

Slackware Installation

My Slackware installation notes. Some old notes have been [archived](#).

Do note that Slackware also has a great [documentation site](#).

Getting Slackware

The official way to do this is, of course, to get it from [slackware.com](#).

Personally, I have [getslack](#), a bash script based on (more accurately, a trimmed-down version of) the excellent (he termed it *infamous*) [mirror-slackware-current.sh](#) by [Alien Bob](#). When going down this path, the next step would be to prepare the installation media.

[Slackware Installer ISO Image](#)

Slackware Installer ISO Image

I no longer need an ISO image (refer to USB installer below). But, I have [slack2iso](#) script (also based on Alien Bob's script) can help in creating one using the tree downloaded by [getslack](#).

[Slackware USB Installer](#)

Slackware USB Installer

[Alien Bob](#) has provided a [script](#) to make/setup/configure a USB-based Slackware installation media. I wanted to do something simpler using the existing files in the Slackware tree that I mirrored using [getslack](#) (mentioned above). So, here is how I got that working.

1. Create a FAT32 partition
 - use `fdisk` and make sure it is bootable (bootable flag enabled)
 - use `mkdosfs` (e.g. `mkdosfs -F 32 /dev/sdb1`)
2. Use `syslinux` to provide bootloader
 - create a `/linux/boot/syslinux` folder on the USB
 - type

```
syslinux -d /linux/boot/syslinux /dev/sdb1
```

Note: On newer `syslinux`, use `-i` to indicate new installation

- a file `ldlinux.sys` should appear in `/linux/boot/syslinux`
3. Copy boot facilities from Slackware tree to the media
 - copy a kernel from slackware tree to `/linux/boot` (I used `huge.s`)
 - copy `initrd.img` and `message.txt` to `/linux/boot`
 - copy `isolinux.cfg` to `/linux/boot/syslinux` as `syslinux.cfg`
 - edit `syslinux.cfg` accordingly (initrd, kernel params, etc.)

4. Copy slackware<64> in the Slackware tree (I used a shorter folder name like slack on the USB)

And... we're done! Now we have a simple Slackware USB Installer and install it on every computer we

can get our hands on! 

Note: When running lilo, the installer will detect the USB FAT32 partition as a Windows partition and create an entry for it in /etc/lilo.conf. What I did was just chroot into /mnt, edit /etc/lilo.conf accordingly and re-run lilo. **OR** you can just do this AFTER you've logged into the system - just don't select that Windows partition at lilo boot menu.

[Note: GPT Disks and EFI](#)

Note: GPT Disks and EFI

Things moving to (U)EFI and GPT... slowly leaving legacy BIOS and MBR.

Instead of MBR, we use GPT partitioning scheme:

- supports bigger disk
- supports EFI booting (easier to maintain actually :p)

Partition codes are 2-bytes instead (only 1-byte on MBR's partition table). Among the common ones:

- EF00 (EFI System Partition): this is what EFI boot look for
 - format FAT32

```
mkdosfs -F 32 -n MY1EFI /dev/sdxx
```

- 0700 (MS Basic Data): Windows Partition
 - format NTFS

```
mkntfs -f -L MY1WIN /dev/sdxx
```

- 8300 (Linux filesystem): Linux Partition
 - format EXT4

```
mkfs.ext4 -L MY1LIN /dev/sdxx
```

Once boot using EFI, efibootmgr tool can be used (available on Slackware 14.2)

- to create an entry labelled Slackware with loader file named \efi\slackware\elilo.efi located on first partition of first disk (/dev/sda1)

```
efibootmgr -c -d /dev/sda -p 1 -L "Slackware" -l "\efi\slackware\elilo.efi"
```

- to delete an entry xxxx (bootnum)

```
efibootmgr -b xxxx -B
```

- to re-order boot sequence

```
efibootmgr -o xxxx,yyyy,zzzz
```

Using Slackware-current

Note: I have removed a section on DE-less installation since my current slackware installations ARE, in fact, DE-less.

Note: I have also removed a section on hijacking other Linux system - this, here, turned out to be VERY similar to what needed to be done.

[LastUpdated20210620]

I need to use GTK3 version that is newer than the one on 14.2, so I tried the development version (**slackware64-current**). I have done the same once (pre-11), so I am aware that there can be some issues when doing this. I am happy to say that I AM writing this on a slackware64-current (15.0 beta?) installation on my laptop.

So, this is a little note to my future self (or anybody that may find this useful **DISCLAIMER: Use this at your own risk!**). I am doing this while still using Devuan and I want to keep that for backup, in case things go wrong. (On a side note, the reason I use Devuan was because of the GTK3 version.) So, I have an extra partition that I have reformatted and prepared to download the stuffs I need.

- download official packages (getslack)
 - create download path: <mount-path>/home/share/slackware
 - create custom getslack config .getslack
 - set VERS=current
 - exclude kde & xfce
- setup EFI boot
 - bzImage in kernels/huge.s (rename to vmlinuz)
 - initrd.img in isolinux/ (this has the slackware setup)
- boot and run installation as usual
 - DO NOT format partition (packages are there!)
 - pick packages from mounted path
 - manually set kernel to boot (I use huge - generic needs initramfs)
- boot newly installed slackware
 - remove gnuchess and xaos packages
 - make sure vim does not create backups (edit /usr/share/vim/vimrc)
 - allow dmesg for user
 - edit /etc/rc.d/rc.local ← echo 0 > /proc/sys/kernel/dmesg_restrict
 - just for personal reference, some useful info on using nmcli

```
nmcli r[adio] wifi
nmcli r[adio] wifi on

nmcli d[evice] wifi list
nmcli d[evice] wifi connect <ssid> password <pass> ifname <wlan0>
```

```
nmcli connection show  
nmcli connection down <ssid>  
nmcli connection up <ssid>
```

- customize /etc/xdg/user-dirs.defaults (standard default paths)
- create user

- **things to note!**

- elogind conflicts with my acpi scripts (lid, to be exact)
 - edit /etc/elogind/logind.conf
 - insert HandleLidSwitch=ignore
- pm-utils not available
 - use loginctl instead (e.g. loginctl suspend instead of pm-suspend)
 - on the plus side, does not need root
- to lock xsession when suspending, create sleep-hook (script)
 - e.g. edit /lib64/elogind/system-sleep/xlock

xlock

```
#!/bin/bash
case "$1" in
    pre)
        upid=$(pgrep dbus-launch)
        user=$(ps -o user --no-headers $upid)
        export XAUTHORITY=/home/$user/.Xauthority
        export DISPLAY=:0.0
        su $user -c 'xlock -mode matrix' &
        sleep 1
    ;;
esac
```

- make it executable (chmod a+x)
- get additional packages (getslackpack)
 - luckily, alienBob's repo 'supports' current
 - create custom getslackpack config .getslackpack
 - (alien) openjdk libreoffice libreoffice-dict-en"
 - *note: ffmpeg already on current!*
- get additional packages (getslackbuild)
 - run as VERS=14.2 getslackbuild fetch <pkg>
 - pkgs: dmenu geany rox-filer slackware-xdm-theme
 - pkgs: slock st wmmname pmount unrar
 - pkgs: nss-mdns avahi libdaemon
 - *note: rox-filer cannot be compiled, needed patching ([this](#))*
 - i have gathered all the scripts from slackbuilds.org that i use and keep them [here](#)
- i want to use dwm
 - using my own custom [build script](#) (which has personalized patches)
 - my dwm xinitrc will run loginctl hibernate when battery<30% (→ what i need on my current laptop)

To maintain:

note: my libmy1slack library will detect current when /etc/slackware-version has '+' suffix. this sign will disappear when -current is near to a stable release.

- run [slack-update](#) as usual
 - when -current going stable, use SLACKVERS=current slack-update
- run [slack-current](#) instead of slackpatch
 - when -current going stable, use -f switch
 - to see removed packages, use [slackview](#) (i.e. SLACKVERS=current slackview find -alien)
- update those installed using getslackbuild if needed

Lean Installation

This is what I do for a lean (not necessarily minimal, but trimmed to my liking) installation. Will continuously update this based on my latest experience.

LastUpdated20230322

- official packages (getslack)
 - without kde & xfce (checkout my getslack config file below)
 - removepkg: gnuchess, xaos, xsnow
 - removepkg: joe, nano, vim-gvim, slackpkg
- setup/config
 - make sure vim does not create backups (edit /usr/share/vim/vimrc)
 - or, run vimstart (from my1shell repo)
 - dmesg no longer allowed for user
 - edit /etc/rc.d/rc.local ← echo 0 > /proc/sys/kernel/dmesg_restrict
 - or, run [setup_slack](#) (from my1shell repo)
 - setup acpi from my personal script
- additional packages (getslackpack)
 - (alien) openjdk libreoffice libreoffice-dict-en"
- additional packages (getslackbuild)
 - dmenu slock st wmmname
 - slackware-xdm-theme
 - geany rox-filer pmount unrar
 - nss-mdns avahi libdaemon
 - scripts from slackbuilds.org all gathered [here](#)
- custom [dwm](#) build
 - using my own [build script](#) (which has personalized patches)

To maintain:

- run [slack-update](#)
 - this actually runs 3 scripts (getslack, getslackpack, getslackbuild)
- run [slackpatch](#) (if required)
- run [getslackbuild build -x -i](#) (if required)

[Configuration file for getslack](#)

dot-getslack

```
# repo selection
RSYNCURLROOT="slackware.mirrors.tds.net::slackware/"
#RSYNCURLROOT="mirrors.kernel.org::slackware/"
#RSYNCURLROOT="rsync.osuosl.org::slackware/"
#RSYNCURLROOT="bear.alienbase.nl::mirrors/slackware/"
# allow VERS to be overridden at command line
VERS=${VERS:="current"}
# include info for extra packages
INCLUDES="$INCLUDES --include=extra/ --include=extra/PACKAGES.TXT"
# exclude non-essentials
EXCLUDES="$EXCLUDES --exclude=extra/*"
# exclude non-standard
EXCLUDES="$EXCLUDES --exclude=pasture/ --exclude=testing/"
# exclude extra installers
EXCLUDES="$EXCLUDES --exclude=usb-and-pxe-installers/"
# exclude source
EXCLUDES="$EXCLUDES --exclude=source/"
# exclude unwanted packages
EXCLUDES="$EXCLUDES --exclude=slackware*/e/ --exclude=slackware*/f/"
EXCLUDES="$EXCLUDES --exclude=slackware*/kde/ --exclude=slackware*/y/"
# exclude kde
EXCLUDES="$EXCLUDES --exclude=slackware*/kde/"
# exclude xfce
EXCLUDES="$EXCLUDES --exclude=slackware*/xfce/"
```

Configuration file for getslackpack

dot-getslackpack

```
# always execute
OPT_CHECK="NO"
# always sweep old packages
OPT_SWEEP="YES"
# allow VERS to be overridden at command line
VERS=${VERS:="current"}
# bare necessities
THIS_CONF="--alien openjdk"
# office suite
THIS_CONF="$THIS_CONF --alien libreoffice libreoffice-dict-en"
# running java applet?
#THIS_CONF="$THIS_CONF --alien icedtea-web rhino"
# java build tool?
#THIS_CONF="$THIS_CONF --alien apache-ant"
# chromium stuff
#THIS_CONF="$THIS_CONF --alien chromium"
#THIS_CONF="$THIS_CONF chromium-pepperflash-plugin"
```

```
#THIS_CONF="$THIS_CONF chromium-widevine-plugin"
# miscellaneous
#THIS_CONF="$THIS_CONF --alien p7zip xdialog"
# torrent-stuff
#THIS_CONF="$THIS_CONF --alien qbittorrent libtorrent-rasterbar"
# my video-tool need this
THIS_CONF="$THIS_CONF --alien ffmpeg"
# requirements for simple screen recorder?
#THIS_CONF="$THIS_CONF --alien ffmpeg jack opus"
# nice pdf reader
THIS_CONF="$THIS_CONF --rwork evince"
# dvd/cd burning on xfce
#THIS_CONF="$THIS_CONF --rwork xfburn libburn libisofs"
```

Configuration file for slackpatch

dot-slackpatch

```
# check these extra package paths as well?
#PKG_EXTPATH="${SLACKRELPATH}-multilib"
# ignoring these standard packages => multilib!
#PKG_IGNORED="glibc glibc-i18n glibc-profile glibc-solibs"
#PKG_IGNORED="$PKG_IGNORED gcc gcc-g++ gcc-gfortran"
#PKG_IGNORED="$PKG_IGNORED gcc-gnat gcc-go gcc-java gcc-objc"
```

Listing for myllive-root...

Core list for Slackware-15.0 (Current)

00core.list

```
#core_system
a!aaa_base
a!aaa_glibc-solibs
a!aaa_libraries
a!aaa_terminfo
a!bash
a!bin
a!coreutils
a!devs
a!e2fsprogs
a!elvis
a!etc
a!eudev
a!grep
```

```
a!kernel-modules
a!kmod
a!libgudev
a!procps-ng
a!sed
a!shadow
a!sysklogd
a!sysvinit
a!sysvinit-scripts
a!tar
a!util-linux
a!xz
```

chroot Installation

20110621 I want to have a 32-bit system running in chroot environment on my Slackware64. I've used such system on Debian using schroot...

20110906 I managed to do this as published [here](#)...

20120518 Minor change to the fstab entry for dev, which needs an rbind option so that the pty inside can be valid! Discussed [here](#).

20120524 This is now part of my slackstuff collection (now known as [my1shell](#))... in form of a script called slackroot.

20121031 The path to the chroot installation MUST ALL BE owned by root - or else, users will get a Write failed: Broken pipe error.

TODO A how-to on creating 32-bit chroot on 64-bit Slackware using slackroot script.

[slack_chroot32.txt](#)

```
- on my pure slack64 (maintained using getsslack/getslackpack)
$ ARCH=i686 getsslack

- create root filesystem using 32-bit packages
# slackroot /opt/chroot32 --arch x86 --desk -x

- copy user/group info from 64-bit system to chroot32
  = will maintain its own login info!
# preroot --init /opt/chroot32

- mount bind 'system' paths
# preroot /opt/chroot32

- ssh into system to use 32-bit chroot
# ssh user@127.0.0.1
```

```
- unmount bind 'system' paths
# preroot --done /opt/chroot32
```

Extra Notes

Some things to note...

20210620 Slackware's CPU frequency scaling works (checkout /etc/rc.d/rc.cpufreq) - just to remind myself, no need to look into this!

Admin-stuff for non-root user

To allow non-root users basic admin (poweroff,reboot,etc.), insert the following to /etc/sudoers (obviously, through sudo). **Note:** Actually, only to users in group power.

```
%power ALL=(ALL) NOPASSWD:/sbin/poweroff
%power ALL=(ALL) NOPASSWD:/sbin/reboot
%power ALL=(ALL) NOPASSWD:/usr/sbin/pm-suspend
%power ALL=(ALL) NOPASSWD:/usr/sbin/pm-hibernate
%power ALL=(ALL) NOPASSWD:/usr/sbin/pm-powersave
```

Note: The pm-* binaries (pm-utils) are no longer available on Slackware 15.0

Multicast DNS @ Zeroconf

- install nss-mdns (requires avahi (requires libdaemon))
- save nss config
 - mv /etc/nsswitch.conf /etc/nsswitch.conf-orig
- use provided nss config
 - cp /etc/nsswitch.conf-mdns /etc/nsswitch.conf
- run daemon at startup

```
script="/etc/rc.d/rc.avahidaemon"
[ -x "$script" ] && $script start
script="/etc/rc.d/rc.avahidnsconfd"
[ -x "$script" ] && $script start
```

Multilib Setup

Main references are [here](#) and [here](#).

1. Basically, need to have C library and compiler capable of multilib. I use getslackpack to

download required packages from [Eric's multilib site](#). Install as instructed.

2. I already have a 32-bit Slackware tree downloaded using `getslack` which I use for my 32-bit chroot installation. I use `massconvert32.sh` script on this tree. The `massconvert32.sh` script can be used to update as well (built packages are not rebuilt). Install as instructed.
3. My `slackpatch` script has been updated to handle 'blacklisted' 64-bit versions and 'upgraded' `compat32` packages

Update20180903

Read [here](#). I now have a more specific script to get multilib stuff (previously part of `getslackpack` script),

- use `getslack-multilib` to download alien_bob's multilib stuff
 - `compat32-tools` `glibc(&friends)` `gcc(&friends)`
 - `compat32` library packages
- upgrade pure-64 `glibc/gcc` packages counterparts
 - `upgradepkg -reinstall -install-new *.t?z`
- install `compat32-tools`
 - some useful helper scripts
- install 32-bit layer support libraries
 - `upgradepkg -install-new slackware64-compat32/*-compat32/*.t?z`
 - obviously, can be used to upgrade as well

Upgrading 14.1 to 14.2

A bash shell script I used to upgrade from 14.1 to 14.2

[upgrade_14-1_to_14-2.sh](#)

```
#!/bin/bash
# - upgrade 14.1 to 14.2
SLACKVERS="14.2"

# setup path to slackware tree
SLACKROOT=${SLACKROOT:=$(pwd)}
[ -z "$SLACKARCH" ] && [ -n "$ARCH" ] && SLACKARCH=$ARCH
SLACKARCH=${SLACKARCH:=$(uname -m)}
SLACKSUFX=${SLACKSUFX=""}
[ "$SLACKARCH" == "x86_64" ] && SLACKSUFX="64"
SLACKFULL=${SLACKFULL:="slackware${SLACKSUFX}"}
[ -z "$SLACKVERS" ] && [ -n "$RELEASE" ] && SLACKVERS=$RELEASE
SLACKVERS=${SLACKVERS:="current"}
SLACKRELS=${SLACKFULL}-$SLACKVERS
SLACKPATH=${SLACKROOT}/${SLACKRELS}

# step 1
upgradepkg ${SLACKPATH}/${SLACKFULL}/a/glibc-solibs-* .txz

# step 2
```

```

upgradepkg ${SLACKPATH}/${SLACKFULL}/a/pkgtools-*.txz
upgradepkg ${SLACKPATH}/${SLACKFULL}/a/tar-*.txz
upgradepkg ${SLACKPATH}/${SLACKFULL}/a/xz-*.txz
upgradepkg ${SLACKPATH}/${SLACKFULL}/a/findutils-*.txz

# step 3
for dir in a ap d k l n t tcl x xap xfce ; do
    do_path=${SLACKPATH}/${SLACKFULL}/$dir
    [ ! -d $do_path ] && continue
    ( cd $do_path ; upgradepkg --install-new *.t?z )
done

# step 4
removepkg ConsoleKit apmd bluez-hcidump cxxlibs foomatic-filters \
gnome-icon-theme imlib kdeadmin kdenetwork kdesdk kdetoys kwallet \
lesstif libelf libjpeg libxfcegui4 networkmanagement obex-data-server \
\
obexfs open-cobol oxygen-gtk3 phonon-mplayer phonon-xine pil portmap \
\
procps qca-cyrus-sasl qca-gnupg qca-openssl udev xchat xf86-input-aiptek \
\
xf86-video-modesetting xfce4-mixer xfce4-volumed xfwm4-themes

# step 5
# - run chknew

# step 7
# - check boot config

```

Building Custom Kernel

- run shell script ([getlinux](#))
- select version, download source
- extract at /usr/src/ (e.g. linux-4.4.199)
- copy a config from /boot

```
# cp /boot/config-generic? > .config
```

- use that config

```
# make oldconfig
```

- configure build

```
# make menuconfig
```

- build the kernel

```
# make -j4 bzImage
```

- build/install modules

```
# make -j4 modules && make modules_install
```

- modules_install requires root, obviously!

- copy (as root) kernel

```
cp arch/x86/boot/bzImage /boot/vmlinuz-generic-4.14.12
cp System.map /boot/System.map-generic-4.14.12
cp .config /boot/config-generic-4.14.12
```

- generate initrd if using generic

```
mkinitrd -c -k 4.14.12 -f ext4 -r /dev/sda3 -m ext4 -u -o
/boot/initrd.gz
```

- a useful initrd generator script IS available
- run /usr/share/mkinitrd/mkinitrd_command_generator.sh
- then run the generated/recommended command
- checkout the -P option if required

Listing Packages

Run ls -l output on /var/log/packages

```
# ls -l /var/log/packages
```

Get package full name ONLY

```
# ls -l /var/log/packages | sed 's|^.* \(.*\)$|\1|g'
```

Get package name ONLY

```
# ls -l /var/log/packages | sed -e 's|^.* \(.*\)$|\1|g' -e 's|^\\(.*)-[^-]*-[^-]*-[^-]*$|\1|g'
```

Find package files for installed packages

```
for pkg in `ls /var/log/packages/` ; do
    pkg_file=`find /home/share/slackware/slackware64-14.2/ -name
"${pkg}.txz"`;
    [ -f "$pkg_file" ] && continue ;
    echo "Cannot find file for $pkg" ;
done
```

Find installation log for standard packages

```
for pkg in `find /home/share/slackware/slackware64-14.2/slackware64 -name
"*.txz" | sort` ; do
    base=`basename $pkg` ;
```

```

name=${base%.*} ;
test="/var/log/packages/$name" ;
[ -f "$test" ] && continue ;
echo "Cannot find file for $test" ;
done

```

To list currently installed packages (to be used in my1live)

- get all installed packages

```
# slackview file --name pkgs.txt --installed --insert
```

- sort based on software sets

```
# slackview file --name pkgs.txt --sort
```

- remove those already selected for my1live

```
slackview file --name pkgs.txt --dups curr.list
```

- rename pkgs.txt to my1live list (e.g. XXmore.list)

Slackware/Linux Desktop

Little tips for better Slackware/Linux desktop experience.

Mplayer (gmplayer) runs without video window

Double-clicking in file manager (Thunar) shows no video (audio is ok, so program is running). Running mplayer from command line is fine. Turns out gui version is gmplayer - running from console shows this error message

```
Failed to open VDPAU backend libvdpau_va_gl.so
```

So, just edit `~/.mplayer/gui.conf`, find the line for `vo_driver` and set it to `gl`. Should be ok after that... at least in my case it is.

Login issue

When using xdm, login fails sometimes with error message ... `Unable to establish ICE listener...`

- edit `/etc/rc.d/rc.local` ← `rm -rf /tmp/.ICE-unix/*`

Mplayer and Xscreensaver

Note: **NOT** tested... just found this



Although the mplayer setting to disable xscreensaver has been selected, xscreensaver still runs!

Possible Solution: Edit `~/.mplayer/config`

```
heartbeat-cmd="xscreensaver-command -deactivate >/dev/null 2>&1"
```

Audio issue when using ALSA

The solution is [here](#)

- in short, ALSA using the HDMI output as default
- use alsamixer, [F6] select the card (the one that is NOT HDMI) and make sure output is not muted
- use aplay -l to identify the card/slot# and device#
- use aplay -D plughw:<card/slot#>,<device#> <WAV file> to test
- edit /etc/asound.conf or `~/.asoundrc` to fix this (sample below)

[asound.conf](#)

```
pcm.!default {
    type hw
    card <card/slot#>
}
ctl.!default {
    type hw
    card <card/slot#>
}
```

Low volume when using ALSA

As seen [here](#)...

Check `/etc/asound.conf`

```
pcm.!default {
    type plug
    slave.pcm "softvol"
}
```

```

pcm.softvol {
    type softvol
    slave {
        pcm "dmix"
    }
    control {
        name "Pre-Amp"
        card 0
    }
    min_dB -5.0
    max_dB 20.0
    resolution 6
}

```

Sometimes need type in pcm. !default to be hw

Slackware Application

Setting up application(s) on Slackware... but, most are applicable to all distributions.

'Hidden' Application

There are a couple (a few?) of applications that I thought have great features but relatively unknown (I only knew about them after looking for specific solution). Here they are:

- xfig - nice app to create figures for use with latex
- xpaint - can do screen capture (xpaint -snapshot)

Apache Setup

Editing /etc/httpd/httpd.conf

- change document root to a folder my main user have full access
 - also create a link to it from my user account
- allow php to be indexed ⇒ add to directory index in dir_module
- enable php (towards the end of the file) ⇒ include mod_php.conf
- enable mod_rewrite (API server?) ⇒ LoadModule ... mod_rewrite.so
- optional: set serveradmin email and servername
- optional: to only serve locally, set listen localhost:80 ???

Setting up SSL: (for HTTPS!)

- modify /etc/httpd/httpd.conf:
 - enable loadmodule mod_ssl.so
 - enable loadmodule mod_socache_shmcb.so

- include httpd-ssl.conf
- generate private key:
 - `openssl genrsa -out privkey.pem 2048`
 - rsa private key with 2048-bit long modulus written to file
- generate cert:
 - `openssl req -new -x509 -key privkey.pem -out cacert.pem -days 1095`
- edit /etc/httpd/httpd-ssl.conf
 - accordingly...

mariadb/mysql Setup

- run

```
mysql_install_db
```

- make sure permission given to user *mysql*

```
chown -R mysql:mysql /var/lib/mysql
```

- start daemon

```
sh /etc/rc.d/rc.mysqld start
```

- run

```
/usr/bin/mysql_secure_installation
```

- create specific database for specific app

```
create database app_db;
```

- create specific user for specific app and grant all access

```
grant all privileges on app_db.* to 'user_app'@'localhost' identified  
by 'pass_app';
```

- just formality, run

```
flush privileges;
```

- to additionally create specific user for specific app

```
create user 'user_app'@'localhost' identified by 'pass_app';
```

- recover root password:

```
# /etc/rc.d/rc.mysqld stop
```

```
# mysql_safe --skip-grant-tables &
# mysql -u root
$ mysql -uroot -p
mysql> use mysql;
mysql> update user set password=PASSWORD('<newpass>') where
User='root';
mysql> flush privileges;
mysql> exit
```

- reset auto_increment primary key:

```
alter table mylvehicle AUTO_INCREMENT=1;
```

- remove foreign key from table (start with checking for key name)

```
select table_name,auto_increment from information_schema.tables;
alter table myldata_use drop foreign key myldata_use_ibfk_2;
```

- backup db

```
mysqldump -p -u user userdb > userapp-`date +%Y%m%d%H%M%S`.sql
```

TeXLive Install/Setup

The default tetex is usable, but it is no longer maintained and some new packages are not available.

The recommended TeX distribution is [TeXLive](#).

- Download from [here](#)
 - get [Unix Installer](#)
 - extract path: /home/share/tool/texlive
- Execute

```
./install-tl -gui
```

- install path: /home/share/tool/texlive/YYYY (currently, YYYY=2020)
- select basic, then customize (e.g. lang:en, graphics, math, etc.)
- setup environment variable

```
TL_VERS="2020"
TL_PATH="/home/share/tool/texlive/${TL_VERS}"
export PATH=${TL_PATH}/bin/x86_64-linux:$PATH
export MANPATH=${TL_PATH}/texmf-dist/doc/man:$MANPATH
export INFOPATH=${TL_PATH}/texmf-dist/doc/info:$INFOPATH
```

- setup tlmgr (tlmgr option repository ctan)
- Run manager

```
tlmgr --gui
```

- needs perl-tk
- to get collection of a package

```
tlmgr show <package>
```

2023/08/29 13:04

rsync server

- create /etc/rsyncd.conf

[rsync.conf](#)

```
max connections = 2
log file = /var/log/rsync.log
timeout = 300

[share]
comment = Shared Stuff
path = /home/share
read only = yes
list = yes
hosts allow = 192.168.3.0/24
uid = nobody
gid = nobody
#auth users = pub
#secrets file = /etc/rsyncd.secrets
```

- modify subnet mask address for host allow accordingly
- in /etc/inetd.conf
 - insert this line

```
rsync    stream     tcp      nowait      root      /usr/bin/rsync      rsync
--daemon
```

- in /etc/services
 - insert this line

```
rsync    873/tcp
```

- maybe create /etc/rsyncd.secrets

[rsyncd.secrets](#)

```
pub:pub
```

- start rsync daemon

```
/usr/bin/rsync --daemon --config=/etc/rsyncd.conf
```

Steam on Slack64 (in chroot32)

on an x86_64 machine,

- install libtxc_dxtn package from slackbuilds.org (64-bit)
- use 'slackroot' to create 32-bit chroot environment (chroot32)
 - target for desktop
- ssh into localhost to enter chroot32 as normal user
 - remember to run 'preroot' (as root) prior to that
 - ... and 'preroot -release' when done
- install packages
 - install alien_bob's steamclient package
 - install libtxc_dxtn package from slackbuilds.org (32-bit)
- run 'linux32' to create an official 32-bit environment
- run 'steam -tcp'

Slackware Miscellaneous

Some personal notes...

Libreoffice

- alien_bob's libreoffice cannot run
 - got unspecified application error
- found a fix in lq forum
 - look for graphic driver used

```
$ lspci -vv | sed -n '/VGA compatible/,/^$/ p'
```

- in my case i915
- have a

```
export MESA_LOADER_DRIVER_OVERRIDE=i915
```

From:

<http://azman.unimap.edu.my/dokuwiki/> - **Azman @UniMAP**



Permanent link:

http://azman.unimap.edu.my/dokuwiki/doku.php?id=linux:slack_install&rev=1700712143

Last update: **2023/11/23 12:02**