Slackware 100% Libre

I was never on the 100% Free Software team. I use some proprietary programs, but every year I try to get rid of proprietary software, and guys that have a bad license, that is not free. Slackware is a powerful distribution that comes with a set of software that without internet you are well stocked. And in this set, some proprietary and licensed software that Stallman doesn't like is present.

After searching and wandering for a while I found an alternative to putting Slackware in Debian state. Calm without apt! I'm talking about the free state that Debian comes by default, its repositories default to MAIN, which in turn has only free software.

So how do you do this magic? Simple, let's use Freenix, formerly Freeslack, which is Slackware-based but 100% Free. Freenix has a list of non-free packages taken from the Slackware base. Let's use this base to make our Slackware 100% Libre.

To begin with we will need to move through the Slackware repository lists and add the Freenix repository in the last line. As root we run:

```
echo '
# 100% Libre
https://freenix.net/fxp/freeslack64-14.2/
' >> /etc/slackpkg/mirrors
```

If you already have an older Slackware installation on your hard drive, comment out the old repository and add # in front of it.

After that we will need to change/add the **SLACKKEY** variable present in */etc/slackpkg/slackpkg.conf*, we will find the SLACKKEY variable on line 65. If the old SLACKKEY variable is not commented out, comment. 2 same variables conflict.

```
SLACKKEY="Connie Dobbs (Free eXpansion Pack for Slackware)"
```

After that, we need to update the project's GPG key then update the repository list.

```
slackpkg update gpg && slackpkg update
```

Repository is up and now is the funniest time to uninstall non-free, licensed software that is not well liked by the FSF. As Julio Neves would say, if it's in a line, it's even better. And let's do just that. Let's run the removepkg command with some packages!

```
**removepkg**               \
  getty-ps              \
  lha                   \
  unarj                 \
  amp                   \
  seamonkey-solibs      \
  bluez-firmware        \
  ipw2100               \
  ipw2200               \
```

```
  trn                   \
  zd1211-firmware       \
  font-bh-ttf           \
  font-bh-type1         \
  mozilla-thunderbird   \
  mozilla-firefox       \
  seamonkey             \
  xfractint             \
  xgames                \
  xv                    \
  kernel-firmware       \
  kernel-generic        \
  kernel-huge           \
  kernel-modules        \
  kernel-headers        \
  kernel-source
```

Yes, mozilla is free but allows non-free plugins. Therefore you will be without browser. You can compile and install Icecat, present in Slackbuilds;) Or even stick with Firefox. The decision is yours … Just remove it from the list. But it will not be 100% Free. One note! Icecat needs a powerful processor to build, its dual-core won't do the trick. I am sorry to inform you, if you are from the “Pc Weak” team use a pre-compiled package, search the Slackonly website for this package.

Next step is to install Linux Libre, the Linux kernel that has been removed from all Blobs. Let's do this with slackpkg, now we're pulling from the Freenix repository :)

```
slackpkg install        \
   linux-libre-headers  \
   linux-libre-image    \
   linux-libre-source
```

Step done? Ok, we're on the final stretch, we now need to point our new kernel at lilo.conf. But Linux Libre Generic will not roll, not yet … We'll need to point Huge first, reboot the machine and then generate the modules and finally make the point in lilo.conf.

Open *lilo.conf* and add at the end of the line:

```
image = /boot/vmlinuz-huge-4.4.172-gnu
  root = /dev/sda1
  label = Huge-Libre
  read-only
```

Note that the image = call will change this article in the future! So make the call according to the huge kernel of the system. List it in / boot / and see which file to point to.

Run lilo and restart the machine.

```
lilo
```

Ok, after the machine has been rebooted and if you have NOT given a panic kernel, which is kind of hard if you followed this HOW-TO, it's time to load Kernel Generic. Let's use the script in

/usr/share/mkinitrd.

```
**/usr/share/mkinitrd/mkinitrd_command_generator.sh**
mkinitrd -c -k 4.4.172 -f ext4 -r /dev/sda1 -m
hid-logitech-hidpp:hid-lenovo:hid-microsoft:hid_multitouch:jbd2:mbcache:ext4
-u -o /boot/initrd.gz
```

The output will be for modules that can be loaded on my machine. Yours will be variable. Run the mkinitrd that was generated.

```
mkinitrd -c -k 4.4.172 -f ext4 -r /dev/sda1 -m
hid-logitech-hidpp:hid-lenovo:hid-microsoft:hid_multitouch:jbd2:mbcache:ext4
-u -o /boot/initrd.gz
```

After that make the script call again, but this time pointing the generic kernel at /boot/. Use the *-l* parameter.

```
/usr/share/mkinitrd/mkinitrd_command_generator.sh \
-l /boot/vmlinuz-generic-4.4.172-gnu
```

The output will be with a configuration for lilo.conf, add in lilo.conf the output generated, then run lilo to get the new calls from the menu.

```
image = /boot/vmlinuz-generic-4.4.172-gnu
  initrd = /boot/initrd.gz
  root = /dev/sda1
  label = Generic-Libre
  read-only
```

```
lilo
```

Sources

- Original source: http://slackjeff.com.br/artigos/slackware_free.html
- Originally written by Slackjeff

howtos, slackware, libre, 100% libre, author slackjeff