CSNB113: System Administration
-
6th Topic:
Software Management

Software comes from ...?

The initial software that we used to install came from the CDROM that we downloaded.
(Try to open the .iso file!)

In this chapter we will explore the organisation of the software, as it is done for Ubuntu, Debian, RedHat, etc.

Basic organisation

All, I repeat **ALL**, software installed on a typical *nix system is organised in **packages**. This applies for the initial install and any other software.

On a typical Windows system, there are a large number of **files** that are copied into the system directory (usually C:\Windows\system32).

Additional software comes with so-called *installers*, executable files, that *add* the files to the system. These are most often installed into C:\Program Files\

Packages? What Packages?

All, I repeat **ALL**, software installed on a typical *nix system is organised in **packages**. This applies for the initial install and any other software.

The idea is, to package all software, including installation instructions, helper files, documentation (the famous 'man'-pages) into a package; a bit like an ***archive***. (Think 'zip' or 'rar' if in doubt.)

There are basically two different packaging formats available:

.rpm – RedHat Package Manager

.deb – DebIan (packages)

There is a number of software to manage packages available:

- rpm
- aptitude
- yum
- dpkg
- apt-get
- ...

Initial Installation

In a nutshell, at the installation, after partitioning, the **base system** was set up by starting the package manager and make the package manager install all packages that are needed by the system, plus the packages selected by the user (e.g. LAMP, ssh).

That means, the package manager read the packages from the installation-CD, and, following the included instruction, copied the files and directories to the designated locations.

Plus, it wrote all the information about this process into a **database on the system**, for later reference, for example to know if a package has been installed, when it has been installed, etc.

Once all basic packages were in place, the system would be **rebooted**.

The tricky and beautiful part

Want to install more, other, packages?

That's possible, as long as they are available on the CD. But other software? Debian, Ubuntu, RedHat and other Linux distributions pride themselves for having easily 20.000+ packages available.

How?

Where?

→ on the Internet!

You can get them from the sites of the vendors (Ubuntu, RedHat, Novell, etc.). Such a collection of software is called ***repository***.

But since this software is Free Software (see chapter 2), everyone can set up a ***mirror*** of that respository and offer this software closer to home; for example from within Malaysia.

There are 200+ mirrors available worldwide for the most popular distributions.

How and where?

There are tools available, on the GUI, to handle all of this easily, with a click of the mouse. We have no GUI (yet); and we look at it from the perspective of the system administrator, not the user. So we need to know how it actually **works**!

/etc/apt/sources.list

contains the *sources* (mirror sites or repositories – NOT: source code!) from where the packages can be downloaded. If one wants to change the mirror, one needs to change the content of the repository, and then the software will be downloaded from a different site, or a different mirror.

Usually, we select the mirror from which we can get the fastest *data rate*.

sources.list - example

```
deb http://ftp.jaist.ac.jp/pub/Linux/ubuntu/ maverick main restricted
# deb-src http://ftp.jaist.ac.jp/pub/Linux/ubuntu/ maverick main restricted
deb http://ftp.jaist.ac.jp/pub/Linux/ubuntu/ maverick-updates main restricted

## N.B. software from this repository is ENTIRELY UNSUPPORTED by the Ubuntu
## team. Also, please note that software in universe WILL NOT receive any
## review or updates from the Ubuntu security team.
deb http://ftp.jaist.ac.jp/pub/Linux/ubuntu/ maverick universe
deb http://ftp.jaist.ac.jp/pub/Linux/ubuntu/ maverick-updates universe

deb http://ftp.jaist.ac.jp/pub/Linux/ubuntu/ maverick-security main restricted
deb http://ftp.jaist.ac.jp/pub/Linux/ubuntu/ maverick-security universe

deb http://download.virtualbox.org/virtualbox/debian maverick non-free
```

Repository *update*

Once a mirror is selected, it is necessary to find out, which packages are available from it.

Once we have changed or selected a mirror, we need to get a copy of the **list of packages** available on that mirror.

While there are many ways to do that, in this course we will focus on **apt**, probably the most user-friendly software for package management.

The command simply is

```
$ apt-get update
```

This contacts the server(s) selected in sources.list, and downloads the most recent package **lists** from there. Be aware, this is not any install yet! It is only obtaining the most recent list, plus **information** about each package, like use, original source, size, dependencies (other packages that it needs to run properly), etc.

In short, after 'apt-get update' all information about all packages available from a repository is stored on our **local** machine. We can use it, even without Internet connection.

local use

The command for using apt **locally**, it is apt-cache, because it uses the locally 'cached' information.

```
$ apt-cache show firefox
Package: firefox
Priority: optional
Section: web
Maintainer: Ubuntu Mozilla Team <ubuntu-mozillateam@lists.ubuntu.com>
Architecture: amd64
Version: 3.6.13+build3+nobinonly-0ubuntu0.10.10.1
Provides: iceweasel, www-browser
Depends: fontconfig, psmisc, lsb-release, debianutils (>= 1.16), libasound2 (>> 1.0.22), libatk1.0-
0 (>= 1.29.3), libc6 (>= 2.11), libcairo2 (>= 1.2.4), libdbus-1-3 (>= 1.0.2),[...]
Recommends: ubufox
Suggests: firefox-gnome-support (= 3.6.13+build3+nobinonly-0ubuntu0.10.10.1), firefox-kde-support,
ttf-lyx, libthai0
Filename: pool/main/f/firefox/firefox_3.6.13+build3+nobinonly-0ubuntu0.10.10.1_amd64.deb
Size: 12665398
MD5sum: d2724085bacfe69253c8e64a784f549b
SHA1: 8b7be543450b9d60cd648220a717a773a2efe7eb
SHA256: a7d7760a6ed39092dac21957c6c4756473b3954ca03a5ccfcc10843f286ce541
Description: safe and easy web browser from Mozilla
 Firefox delivers safe, easy web browsing. A familiar user interface,
 enhanced security features including protection from online identity theft,
 and integrated search let you get the most out of the web.
Bugs: https://bugs.launchpad.net/ubuntu/+filebug
```

Finding a software

In the previous slide, the program name (firefox) was known. Most often, it is not. Do not despair, it is possible to **search** the **local list** for applications.

In this example, we will search for all packages that have the word 'tablet' in name or description

```
$ apt-cache search tablet
krita - a pixel-based image manipulation program for the KDE Office Suite
xserver-xorg-input-wacom - X.Org X server -- Wacom input driver
cycle - calendar program for women
easystroke - gesture recognition program
gromit - GTK based tool to make annotations on screen
gtkwhiteboard - GTK+ Wiimote Whiteboard
kde-config-tablet - implements a KDE configuration GUI for the Wacom drivers
libhildon-1-0 - Hildon libraries - shared libraries
libhildon-1-0-dbg - Hildon libraries - detached debug symbols
libhildon-1-dev - Hildon libraries - development files
libhildon-1-doc - Hildon libraries - API documentation
mednafen - multi-platform emulator, including NES, GB/A, Lynx, PC Engine
modest - small e-mail program targetting hardware with modest resources
modest-dbg - small e-mail program targetting hardware with modest resources -- debug symbols
mypaint - Paint program to be used with Wacom tablets
mypaint-data - Brushes and backgrounds for the mypaint program
tablet-encode - video converter for Nokia Internet Tablets
xserver-xorg-input-aiptek - X.Org X server -- Aiptek input driver
xserver-xorg-input-fpit - X.Org X server -- FPIT input driver
```

Installing a software

Looking through the list, we find an application to draw on our newly acquired tablet: mypaint. So we decide to install this application. Maybe we want to check the description before installing it?

```
$ apt-cache show mypaint
Package: mypaint
[...]
Description: Paint program to be used with Wacom tablets
 This is a pressure sensitive Wacom tablet paint program. It comes with a large
 brush collection including charcoal and ink to emulate real media, but the
 highly configurable brush engine allows you to experiment with your own
 brushes and with not-quite-natural painting.
Homepage: http://mypaint.intilinux.com/
```

This sounds okay to us; the program seems to offer what we were looking for. Now we need to install it.

```
$ apt-get install mypaint
```

will contact the repository selected in /etc/apt/sources.list and download the package, as well as all other eventually needed packages, and install them for us. There is nothing else we have to do, the package contained a menu icon, and we could now use it. (Provided, we had a GUI, which we don't have yet because we were doing a server install. On the desktop version, we could now start drawing … :)

Also, this package is registered into the database of our installed packages, that we could query to retrieve the installation information.

Uninstalling a software

Maybe we don't like the software? Maybe it didn't do what we were looking for? Or, we want to re-install it later; once we have a GUI?

Nothing easier than that:

```
$ apt-get remove mypaint
```

This will remove all the files and directories installed and created during the installation. It will also remove it from the list of installed software. It will, however, **leave the package** on our machine, so that it does not need to be downloaded if we later decide to try again.

Actually, **all** packages **downloaded** are stored locally, in the directory /var/cache/apt/archives/ for later use – e.g. on another machine, or on the same machine.

"Come on, I don't want to use all that space for packages that I do not actually have installed on my machine!" - No problem:

```
$ apt-get autoclean
```

will remove all the packages from /var/cache/apt/archives/ that are **not currently** installed.

"No, I don't need **any** of the downloaded packages!"

```
$ apt-get clean
```

removes **all** packages from /var/cache/apt/archives/

Upgrading software

There are always patches available for software, because software happens to have ***bugs*** or ***vulnerabilities***. In Windows, there is the Windows Update.

How is this done with apt-get? Very easy. But: When you have a proxy involved, and a rather slow Internet connection, it is not very comfortable. For the moment, here is how to do it when one has **no proxy** to take care for:

Firstly, one updates the information from the repository, from the mirrors, to obtain a list of the most recent software:

```
$ apt-get update
```

Then, one asks apt-get to **compare** the **versions** of the software **installed**, with the versions **available** from the mirrors and repositories. When there is a newer, more recent, version available, apt-get will download and install this / these newer versions for us automatically. The command is

```
$ apt-get upgrade
```

This is in principle all.

Comparing it to update on Windows, there is one advantage here: Windows Update will update/upgrade the basic system (only). If one has other applications, like Chrome, Thunderbird, or any other software, the Windows Update does not know about that, and one needs to update that software as well, manually.

Due to the organisation in packages, with a 'apt-get upgrade', **all** packages will be updated, including all **applications** installed via apt-get. In the earlier example, mypaint, the drawing program for tablets, would be updated as well.

Querying packages

While apt-get is **very** comfortable to use, it is not very helpful for more complex tasks, like querying packages.

Here, **dpkg**, another package manager, is better. (dpkg also allows all of those tasks as mentioned earlier, but does not do it that simple.)

The main tasks for using dpkg: list and search installed packages.

```
$ dpkg -L mlocate
```

shows the files in a specific package, here mlocate

```
$ dpkg -l
```

shows all installed packages. Remember grep to obtain a display containing a list of specific terms:

```
$ dpkg -l | grep firefox
ii  firefox                                 3.6.13+build3+nobinonly-
0ubuntu0.10.10.1          safe and easy web browser from Mozilla
ii  firefox-branding                        3.6.13+build3+nobinonly-
0ubuntu0.10.10.1          Package that ships the firefox branding
ii  kubuntu-firefox-installer               10.10ubuntu4
                Mozilla Firefox installer for Kubuntu
```

References

- https://help.ubuntu.com/community/InstallingSoftware
- http://www.thegeekstuff.com/2009/10/debian-ubuntu-install-upgrade-remove-packages-using-apt-get-apt-cache-apt-file-dpkg/
- https://help.ubuntu.com/community/AptGet/Howto
-