

Slackware Experience

Personal notes on using [Slackware](#). Some old notes have been [archived](#).

Do note that Slackware 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

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

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: 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
```

2026/02/09 22:23

Installing Slackware

Installation notes (i.e. packages, configs).

LastUpdated20250112

Basic Install

Using Slackware installer.

- official packages (getslack)
 - checkout my getslack [config file](#)
 - without kde (AND xfce if going DE-less)
 - removepkg gnuchess xaos xsnow
 - removepkg joe nano vim-gvim slackpkg
- setup/config
 - sample elilo.conf

[elilo.conf](#)

```
prompt
#chooser=simple
timeout=50
default=Slack
image=vmlinuz-huge
    label=SlackHuge
    read-only
    append="root=/dev/sda2 resume=/dev/sda4 vga=normal"
image=vmlinuz
    label=Slack
    initrd=initrd.gz
    read-only
    append="root=/dev/sda2 resume=/dev/sda4 vga=normal"
```

- make sure vim does not create backups (edit /usr/share/vim/vimrc)
 - or, run vimstart (from mylshell repo)
- dmesg no longer allowed for user
 - append rc.local ← echo 0 > /proc/sys/kernel/dmesg_restrict
 - or, run setup_slack (from mylshell repo)
- additional packages (getslackpack)
 - checkout my getslackpack [config file](#) and [repository list](#)
 - (alien) openjdk libreoffice libreoffice-dict-en
- additional packages (getslackbuild)
 - slackware-xdm-theme
 - geany unrar
 - nss-mdns avahi libdaemon

- actually, scripts from slackbuilds.org ([commonly used](#))

DE-less config

This is what I do for a lean (not necessarily minimal, but trimmed to my liking) installation.

- setup acpi from my personal script
- additional packages (getslackbuild)
 - dmenu slock st wmname
 - rox-filer pmount
- custom [dwm](#) build
 - using my own [build script](#) (which has personalized patches)

Updating

To maintain:

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

Sample configuration files for the above scripts are [here](#).

2026/02/09 22:23

Using Slackware-current

This is actually NOT recommended for beginners. But, sometimes, the need to use the latest software

is unavoidable and this COULD be a solution. Plus, this will add a LOT of COOL-points



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.

Installing

[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
 - append 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 c[onnection] show
nmcli c[onnection] down <ssid>
nmcli c[onnection] up <ssid>
```

- customize etc/xdg/user-dirs.defaults (standard default paths)
- create user
- get additional packages (getslackpack)
 - luckily, alienBob's repo 'supports' current
 - create custom getslackpack config .getslackpack
 - (alien) openjdk libreoffice libreoffice-dict-en"
- get additional packages (getslackbuild)
 - run as VERS=14.2 getslackbuild fetch <pkg>
 - pkgs: dmenu geany rox-filer slackware-xdm-theme
 - pkgs: slock st wmname 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)

Updating

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

2026/02/09 22:23

Special Needs

Sometimes these can be useful.

Slackware in chroot

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 `getslack/getslackpack`)

```

$ ARCH=i686 getslack

- 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!
# preproot --init /opt/chroot32

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

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

- unmount bind 'system' paths
# preproot --done /opt/chroot32

```

2026/02/09 22:23

Slackware Multilib

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 Update20250326

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) [multilib packages]
 - compat32 library packages [32-bit packages]
- upgrade pure-64 glibc/gcc packages counterparts
 - upgradepkg -reinstall -install-new *.t?z
 - *note:* this includes compat32-tools package (helper scripts, noarch)
- install 32-bit layer support libraries
 - upgradepkg -install-new slackware64-compat32/*-compat32/*.t?z
 - obviously, can be used to upgrade as well

Configure slackpatch to check/ignore multilib stuff

[dot-slackpatch](#)

```
PKG_IGNORED="ffmpeg"
# ignoring these standard packages => multilib!
PKG_IGNORED="$PKG_IGNORED aaa_glibc-solibs"
PKG_IGNORED="$PKG_IGNORED gcc gcc-brig gcc-g++ gcc-gdc gcc-gfortran"
PKG_IGNORED="$PKG_IGNORED gcc-gnat gcc-go gcc-objc"
PKG_IGNORED="$PKG_IGNORED glibc glibc-il8n glibc-profile"
```

2026/02/09 22:23

Slackware Upgrade

For reference.

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 kdedadmin kdenetwork kdesdk kdetools kwallet \
    lesstif libelf libjpeg libxfcegui4 networkmanagement obex-data-server \
    obexfs open-cobol oxygen-gtk3 phonon-mplayer phonon-xine pil portmap \
    procs qca-cyrus-sasl qca-gnupg qca-openssl udev xchat xf86-input-aiptek \
    xf86-video-modesetting xfce4-mixer xfce4-volumed xfw4-themes

# step 5
# - run chknew

# step 7
# - check boot config
```

2026/02/09 22:23

Slackware System

Useful system level notes.

Listing Packages

Simply list files in /var/log/packages (path is actually a link to /var/lib/pkgtools/packages/).

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

List full package name (with version) only

```
$ for that in $(ls /var/log/packages/) ; do echo $that ; done | sort
```

List installed package name only

```
$ list=$(ls /var/log/packages/) ; for that in $list ; do that=${that%-*} ;
that=${that%-*} ; that=${that%-*} ; echo $that ; done
```

Find installed packages that are not in Slackware tree

```
$ slackfull=slackware64-15.0 ; slacktree=/home/share/slackware/$slackfull ;
for pkg in $(ls /var/log/packages/) ; do
    pkgf=$(find $slacktree/ -name "${pkg}.txz" ) ;
    [ -f "$pkgf" ] && continue ;
    echo "*** Package '$pkg' is alien!" ;
done
```

This can also be done using slackview script (in my1shell repo)

```
$ slackview find --alien
```

View information on specific package

```
$ slackview find [pkg_name]
```

To list all official packages without DE

```
$ for pack in a d k l n ap t tcl x xap ; do
    slackview file --name pkgs.txt --insert --pack $pack ;
done
```

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

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

To sort packages in *pkgs.txt* based on software sets

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

To remove packages in *dups.txt* that is/are found in *pkgs.txt*

```
$ slackview file --name pkgs.txt --dups dups.txt
```

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

2026/02/09 22:23

Slackware Desktop

Little tips for better Slackware desktop experience. Most of these can be used in other Linux distributions as well.

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`. Show be ok after that... at least in my case it is.

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"
```

Login issue

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

- append rc.local ← `rm -rf /tmp/.ICE-unix/*`

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
- modify asound.conf in etc 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 asound.conf in etc:

```
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

2026/02/09 22:23

Slackware Applications

Running applications (@software) on Slackware... most are applicable to all distributions.

'Hidden' Application

There are a couple of (a few?) 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 (web server) Setup

Modify httpd config file:

- 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` ???

More modifications for https:

- enable loadmodule mod_ssl.so
- enable loadmodule mod_socache_shmcb.so
- 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`
- include httpd-ssl.conf
 - modify 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 `rc.mysqld`
- 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:
- stop daemon `rc.mysqld`
- run

```
# mysqld_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
```

Backup a 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>
```

2026/02/09 22:23

rsync server

- create `rsyncd.conf` in `etc`

[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 `inetd.conf` (`etc`), insert

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

- in `services` (`etc`), insert

```
rsync  873/tcp
```

- create `rsyncd.secrets` in `etc`

[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 'preproot' (as root) prior to that
 - ... and 'preproot -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'

2026/02/09 22:23

Issues

Issues when running Slackware...

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'
```

- e.g. for i915, do

```
export MESA_LOADER_DRIVER_OVERRIDE=i915
```

2026/02/09 22:23

Extra Notes

Some things to note...

20210620 Slackware's CPU frequency scaling works (checkout `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.), add them to power group and insert the following to sudoers.

```
%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

Note20250112: use *loginctl* on Slackware 15.0 - no need to add *sudoers*

Multicast DNS @ Zeroconf

- install *nss-mdns* (requires *avahi*, which requires *libdaemon*)
- save *nss* config (in etc)
 - `mv nsswitch.conf nsswitch.conf-orig`
- use provided *nss* config
 - `cp nsswitch.conf-mdns nsswitch.conf`
- run daemon at startup `rc.avahidaemon` and `rc.avahidnsconfd`

Note20260720: latest install seems to NOT need to `switch@edit nssswitch.conf`

Using kolourpaint in XFCE without KDE

Simply install these packages

[pkgs4_kolourpaint.txt](#)

```
ark
karchive
kauth
kbookmarks
kcodecs
kcompletion
kconfig
kconfigwidgets
kcoreaddons
kcrash
kdbusaddons
kglobalaccel
kguiaddons
ki18n
kiconthemes
kinit
kio
kitemviews
kjobwidgets
kolourpaint
```

```
kservice
ktextwidgets
kwallet
kwidgetsaddons
kwin
kwindowsystem
kxmlgui
libksane
oxygen
solid
sonnet
```

Note: Can use `getslack --reset --slack $(cat pkgs4_kolourpaint.txt)?`

NIS on Slackware

Got this from [here](#).

Slackware-NIS-mini-HOWTO

How to set up a NIS server and configure NIS clients on Slackware Linux
14-Dec-2000

David Cantrell <david@slackware.com>

```
=====
THE NIS SERVER
=====
```

STEP 1: Decide which machine will be the NIS server

This is usually not a very involving process, but does require some thought. You'll want to use a machine that is on most of the time and does not need to reboot into other operating systems. If you have an existing fileserver, that would be a good choice for your NIS server.

STEP 2: Decide on what you want NIS to serve

NIS is fairly complex. It allows you to distribute all kinds of information. Things like usernames, passwords, groups, hostnames, and network services that should be running. The most common use is to use NIS as a common user logon authentication system. The second most common use is to have it share NFS automount maps so that users logging in on a system will have their home directory automounted for them.

In this mini-HOWTO, I will explain how to set up your NIS server and

clients for passwd/group sharing and home directory sharing.

STEP 3: Move home directories to sharing location

In order to make the home directory NFS shares work seamlessly across the NIS server and clients, I move them to /export/home. This is the location they will be mounted on the client, so I make them exist there on the server as well. You might not have /export on your server, so let's make it:

```
mkdir -p /export
```

Now make the home directory location:

```
mkdir /export/home
```

Now move all your home directories over to /export/home. You should leave "ftp" in /home, as that's where Slackware installs it and it is sometimes site specific. These commands should move all home directories to /export/home and leave ftp in /home:

```
mv /home/* /export/home
mv /export/home/ftp /home
```

STEP 4: Edit home directory paths on the server

Since home directories are now in /export/home, you need to edit /etc/passwd on the server to reflect those locations. If you have a user called "david", you will want to change his home directory to /export/home/david. Do this for all users on the system.

STEP 5: Write the automount maps

I usually make two automount maps for the home directories. One is called auto.master and controls how the automounter works. The other is called auto.export and will describe the /export map.

Edit /etc/auto.master and add this to it:

```
#
# auto.master - Master map file for NFS automounter
#

/export auto.export --timeout 60
```

Now edit `/etc/auto.export` and add this to it:

```
#
# auto.export - NFS automounter configuration file
#

# Home directories
home -fstype=nfs <NFS/NIS server>:/export/home
```

Of course, replace `<NFS/NIS server>` with the hostname of your NFS/NIS server.

STEP 6: Edit the NFS exports file on the server

Now we need to make sure that `/export` is shared from the server. Edit `/etc/exports` and make sure this line is present:

```
/export (rw,no_root_squash)
```

Save the file and restart the NFS server daemons:

```
killall -HUP rpc.nfsd rpc.mountd
```

STEP 7: Pick a name for your NIS domain

Each NIS domain needs to have a unique name. Do not use the same name as your DNS domain. This should be something different. I will use "mynis" in this example.

Edit `/etc/defaultdomain` and add the name of your NIS domain:

```
mynis
```

Save and exit. The name of your NIS domain should be the only thing in that file. Once you've done that, set the NIS domain with this command:

```
nisdomainname `cat /etc/defaultdomain`
```

STEP 8: Edit /var/yp/Makefile

The NIS server has a Makefile that controls what maps are served out by NIS. Each time you make a change (such as adding a user), you should go into /var/yp and type "make" to update the NIS information.

In this file we need to edit the all: target and specify the stuff we want to share. Here's what your all: target should look like:

```
all: passwd group hosts rpc services netid protocols netgrp mail \
    shadow auto.master auto.export
```

Now find the line that looks like this:

```
AUTO_LOCAL = $(YPSRCDIR)/auto.local
```

We want to add this line just after that line:

```
AUTO_EXPORT = $(YPSRCDIR)/auto.export
```

Now find the auto.local: target. We need to add this target just after that target:

```
auto.export: $(AUTO_EXPORT) $(YPDIR)/Makefile
@echo "Updating $@"
-@sed -e "/^#/d" -e s/#.*$$// $(AUTO_EXPORT) | $(DBLOAD) \
    -i $(AUTO_EXPORT) -o $(YPMAPDIR)/$@ - $@
-@$(NOPUSH) || $(YPPUSH) -d $(DOMAIN) $@
```

Save the Makefile and exit.

STEP 9: Edit /var/yp/securenets

Edit /var/yp/securenets and change the "0.0.0.0 0.0.0.0" line to something like this:

```
255.255.255.0 192.168.1.0
```

You should specify your network address and submask there, this is only an example. This example restricts NIS access to any machine on the 192.168.1 class C subnet.

STEP 10: Create an /etc/ypserv.conf file

The only line you need in this file is:

```
dns: yes
```

This line tells ypserv to resolve hostnames using a DNS server instead of relying on NIS hostname maps (which you probably aren't using). Save and exit.

STEP 11: Start ypbinding

Start ypbinding:

```
/usr/sbin/ypbinding
```

Use ps to make sure it started and is running properly. You will probably see a master and slave process for it.

STEP 12: Start ypserv

Start ypserv:

```
/usr/sbin/ypserv
```

Use ps to make sure it started and is running properly.

STEP 13: Initialize the NIS maps

We need to initialize the NIS maps with this command:

```
/usr/lib/yp/ypinit -m
```

Fill in the blanks for all machines on your network that will be acting as NIS servers. At this point, you probably want to fire up rpc.yppasswdd to make sure that users on NIS clients can run chfn and passwd.

STEP 14: Done, making sure NIS stuff loads at bootup

Your NIS server is now up and running. You should edit /etc/rc.d/rc.inet2 and make sure the following block is uncommented:

```
# Setting up NIS:
# (NOTE: For detailed information about setting up NIS, see the
documentation
# in /usr/doc/yp-tools, /usr/doc/ypbinding, and /usr/doc/ypserv)
#
```

```
# First, we must set the NIS domainname.  NOTE: this is not
# necessarily the same as your DNS domainname, set in
# /etc/resolv.conf!  The NIS domainname is the name of a domain
# served by your NIS server.

if [ -r /etc/defaultdomain ]; then
    nisdomainname `cat /etc/defaultdomain`
fi

# Then, we start up ypbind.  It will use broadcast to find a server.

if [ -d /var/yp ] ; then
    echo -n " ypbind"
    ${NET}/ypbind
fi

# If you are the NIS master server for the NIS domain, then
# you must run rpc.yppasswdd, which is the RPC server that
# lets users change their passwords.

if [ -x ${NET}/rpc.yppasswdd ]; then
    echo -n " yppasswdd"
    ${NET}/rpc.yppasswdd
fi
```

It's probably adviseable to run rpc.yppasswdd with this line:

```
${NET}/rpc.yppasswdd -e chsh -e chfn
```

This ensures that chfn and chsh work. You also need to add this after the ypbind if block:

```
if [ -x ${NET}/ypserv ]; then
    echo -n " ypserv"
    ${NET}/ypserv
fi
```

The ypserv program needs to run on the NIS server.

```
=====
THE NIS CLIENT
=====
```

```
STEP 1:  Set the NIS domain
-----
```

Edit /etc/defaultdomain just like you did on the server and enter the name of your NIS domain.

STEP 2: Edit /etc/yp.conf

Open /etc/yp.conf in a text editor and make sure this line is present:

```
ypserver <IP or hostname of the NIS server>
```

Specify either the IP address or hostname of your NIS server. If you don't have working hostnames, you must use the IP address here.

STEP 3: Set the NIS domain

Set the NIS domain by typing this command:

```
nisdomainname `cat /etc/defaultdomain`
```

STEP 4: Run ypbind

Start ypbind with this command:

```
/usr/sbin/ypbind
```

STEP 5: Test the NIS connection

See if the NIS server is working with this command:

```
ypcat passwd.byname
```

If you see the /etc/passwd entries from the NIS server, you're good to go. Otherwise, go back to the NIS server section and see if you missed any steps.

STEP 6: Edit /etc/nsswitch.conf

You need to edit /etc/nsswitch.conf and tell it to use NIS for passwd, shadow, and group. Change those entries to:

```
db files nis
```

Change the automount entry to:

```
files nis
```

STEP 7: Enable NIS logins in /etc/passwd and /etc/group

Edit /etc/passwd and add this line to the end:

```
+:::
```

Edit /etc/group and add this line to the end:

```
+:::
```

STEP 8: Create the /export directory for home directory mounting

Make sure you have that directory:

```
mkdir -p /export
```

STEP 9: Create an rc.autofs script

The included rc.autofs script should be moved into /etc/rc.d. Edit rc.local and add this line:

```
/etc/rc.d/rc.autofs
```

STEP 10: Done, enable NIS stuff at bootup

Your NIS client is now configured. Edit /etc/rc.d/rc.inet2 and uncomment this block:

```
# Setting up NIS:
# (NOTE: For detailed information about setting up NIS, see the
documentation
# in /usr/doc/yp-tools, /usr/doc/ypbind, and /usr/doc/ypserv)
#
# First, we must set the NIS domainname. NOTE: this is not
# necessarily the same as your DNS domainname, set in
# /etc/resolv.conf! The NIS domainname is the name of a domain
# served by your NIS server.

if [ -r /etc/defaultdomain ]; then
    nisdomainname `cat /etc/defaultdomain`
fi
```

```
# Then, we start up ypbind. It will use broadcast to find a server.

if [ -d /var/yp ] ; then
    echo -n " ypbind"
    ${NET}/ypbind
fi
```

2026/02/09 22:23

From:

<https://www.azman.my1matrix.cc/> - Azman @MY1

Permanent link:

<https://www.azman.my1matrix.cc/doku.php?id=linux:slackware>Last update: **2026/02/08 03:46**