Mise à jour de Slackware

Mise à jour ou Installation à partir de zéro

Installer une version complète de Slackware à partir de zéro est toujours la meilleure solution si vous utilisez une version relativement ancienne de Slackware et que vous voulez sauter quelques versions intermédiaires. Beaucoup trop de changements intrusifs dans la distribution vont arriver si votre version de Slackware est relativement vieille. Cela nécessitera une mise à jour manuelle pénible sans garantie de succès.T.
Il est mieux en pareil cas de faire une sauvegarde de votre listes de paquetages (“`ls -lart /var/log/packages`”), une sauvegarde de votre répertoire “`/etc`” et (vous l'avez déjà fait bien sur) une sauvegarde de vos données personnelles. Formatez votre disque dur et installez Slackware à partir de votre medium de boot (CD, DVD ou disque USB), et une heure plus tard vous serez de nouveau opérationnel.

Si vous voulez mettre à jour vers la nouvelle version de Slackware, vous pouvez le faire manuellement en suivant les instructions du fichier “`UPGRADE.TXT`” que vous trouverez à la racine du CD1/DVD de Slackware. Les instructions avancées et autres astuces peuvent être trouvées dans le fichier “`CHANGES_AND_HINTS.TXT`” au même endroit.Il ya aussi une procédure semi-automatique pour ce type de mise à jour en utilisant slackpkg. Cela vous évitera beaucoup de travail laborieux.
J'utilise *toujours* slackpkg (avec précaution) pour mettre à jour mon système Slackware d'une version stable à la suivante. J'appelle ce processus une *mise à jour* du système.
Vous pouvez utiliser la même procédure pour migrer vers la version courante *slackware-current*, garder un système *slackware-current* à jour, ou mettre à jour un système *slackware-current* vers une version stable fraichement publiée.

Considerations about the Kernel

Just running `slackpkg` and hoping for the best is not going to work. Some considerations have to be cared for. One important thing to remember:

Never upgrade your working kernel.

Why is that? Simple - you will be upgrading potentially hundreds of packages and should be prepared for the unlikely event that your computer does not work properly anymore after a system upgrade. One thing you don't want to get hit by is a system which does not boot at all. A new Slackware release may install a kernel that refuses to boot your computer (small chance but nevertheless... be prepared). For that reason, you need to keep your “old” working kernel installed, and keep a section for it in your `/etc/lilo.conf` file. That way, if the new kernel fails to boot, you can fall-back to the old kernel and start investigating what went wrong.
Basically, these are the same precautions you must take when you are compiling a new kernel yourself.

System Upgrade using SlackPkg

The following steps should work for all situations:

- Blacklist the following kernel packages in "`/etc/slackpkg/blacklist`":

```
kernel-generic
kernel-generic-smp
kernel-huge
kernel-huge-smp
kernel-modules
kernel-modules-smp
```

  This will prevent an accidental upgrade of your working kernel.

- Blacklist packages from 3rd party repositories by adding appropriate lines for their *repo tags*. Examples for SlackBuilds.org, AlienBOB and multilib:

```
[0-9]+_SBo
[0-9]+alien
[0-9]+compat32
```

- If new kernel(s) have been added to the Slackware release you are upgrading to, then use "`installpkg`" to install those new kernel packages first (do not use "`upgradepkg`" because that will wipe your existing kernel).
  You will have to install at least one kernel (kernel-generic, kernel-generic-smp, kernel-huge, or kernel-huge-smp) and the corresponding kernel modules package (kernel-modules or kernel-modules-smp).
  You can not use `slackpkg` for this step.
- Now that we have the new kernel(s) plus modules in place, we can start upgrading the rest of the packages. First, update the `slackpkg` package database:

```
# slackpkg update
```

- When slackpkg has updated its internal database we can let it update the computer to the new Slackware release:

```
# slackpkg install-new
# slackpkg upgrade-all
# slackpkg clean-system
```

  - The first of those three commands (`slackpkg install-new`) will install every package which is marked in the Slackware ChangeLog.txt file with the string "`Added.`" This command will **not** install any other packages which are not yet installed.
    For instance if you did not have KDE installed before, the "`slackpkg install-new`" command will not add KDE packages to your computer all of a sudden.
  - The second command (`slackpkg upgrade-all`) will compare every official Slackware package which you currently have installed, with the package list on your Slackware

mirror. If a different version is available, that version will be (downloaded and) upgraded. [1)]

- The third commmand (`slackpkg clean-system`) will show you an overview of all packages that you currently have installed and are not part of the Slackware Linux release you are upgrading to.

- You should probably decide at this moment to use a generic kernel, especially if you are using LVM or RAID, or installed Slackware to a LUKS-encrypted disk. This is also being recommended in the Slackware README on the DVD/CD. On the other hand - if your system setup is straight-forward and your hardware is fairly new, you could decide to stick with a "huge" kernel. *Remember that you **cannot** use an initial ramdisk in combination with a "huge" kernel, but you **have** to create a new initial ramdisk if you are going to use a generic kernel!*
  If you are uncertain at this point how that must be accomplished, then you should make sure that you are booting a "huge" kernel which won't require an initial ramdisk.
  However, the "`mkinitrd_command_generator.sh`" script can help you here. Run this script with the *new* kernel version as a parameter and it will show you an example "`mkinitrd`" command which will work for your particular hardware setup and system configuration:

```
# /usr/share/mkinitrd/mkinitrd_command_generator.sh -k 3.2.29
```

  will give this as output (the kernel version 3.2.29 is that of Slackware 14)

```
#
# mkinitrd_command_generator.sh revision 1.45
#
# This script will now make a recommendation about the command to use
# in case you require an initrd image to boot a kernel that does not
# have support for your storage or root filesystem built in
# (such as the Slackware 'generic' kernels').
# A suitable 'mkinitrd' command will be:

mkinitrd -c -k 3.2.29 -f ext4 -r /dev/sdb2 -m usb-
storage:pcmcia_core:pcmcia:mmc_core:ssb:modprobe:usbhid:ehci-hcd:ohci-
hcd:mbcache:jbd2:ext4 -u -o /boot/initrd.gz
```

  You can copy and paste this command line in your console, and let it create an initial ramdisk for you.

If you were already running a generic kernel and therefore already have an initrd, we strongly advise you to create a **new** initrd with a **new** unique name! For instance, you can copy the above example and modify the name of the initrd file as follows:

```
mkinitrd -c -k 3.2.29 -f ext4 -r /dev/sdb2 -m usb-
storage:pcmcia_core:pcmcia:mmc_core:ssb:modprobe:usbhid:ehci-
hcd:ohci-hcd:mbcache:jbd2:ext4 -u -o /boot/initrd_3.2.29.gz
```

- After you decided what kernel you are going to use and have created an initial ramdisk, you must update your "`/etc/lilo.conf`" file with a section for the new kernel (*don't remove your*

running kernel!). The “`mkinitrd_command_generator.sh`” script can help you finding the right block to append to `/etc/lilo.conf`. For instance, the command:

```
# /usr/share/mkinitrd/mkinitrd_command_generator.sh -l /boot/vmlinuz-
generic-3.2.29
```

will result in the following output which you can copy/paste:

```
# Linux bootable partition config begins
# initrd created with 'mkinitrd -c -k 3.2.29 -f ext4 -r /dev/sdb2 -m
usb-storage:pcmcia_core:pcmcia:mmc_core:ssb:modprobe:usbhid:ehci-
hcd:ohci-hcd:mbcache:jbd2:ext4 -u -o /boot/initrd.gz'
image = /boot/vmlinuz-generic-3.2.29
  initrd = /boot/initrd.gz
  root = /dev/sdb2
  label = 3.2.29
  read-only
# Linux bootable partition config ends
```

Note that this command adds a “initrd” line to the kernel section. If for your new Slackware release from a mirror, like this:

```
# rsync -av
rsync://taper.alienbase.nl/mirrors/people/alien/multilib/14.0/
multilib-14.0/
```

This command will create a new subdirectory “`multilib-14.0`” in your current directory with all packages inside

- install/upgrade the existing `gcc` and `glibc` packages, and `compat32-tools`:

```
# cd multilib-14.0
# upgradepkg --install-new *.t?z
```

- Upgrade the set of converted 32-bit Slackware packages (often referredto as the “*compat32*” packages):

```
# upgradepkg --install-new slackware64-compat32/*-compat32/*.t?z
```

Alternatively you can run the “`massconvert32.sh`” script which will have been installed as part of the `compat32-tools` package. Pass it a 32-bit Slackware package directory (or a 32-bit Slackware mirror URL) as parameter and that will create a set of converted “`compat32`” packages which you can then install. You would only have to do this if you suspect that the content of the “`slackware64-compat32`” directory is not up to date.

Video driver considerations

If your computer is equipped with a video card powered by an Nvidia or Ati graphics processor and

you have installed these companies' accelerated graphics drivers (closed-source and binary-only), you should not attempt start an X session after upgrading to the next Slackware release.
These drivers depend on kernel version, Mesa version and the X.Org server. You must re-install the binary driver before starting graphical mode. Also, the mesa and xorg-server packages of Slackware overwrite essential files of these closed-source accelerated graphics drivers anyway.

If you want to know how to deal with these binary drivers, we have more detailed instructions in the "Proprietary Graphics Drivers" article on this Wiki.

Sources

- Originally written

1)

Slackware's package manager does not work with the concept of "lower version" and "higher version". The package tools only look at "*different version*" and thus downgrading packages (reverting to an earlier version) is just as easy as upgrading to a newer version!