution: use this fixed version bootfloppybox[301]
- The ISO images `slitaz-4.0-base.iso`, `slitaz-4.0-core.iso`, `slitaz-4.0-firefox.iso`, `slitaz-4.0.iso`, `slitaz-4.0-justx.iso`, `slitaz-4.0-preinit.iso`, `slitaz-4.0-proxy.iso` and `slitaz-4.0-xorg-light.iso` are not hybrid. Solution 1: run **`isohybrid`** on the image, example:

  ```
  $ isohybrid slitaz-4.0.iso
  ```

  Solution 2: use `slitaz-4.0-loram-cdrom.iso` or `slitaz-4.0-loram.iso`.
- The installer in **`tazpanel`** does not find **`tazinst`** if you use the `pt_BR` locale. Solution: this bug has been fixed, update your system with the latest **`tazpanel`** and **`slitaz-tools`** packages.

10.2 SliTaz 5.0 Errata

author pankso

10.3 SliTaz 6.0 Errata

author pankso

10.4 SliTaz 6.0 Release Notes

author pankso

[301] http://hg.slitaz.org/slitaz-tools/raw-file/fb636ac549ab/tinyutils/bootfloppybox

CHAPTER 11

SliTaz Base Corner

author oui, linea

Animated by this starting thread[302] at the forum, I felt the need for a central place to house information on how use SliTaz in pure console mode installing only the base of SliTaz!

As this place is now available, we can feed it with useful information and links with the intention of helping to start more easily — as the first objective of SliTaz is to be a graphical Linux system and not really destined to be used as base only. . .

11.1 Installation alt. use base without installation (frugal method)

11.1.1 Read information!

The installation of the base can use the same methods (as described in following pages) for the complete installation:

- *Using the LiveCD* (page 274)
- *Hard Disk Installation* (page 286)
- *Unusual install methods* (page 116)

being conscious that you only install a part of them! That part is included in the ISO and is named `rootfs4.gz`.

11.1.2 Transfer onto hard disk for a frugal start

As you will work in the command line, I assume you also will make instructions in a command line!

Step-by-step operations:

First step: we need an ISO file containing the software that we will use:

[302] http://forum.slitaz.org/topic/how-to-use-my-pc-mainly-in-cli

SliTaz offers clear and strong organisation:

- home[303] is the main site with a special page to search[304] for all information about SliTaz in all SliTaz pages and media
- doc[305] is the starting point for your search for documentation
- forum[306] is our common forum in all languages
- mirror[307] is the starting point for your downloads

So we will have to search for our ISO file in the last one on the mirror! The organisation is very simple! I will use the actual ISO for the new version 5.0 in elaboration. I visit the mirror with my browser and find the file needed. I will download it with **wget** by typing in the command line:

```
$ wget http://mirror.slitaz.org/iso/5.0/slitaz-5.0-rc2.iso
$ ls
```

wget echoes for ex. `(914 KB/s) - slitaz-5.0-rc2.iso saved.`

I find the file ready in my home directory: **ls** echoes the file name in the population of my home!

I now have to know a bit about my actual system. Where is the location of my compact disk drive in the tree? For example `/mnt/cdrom` or `/media/cdrom`?

I will mount the file at this point:

I type in the command line (or copy it from here marking / hitting on the middle mouse key):

```
# mount -o loop,ro
```

marking the preceding lines with the name of the ISO file with the mouse and hitting after the beginning of the line with the middle mouse key,

```
# mount -o loop,ro slitaz-5.0-rc2.iso
```

and continue as follows:

```
# mount -o loop,ro slitaz-5.0-rc2.iso /mnt/cdrom
# ls /mnt/cdrom
```

(or `/media/cdrom` etc.)

ls /mnt/cdrom echoes:

```
boot  index.html  md5sum  README  style.css
```

I look deeper:

```
# ls /mnt/cdrom/boot
```

and can see the entries for the files I am looking for (marked in the following view):

> **bzImage** grub ipxe isolinux memtest rootfs1.gz rootfs2.gz rootfs3.gz **rootfs4.gz vmlinuz-3.2.53-slitaz**

[303] http://www.slitaz.org/
[304] http://www.slitaz.org/en/search.php
[305] http://doc.slitaz.org/
[306] http://forum.slitaz.org/
[307] http://mirror.slitaz.org/

I will now copy those 3 files using:

```
# cp -a FILENAME
```

into a new directory in the root of an adequate partition (please not a Windows NTFS! But it can be a fat32 from an old Windows…).

I only have to add now an adequate entry in my `grub.cfg` or `/.../grub/menu.lst` to start my new SliTaz base using **GRUB**!

You need only a few minutes to do all this and can immediately begin after restarting to experiment in the frugal started SliTaz base system!

This method has a major inconvenience: All what you do in the frugal started SliTaz base is not persistent! If you want to preserve your next steps you have to decide between two ways: remaster your SliTaz (if you find it is to much) or install fully your new SliTaz base somewhere (probably your own little partition for it, — see the following item, — or you can look *Subdirectory install in a Posix filesystem* (page 120)).

11.2 Install the SliTaz base on an own little partition

I assume now you used the above frugal start to proceed to that full installation and that your ISO file is still in the same partition in your preceding ~ (home) directory! If not you only have to repeat later some of preceding steps but I will not describe them again…

I also assume you have already created the newly needed partition (if not look for information on how to do that in the pages linked under the above item *Read information!* (page 491)). The partition doesn't have to be empty: Only the usual Linux directories have to be moved somewhere else in that partition (for ex. `/oldSystem`!).

You started using **GRUB** and the **login** console appears. Please enter the SliTaz standard username **tux**. It needs no password :-) ! To work as superuser you will later need the superuser password. In the new system it is **root** (you can change it in the full installation using the command **passwd** and following the dialog!). But as user, you can continue with **tux**, why not!

11.3 Choice of standard SliTaz packages useful in a SliTaz base system

11.3.1 Installation of packages

As you are always in a command line system, you have only options:

- install packages with *Tazpkg* (page 456). Example:

  ```
  # tazpkg get-install gpm
  ```

- compile a non SliTaz package (very useful in console mode: **didiwiki**, works well in **lynx** or **links**, **wordgrinder**, text processor for CLI)
- add per simple copy (buuuuuuuuuuh!) :-| a binary package from another distro and hope it works and does nothing bad…

11.3.2 Usual jobs

- mouse driver under console mode: **`gpm`** (this author did also write an interesting text editor with full performance using … the mouse of course!)
- **`sudo`**
- directories and file manager: **`clex`**
- commander: **`mc`**
- email client: **`alpine`**
- **`alpine`** includes the very interesting text editor: **`pico`** (SliTaz includes **`e3`** in **`vi`** mode, uses trad. **`nano`**, and proposes also **`vim`**)
- IRC client: **`rhapsody`**
- web browser: **`lynx`** (or **`links`**, but **`links`** is having a few dependencies as it would be also usable in graphical mode, where it can show pictures)
- *retawq* did show me only the code of http://encrypted.google.com (retawq was the trad. SliTaz browser)
- *elinks* has too many dependencies in SliTaz…
- spreadsheet: **`sc`**

11.3.3 Server jobs

11.4 Helpful documents

- the, of course, most interesting documents are the man pages of diverse **`BASH`** commands including **`BASH`** itself and of the above app's! This may be really difficult to get in foreign languages and I can't say that the Google translation is very useful in this case depending on the language. For my own use, I did collect the text of the following man pages: **`adduser`**, **`alpine`**, **`ar`** (useful to extract sources from Debian), **`arch`**, **`ark`**, **`cat`**, **`chmod`**, **`chown`**, **`chroot`**, **`clear`**, **`cp`**, **`cut`**, **`date`**, **`dd`**, **`debmany`** (to search Debian man pages), **`didiwiki`**, **`dpkg`** (regarding Debian), **`e2freefrag`**, **`e2fsck`**, **`export`**, **`filefrag`**, **`fim`** (see on the internet, perhaps http://savannah.nongnu.org/projects/fbi-improved/!), **`free`**, **`gpm`**, **`groupadd`**, **`imagemagick`**, **`init`**, **`less`**, **`libsvga`**, **`links`**, **`ln`**, **`locate`**, **`ls`**, **`lynx`**, **`man2html`** (good!), **`mkdir`**, **`mkfs`**, **`more`**, **`mount`**, **`mv`**, **`pico`**, **`pr`**, **`retawq`**, **`rm`**, **`rsync`**, **`sc`**, **`su`**, **`tail`**, **`tar`**, **`uuid`**, etc. …

11.5 Example of Grub texts

CHAPTER 12

Old Cookbook

author linea

The Old Cookbook brings together deprecated information about the project management, operation and development of the distribution. It talks about creating packages, receipts, the wok, and scripts that start SliTaz.

Table of contents

12.1 Recipes

author linea

This document describes the opportunities offered by the recipes used by **Tazwok** to compile and generate packages for SliTaz and **Tazpkg** through The wok and tools. The recipe for a package is also used by **Tazpkg** to install/uninstall and provide information about a `.tazpkg` package. Each recipe begins with a comment in English:

```
# SliTaz package receipt
```

12.1.1 Variables

The first 5 variables should always be present and defined. They respectively configure the package (`$PACKAGE`), its version, its category, provide a short description and the name of the maintainer. Example for the package, file manager **Clex**:

```
PACKAGE="clex"
VERSION="3.16"
CATEGORY="base-apps"
SHORT_DESC="Text mode file manager."
MAINTAINER="pankso@slitaz.org"
```

12.1.2 Variables (optional)

Tazwok also knows how to use various optional variables. It can, for example, use the name of another source package. There are also variables that are used by **Tazpkg** to manage dependencies or provide information about the package.

$DEPENDS: Set dependencies, there may be several dependencies separated by a space or on several lines. This variable is used mainly by **Tazpkg** when installing the package and **Tazwok** to build large packages such as **Xorg**. Example for **Clex** which depends on **ncurses**:

```
DEPENDS="ncurses"
```

$BUILD_DEPENDS: Set compilation dependencies, again separated by a space or several lines. This variable is used by **Tazwok** during the cooking of a package. Example:

```
BUILD_DEPENDS="ncurses-dev"
```

$TARBALL: The archive is a source with the extension (`tar.gz`, `tgz` or `tar.bz2`). In general, the variables `$PACKAGE` and `$VERSION` are used to just change the extension, it helps to upgrade the package without changing the `$VERSION` variable. Generic example (see also `$SOURCE` example):

```
TARBALL="$PACKAGE-$VERSION.tar.gz"
```

$WEB_SITE: The official website of the package. It may be that some libraries have no website, in this case, there is no need to specify a URL. Note **Tazwok** and **Tazpkg** both expect to find a URL with the complete HTTP:

```
WEB_SITE="http://www.clex.sk/"
```

$WGET_URL: URL to download the source file. In general the variable `$TARBALL` should be used to facilitate the updating of the package without changing the `$VERSION`. Using a configuration file, **Tazwok** also configures by default 3 mirrors: `$GNU_MIRROR` for the GNU mirror, `$SF_MIRROR` for SourceForge and `XORG_MIRROR` for mirroring the graphical server **Xorg**. Example for **Clex**:

```
WGET_URL="http://www.clex.sk/download/$TARBALL"
```

$CONFIG_FILES: Some packages provide customized configuration files. The `$CONFIG_FILES` variable provides a list of these files that can be saved by the **tazpkg repack-config** command. These files are not overwritten when reinstalling the package if they already exist and the package can be successfully recreated with **tazpkg repack**, (even if they have been modified since). **Netatalk** for example:

```
CONFIG_FILES="/etc/netatalk/AppleVolumes.* /etc/netatalk/*.conf"
```

$SUGGESTED: Lists useful packages without being essential. Also used to activate optional features.

$WANTED: SliTaz packages normally depend on the compilation of a source package. Sometimes the recipe of a package requires no compilation of rules, then `$WANTED` is used to copy files from the source of another package by using the variable `$src`.

$SOURCE: It may be that the **Tazpkg** package name differs from the name of the source package. Example for **Xorg** packages, the name of **Tazpkg** library **X11** is `xorg-libX11` and the name of the package source is `libX11`. `$SOURCE` allows you to use the variables `$src` and `$_pkg` during the cooking of a package. It should be noted that in the case of `libX11`, the name of the source archive becomes `$SOURCE-$VERSION.tar.gz`.

$PROVIDE: Some packages offer the same functionality, for instance the web server was at first **lighttpd**; now **apache** is available. All packages dependent on a web server refer to **lighttpd**. The `PROVIDE="lighttpd"` variable in the **apache** recipe states that packages dependent on **lighttpd** do not need to install the **lighttpd** package if **apache** is already on the system.

Some packages may vary according to the webserver you choose, ie. the **php** package is dependent on **lighttpd**, as is **php-apache** on **apache**. The `PROVIDE="php:apache"` in the **apache** recipe says that you must install **php-apache** instead of **php**, if **apache** is already on the system. Therefore each package dependent on **php** will install either **php-apache** or **php** according to the webserver on the system.

This variable also chooses packages compiled with different options. The `PROVIDE="epdfview:cups"` in the **epdfview-cups** recipe allows you to install **epdfview** with printer support via **cups** if **cups** is already on the system.

You can also define virtual packages with this variable. The lines `PROVIDE="libgl"` in the **mesa** package and `PROVIDE="libgl:nvidia"` in the **nvidia-glx** package, define that **libgl** is an optimized version when the **nvidia** package is installed.

$SELF_INSTALL (obsolete): Certain packages use commands provided by the package itself in the `post_install` function. To install this package into a directory other than root and still be able to use these commands, the package must have been installed in / in earlier stages. The line: `SELF_INSTALL=1` alerts **tazpkg** to this feature. This variable is depreciated. The command **chroot "$1/" a_package_command** in `post_install` does the job.

12.1.2.1 Variables generated by Tazwok

Certain factors are known only during the cooking of a package or after the package has been cooked. **Tazwok** will add them to the recipe automatically.

$PACKED_SIZE: **Tazpkg** file size.

$UNPACKED_SIZE: Space taken up by the package after installation.

$EXTRAVERSION: Some packages have 2 different versions. This is in case of modules added to the Linux kernel, such as **squashfs**, because the module depends on the version of the kernel with which it was compiled. In this case `$EXTRAVERSION` contains the kernel version and **Tazwok** determines the module from the contents of `/lib/modules`.

12.1.2.2 Variables used in functions

Cookutils configures several variables that facilitate the compilation and construction of **Tazpkg** packages. These variables are controlled automatically by **cookutils** using the information contained in the recipe; they can be used by the functions `compile_rules` and `genpkg_rules` described in the chapter Functions.

`$src`: Defines the path to the directory of unarchived sources.

`$_pkg`: Defines the path to the compiled binaries installed via **`make DESTDIR=$PWD/_pkg install`**. This variable is used to copy the generated files and create **`Tazpkg`** packages.

`$install`: Same as `$_pkg`.

`$fs`: Defines the path to the pseudo filesystem (fs) in each package. The 'fs' of the package corresponds to the root of the system, a bit like **`Clex`** will for example be in `$fs/usr/bin/clex`. Note the need to create the necessary directories via function `genpkg_rules()` before copying the files.

`$CONFIGURE_ARGS`: This variable is defined in the **`cookutils`** configuration file (`cook.conf`). It allows you to specify generic optimization arguments during construction of a package. Default is the i486 architecture.

`$DESTDIR`: Defines the path to install compiled binaries after the build via **`make DESTDIR=$DESTDIR install`**.

12.1.3 Functions

A recipe may contain 4 functions. **`Tazwok`** knows how to deal with functions containing compilation rules (`compile_rules`) and rules used to generate a package (`genpkg_rules`). These functions may contain all sorts of GNU/Linux standard commands, such as **`sed`**, **`awk`**, **`patch`** and variables automatically configured.

`compile_rules()`

To compile a package you can use the variable `$src` to move (**`cd`**) in the directory of sources and use `$CONFIGURE_ARGS` to include arguments from the Tazwok configuration file. To build the package you usually launch **`make`** without any arguments, and to install the package into the directory `_pkg`: it's necessary to use the command **`make DESTDIR=$PWD/_pkg install`**. Generic example:

```
# Rules to configure and make the package.
compile_rules()
{
     cd $src
     ./configure --prefix=/usr --infodir=/usr/share/info \
     --mandir=/usr/share/man $CONFIGURE_ARGS
     make
     make DESTDIR=$PWD/_pkg install
}
```

`genpkg_rules()`

To generate a **`tazkg`** package we must specify commands in the function `genpkg_rules`. In this example we create a psuedo directory `/usr` in the filesystem of the package, copy the whole binary(s) and finally use **`strip`** to clean the files:

```
# Rules to gen a SliTaz package suitable for Tazpkg.
genpkg_rules()
{
     mkdir -p $fs/usr
     cp -a $_pkg/usr/bin $fs/usr
     strip -s $fs/usr/bin/*
}
```

`pre_install()` and `post_install()`

These functions are initiated by **`Tazpkg`** when installing the package. They must be defined before generating the `.tazpkg` package with **`Tazwok`**. If no rules are given for these functions, they have no raison d'etre and can be removed. Example using **echo** to display some text (no function should be empty):

```
# Pre and post install commands for Tazpkg.
pre_install()
{
     echo "Processing pre-install commands..."
}
post_install()
{
     echo "Processing post-install commands..."
}
```

`pre_remove()` and `post_remove()`

These functions are initiated by **`Tazpkg`** when removing the package. They must be defined before generating the `.tazpkg` package with **`Tazwok`**. If no rules are given for these functions, they have no raison d'etre and can be removed. Example using **echo** to display some text (no function should be empty):

```
# Pre and post remove commands for Tazpkg.
pre_remove()
{
     echo "Processing pre-remove commands..."
}
post_remove()
{
     echo "Processing post-remove commands..."
}
```

`clean_wok()` (deprecated)

This is useless with latest **`cookutils`**, source are all uncompressed in `wok/`*`pkg`*`/source` to keep build wok clean and structured.

This function helps to define additional commands to be run when cleaning the wok, it is useful to delete files or directories that are not supported by **`Tazwok`**:

```
# clean commands for Tazwok.
clean_wok()
{
     rm -rf $WOK/$PACKAGE/vim71
}
```

12.2 Wok & Tools

author linea

12.2.1 Overview

Tazwok[308] is used to compile and generate code (cooking) via instructions found in a receipt. It places compiled files in to a directory and calls upon **Tazpkg** to package said directory. The receipt found in a wok has a different "bottom half" to that of a **Tazpkg**; **Tazwok** has rules for compilation *and* packaging (which it forwards to **Tazpkg**), whereas **Tazpkg** is only concerned with packaging.

Tazwok is one of many small utilities the SliTaz project uses to automatically rebuild the distribution from source. The project also offers an archive of *tools* (page 380) containing various small utilities, examples and configuration files. The distribution generator Tazlito[309] is designed for users and developers; it can retrieve and reconstruct a LiveCD ISO image and generate a distribution flavor from a list of packages, a configuration file and a description. The utilities are all distributed as a source archive and are installed by default on SliTaz.

Tip: Developers and future contributors can refer to the development page that provides information on SliTaz project management.

12.2.1.1 Wok Structure and Organisation

The `wok` is a directory structure that houses all the available packages. Each directory contains at least one `receipt` to download, unpack, compile and generate a package. **Tazwok** also needs to create a directory to store downloaded sources (`$SOURCES_REPOSITORY`, usually `/home/slitaz/src`) and a repository of generated packages (`$PACKAGES_REPOSITORY`, usually `/home/slitaz/packages`); these values can be configured in the `/etc/tazwok.conf` file.

There is more than one Wok on the Mercurial repositories[310]:

wok-undigest: contributions awaiting testing/bug-fixing for inclusion in the unstable Wok

wok: packages for the unstable, Cooking release

wok-stable: packages for the stable SliTaz release

Initially, any contributions will be committed to the *undigest* repository. When the package has seen sufficient testing with regards to automatic generation, it can be moved to the Wok.

[308] http://hg.slitaz.org/tazwok/raw-file/tip/doc/tazwok.en.html
[309] http://hg.slitaz.org/tazwok/raw-file/tip/doc/tazwok.en.html
[310] http://hg.slitaz.org

12.2.2 Preparation

The Developer's Corner provides invaluable background information. Please ensure you have read and understood it before continuing.

To begin using the Wok, Tazwok[311] must already be installed on the system along with the main development tools (**binutils**, compiler, libraries-dev, **make**). This requires you to install the meta-package **slitaz-toolchain**:

```
# tazpkg recharge
# tazpkg get-install slitaz-toolchain
```

To access the SliTaz repositories, you will need to install the **mercurial** package:

```
# tazpkg get-install mercurial
```

More information on the use of the **Mercurial** VCS is available from its website[312] and the "Hg Book[313]".

12.2.2.1 Cloning the Wok

If you are to generate a package for inclusion in the SliTaz repositories, it is necessary to first obtain the current wok by using **Mercurial**. This is called *cloning* the Wok, a procedure that downloads the entire Wok and all its history to a working directory. **If you wish to only use Tazwok to build packages for personal use, this is not necessary.** See the *Creating a Personal Wok* (page 501) section below instead.

The usual destination for a Wok clone is `/home/slitaz/wok`:

```
$ hg clone http://hg.slitaz.org/wok/ /home/slitaz/wok
```

This download may take some time; you will have a complete directory structure of the Cooking wok[314] as a working directory.

Important: The Wok is one of many projects hosted in the Mercurial repositories[315]. Individual packages are grouped as a large project (the Wok, Wok-Stable or Wok-Undigest) and is not its own sub-project but merely a sub-directory; Mercurial cannot (yet) clone specific parts of a project thus you cannot clone an individual package.

12.2.2.2 Creating a Personal Wok

If your packages are only for personal use and are not intended for inclusion in the SliTaz repositories, a wok can be created from scratch.

```
# tazwok gen-clean-wok
```

[311] http://hg.slitaz.org/tazwok/raw-file/tip/doc/tazwok.en.html
[312] http://mercurial.selenic.com/
[313] http://hgbook.red-bean.com/
[314] http://hg.slitaz.org/wok/
[315] http://hg.slitaz.org

12.2.3 Compiling and Generating Packages

Before compiling your first package, **Tazwok** must know where your working directory is. By default the path is /home/slitaz/wok but you can change this or rename the wok that you want to download. To view and check **Tazwok** paths that will be used, and the number of packages in the wok, you can ask **Tazwok** for statistics:

```
# tazwok stats
```

The process for generating a SliTaz package from source can be summarised thus: configure[316], compile[317] & strip[318].

Note: We do not carry out the '**make install**'-style step ourselves; the built files are not to be installed in the system but left in the output directory (_pkg), ready for packaging.

When generating your first package, it is advisable to keep it simple[319] and build your package without changing its receipt or seeking dependencies. **M4** is an ideal candidate for your first *cook*:

```
# tazwok cook m4
```

When **Tazwok** has finished building **M4**, its package is placed in the directory specified by the configuration file (/home/slitaz/packages by default). If all went well, you can install the package on the host system or use it to generate a LiveCD distribution via **Tazlito**!

When you are familiar with *receipts* (page 495) and the compilation process, you can use the following command to create a new package (and a wok, if you don't have one) before interactively writing its receipt:

```
# tazwok new-tree <packageName> --interactive
```

Be sure to read the documentation on the options provided by the *receipt* (page 495) and the *Tazwok Tips* (page 240) to avoid frustration!

12.2.3.1 Cooking Multiple Packages with cook-list

Tazwok can compile several packages with a single command. This is achieved with a *cooking list*, a text file of one package per line. **Tazwok** can accept a cook-list with the command of the same name; for example, to cook the *mypkgs* cook-list:

```
# tazwok cook-list mypkgs.cooklist
```

Tip: There are example lists in /usr/share/examples/tazwok/cooklists.

[316] http://www.tuxfiles.org/linuxhelp/softinstall.html#s2
[317] http://www.tuxfiles.org/linuxhelp/softinstall.html#s3
[318] http://linux.die.net/man/1/strip
[319] http://doc.slitaz.org/en:cookbook:devcorner#kiss-comply-to-standards

12.2.4 Package Compilation Options

While you are free to use any options you want, it is necessary to respect the FSH, the documentation in `/usr/share/doc` and follow the FreeDesktop standards (.desktop).

12.2.4.1 Package-Specific

Package-specific options are your choice; for example, you can disable support for XML, have smaller binaries for **PHP** and get rid of **libxml2**, but in the case of **PHP**, it's not worth the cost in terms of loss of functionality. If you have any doubts, look at the receipts and compiler options in `compile_rules`.

12.2.4.2 Optimization

The official SliTaz packages are optimized for **i486**, the optimization arguments used to configure are specified in `/etc/tazwok.conf` and can be called via the variable `$CONFIGURE_ARGS`. If you want to compile a package with different arguments, you can modify the **Tazwok** configuration file:

```
CONFIGURE_ARGS="--build=i486-pc-linux-gnu --host=i486-pc-linux-gnu"
```

12.2.4.3 Files to Include/Exclude

Generally, the base packages contain no `man`, `info` or `doc` files, nor static libraries; we have to create them via a package-doc or a package-dev. Note that SliTaz does not intend to use the **man** or **info** command so there's no manual or GNU info file. The creation of packages containing docs is really optional. By contrast, writing documentation in the Handbook is more appreciated as it is widely-available and can be updated and improved easily.

In terms of configuration, the aim is to offer basic configuration files to run the package directly. Special cases exist such as the web server **LightTPD**, for example, where SliTaz supplies configuration files and start-up scripts in `/etc/init.d` (documented in the Handbook). For a new package, you are free to choose its default configuration depending on what you think is best for the end-user. The `/usr/share/examples` directory has example configurations and other kinds of useful information.

12.2.5 Package Categories

The categories of packages exist only for informational purposes and are not fixed. The idea is to classify packages so that a web page that recovers data in the package receipt, can be generated each night. For the short term, place development packages in 'devel', Xorg in 'x-window' and the variety of new packages in 'extra'.

12.2.6 Structure of a Wok Package

The structure of the packages in the wok should always be respected so that **Tazwok** can find the correct files and directories. Possible contents of a package (note the directory `taz/` is created at time of cooking):

stuff/: The material used to configure, compile and generate the package (patch(es), Makefile, pseudo fs, etc);

`receipt`: The ever-present *receipt* (page 495);

`description.txt` (optional): The description of the package is included in the final package, copied to its root. Once installed, **`Tazpkg`** identifies this file as the description and can display it via **`tazpkg desc pkgname`**.

`taz/`: Directory tree containing the package **`Tazpkg`** generated, the compressed package is stored in the directory specified by `$PACKAGES_REPOSITORY` in the **`Tazwok`** configuration file.

`Tazwok` will automatically call upon **`Tazpkg`** to package the `taz` directory. It also forwards any packaging instructions found in the receipt.

12.2.7 Structure of a Tazpkg

The SliTaz packages are cpio archives containing files and a file-system compressed with gzip:

`fs/`: Pseudo-file-system containing all the files to install.

`receipt`: The *receipt* (page 495).

`files.list`: A list of files in the package.

`description.txt`: The description of the package (optional).

CHAPTER 13

Developers Corner

author bellard, linea

This page aims to list SliTaz tricks that you can't find in other distributions.

13.1 ISO9660 image file

The SliTaz GNU/Linux distrbution is published as a downloadable iso image file. The different flavors (base, core64, loram…) are built in this format.

Nowadays booting on a CD-ROM is not so fashionable. People prefer to boot their SliTaz from a USB key, a memory card or a hard disk.

It's easy to create a USB key under Linux with the dd[320] command. Some third party Windows tools aim to be able to create USB keys, but most of them don't support the SliTaz *many-in-one* (page 133) format (also known as *russian dolls*).

Tip: SliTaz ISO images are also a DOS/Windows program to create USB keys.

```
C:\> ren slitaz.iso mkusbkey.exe
```

People would like to add some personal data to the ISO image such as a *Wifi configuration* or *SSH keys*. But the ISO remastering is not an easy thing to do.

SliTaz provides a linux tool iso2exe[321] and the DOS/Windows tool isohybrid.exe[322] to add a custom initrd and some extra cmdline arguments.

Note: By the way, these tools add the DOS/Windows program to create USB keys.

[320] http://en.wikipedia.org/wiki/dd_(Unix)
[321] http://cook.slitaz.org/cooker.cgi?download=../wok/syslinux-extra/taz/syslinux-extra-4.06/fs/usr/bin/iso2exe
[322] http://mirror.slitaz.org/boot/isohybrid.exe

A Linux usage can be:

```
iso2exe -a "rdinit=init.custom" -i initrd.gz slitaz.iso -f
```

A DOS (16 bits) or Windows (32 bits) usage can be:

```
C:\> isohybrid -a "rdinit=init.custom" -i initrd.gz slitaz.iso -f
```

Tip: The executable shell script `/init.custom` in the `initrd.gz` file installs the extra software in the boot scripts:

```
#!/bin/sh

# Add custom kernel modules
sed -i 's/LOAD_MODULES="/&amodule anothermodule/' /etc/rcS.conf

# Start extra daemons
sed -i 's/RUN_DAEMONS="/&demon1 demon2/' /etc/rcS.conf

# Custom boot commands
cat >> /etc/init.d/local.sh <<EOT
shell commands...
EOT

# Continue normal boot sequence
exec /init
```

An iso9660 image file has three parts:

- a 32Kb header filled with zeros
- a ISO 9660[323] filesystem
- a tail filled with zeros to round up the file size to the next megabyte

13.1.1 The ISO header: to create a USB key from DOS/Windows

This is a 3 in 1 header:

- a Master boot record[324] from isolinux[325] to boot from a USB key
- a 16 bits DOS .exe[326] file to launch a Linux utility menu with a USB creation item
- a 32 bits Windows .exe[327] file to create a USB key

Note: The El Torito[328] boot is untouched (as a part of the ISO9660 filesystem)

[323] http://en.wikipedia.org/wiki/ISO_9660
[324] http://en.wikipedia.org/wiki/Master_boot_record
[325] http://en.wikipedia.org/wiki/isolinux
[326] http://en.wikipedia.org/wiki/.exe
[327] http://en.wikipedia.org/wiki/.exe
[328] http://en.wikipedia.org/wiki/El_Torito_(CD-ROM_standard)

It includes a iso9660 filesystem md5sum[329] hash and its own checksum (in the .exe[330] file header). Design rational can be found in the README file[331]

The taziso[332] tool can both show and use the ISO header features.

Tip: A taziso graphical/web interface is available in the tazpanel utility under the *boot* → *mine* menu item.

Tip: Most of the ISO header features can be used with other live Linux distributions such as Tiny Core Linux[333], Puppy Linux[334] or KNOPPIX[335] (i.e. you can use **`iso2exe`** or **`isohybrid.exe`** with these distributions too).

13.1.2 The ISO tail: to store your configurations

The ISO tail has a magic string and its own md5 hash. The md5 can be checked by taziso and the ISO boot menu under DOS. The ISO9660 filesystem is untouched. The md5 hash in the boot area is still valid.

The **`isolinux`** bootloader is modified by SliTaz to load the custom configuration unlike other distributions.

Small custom configurations should not change the ISO image size thanks to the megabyte alignment. Larger configurations can extend the ISO image without limitation.

Tip: Custom configurations help to test the SliTaz weekly build[336] with automatic wifi setups or SSH keys... You can add the following entry in your grub config, have a custom automatic setup and still test the real CD-ROM boot sequence.

```
SliTaz rolling iso
   map --mem --heads=0 --sectors-per-tracks=0 /boot/slitaz-rolling.iso␣
↪(hd32)
   map --hook
   chainloader (hd32)
```

13.2 Packages enhancements

[329] http://en.wikipedia.org/wiki/md5sum
[330] http://en.wikipedia.org/wiki/.exe
[331] http://hg.slitaz.org/wok/file/tip/syslinux/stuff/iso2exe/README#l1
[332] http://cook.slitaz.org/cooker.cgi?download=../wok/syslinux-extra/taz/syslinux-extra-4.06/fs/usr/bin/taziso
[333] http://en.wikipedia.org/wiki/Tiny_Core_Linux
[334] http://en.wikipedia.org/wiki/Puppy_Linux
[335] http://en.wikipedia.org/wiki/KNOPPIX
[336] http://mirror.slitaz.org/iso/rolling/slitaz-rolling.iso

dropbear

`Dropbear` is a SSH2 client and server. The server side supports X11 forwarding but the client does not. SliTaz provides a tiny shell script named sshx[337] to restore this feature. By the way 2 other scripts are given:

- **`pppssh`** a poor mans VPN. TCP based, it can add delays to the VPN network.
- **`sshfbvnc`** adds authentication and encryption to the **`fbvnc`** viewer.

sshfs-fuse

`Sshfs-fuse` can mount a remote filesystem on the local machine. SliTaz provides a tiny script named rsshfs[338] to mount a local filesystem on a remote machine.

cloop

Cloop packages (**`fusecloop`**, **`cloop-utils`**...) are able to mount **any** cloop formats. The official software fails to mount earlier formats.

They add a new format[339] to reduce the memory consumption during the compressed file creation.

ipxe

This PXE has a built in configuration[340] to boot from a SliTaz server and can be used without a local PXE server.

sane-backends

The scanner drivers package adds a GUI in a tazpanel module[341].

[337] http://hg.slitaz.org/wok/file/tip/dropbear/stuff/sshx#l1
[338] http://hg.slitaz.org/wok/file/tip/sshfs-fuse/stuff/rsshfs#l1
[339] http://hg.slitaz.org/wok/file/tip/fusecloop/description.txt#l1
[340] http://hg.slitaz.org/wok/file/tip/ipxe/stuff/ipxe.cmd#l1
[341] http://hg.slitaz.org/wok/file/tip/sane-backends/stuff/tazpanel/sane.cgi#l1