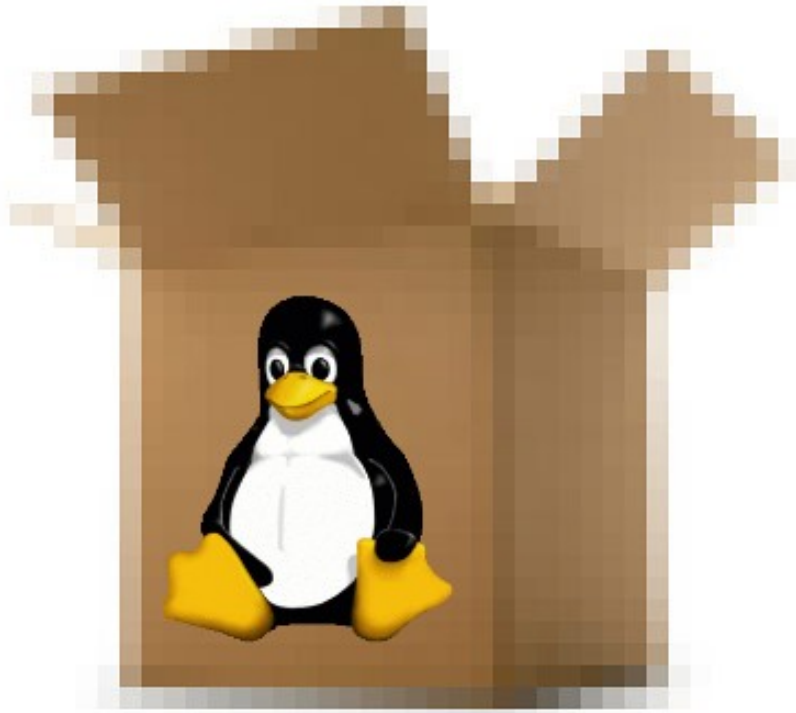


Package Management in LINUX



System Administration
Teaching Professor: Pilar Manzano García
Erasmus Student: Isaakidis Marios
DNI Number: AE196428



POLITÉCNICA



Spring Semester 2011
UNIVERSIDAD POLITÉCNICA DE MADRID



Installing software in LINUX



It's true that a computer's most important element nowadays is software. The LINUX operating systems are well known for the diversity and adjustability they offer in this field. Nevertheless, installing new software might be frustrating in the beginning, but soon the user will realize the unlimited options he has in his hands.

Unlike Windows' setup.exe and Mac's .sit files which provide only the option of following basic steps through a series of dialogs, in LINUX you can choose either to compile the source code by yourself, or use precompiled binaries known as **software packages**, to upgrade your software list. The first way is really useful, as the administrator can totally configure from the "core" the software to be installed, though sometimes might be a long and demanding task. The second option, software packages, considered by many as LINUX's greatest innovation, can help the junior user to automate and secure that procedure of software setup, upgrade or even uninstallation. This project is focused on presenting briefly LINUX software packages and their administration fundamentals.

Installing from Source Code

Software for LINUX systems in general is open source, meaning that it's creators publish the code freely in websites like SourceForge.net, allowing you to study, contribute on it and alter it under the GNU General Public License (currently version 3.0) or similar. For this example, we will use the source code of Audacity, a multi-track audio editor which supports several formats and interesting features. Usually, you can download the source code in GNU tarball format with the extension .tar.gz (or .tar.bz2 if bzip2 compression is used).

Tarball files, though it's not recommended and should be done strictly in root directory, can also be used for installing by the command:

```
$ tar xvzf [tar file]  
x = extract  
v = verbose  
z = zip. If bzip2 compression is used then replace with j  
f = file
```

The files of archive will be extracted in the respective FHS directories.

The following procedure is considered obsolete and is now used when there are no available binary packages, when there is incompatibility with libraries, or when we want to adjust the program to our needs (adding fixes, removing unneeded features, enabling compile-time options for better performance etc.).

It is interesting to mention that Gentoo distribution employs a source-based package system, Portage, which automates the next steps and gives you more or less the mentioned advantages.



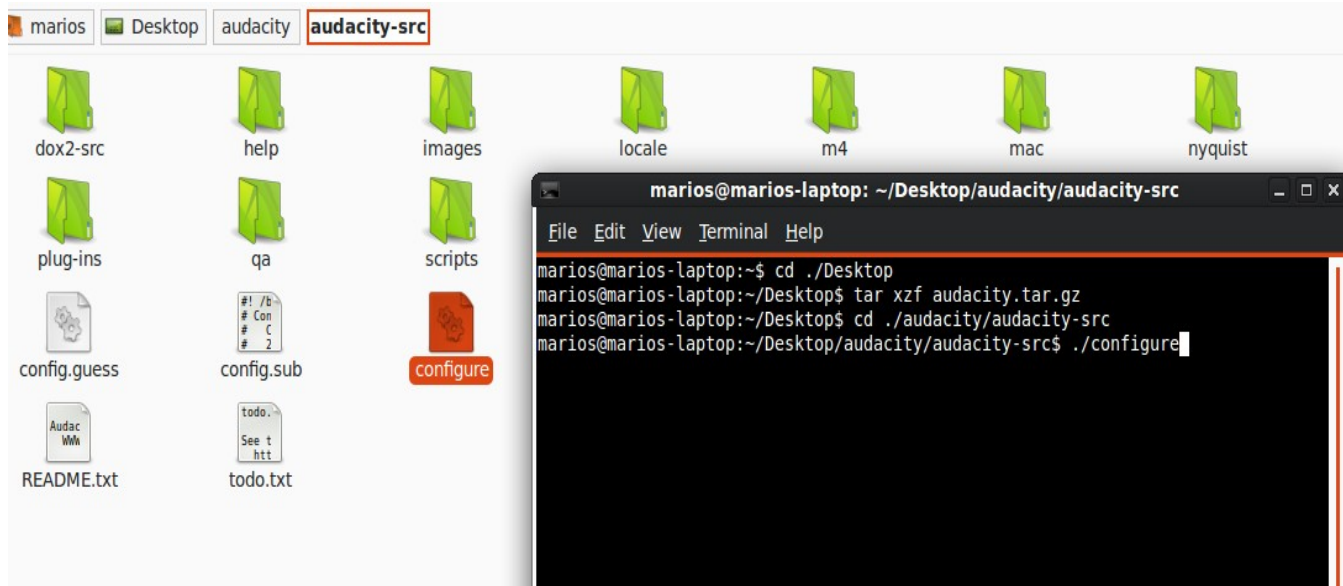
Audacity in Windows XP environment

First step, decompress the .tar file with the above command, read the Documentation and accept the License. In the main directory of the source code you will find the file README.txt, which contains programmers' notifications and usually Install instructions.

Now it's time to parse first configurations. Most of the times it will be easy to notice the bash program "configure", which will examine your computer for it's characteristics, presence of specific features including dependencies etc.

Following LINUX's simplicity principles, a package should do only one task and should be able to be used by the others. Dependencies are "requirements that exist between packages" [www.linuxselfhelp.com].

A wise thing to do if you are planning to modify some features, is to execute **\$./configure --help**. This command will give you information about the command-time configurations you can use.



If there are missing libraries, you will be notified and you can download them from internet. Debian system users can install essentials for building from source code, via executing the command:

\$ sudo apt-get install build-essential

When configure has finished, you will notice some files have been created, including Makefile, which have stored your preferences. You can now proceed by actually compiling the code, using by default the command **\$ make**. To finish the installation, execute as superuser **\$ make install**.

Software Packages

“A package is a file containing the files necessary to implement a set of related commands or features”; in this case “the executable files, libraries, and documentation associated with a particular suite of program or set of related programs” [www.debian.org]. “Packages also contain metadata, such as the software's name, description of its purpose, version number, vendor, checksum, and a list of dependencies necessary for the software to run properly. Upon installation, metadata is stored in a local package database. Operating systems based on LINUX and other Unix-like systems typically consist of hundreds or even thousands of distinct software packages; in the former case, a package management system is a convenience, in the latter case it becomes essential” [www.wikipedia.org]. It is the most common and easiest way to share software as automates and secures the management of programs in LINUX systems.

These packages are administrated by **package managers**, a collection of software tools included by default on all LINUX distributions. The two dominants on this field are the Red Hat Package Manager (RPM) and the Debian GNU/LINUX Package Manager.






Among other things, Package Management Systems (PMS) offer many benefits:

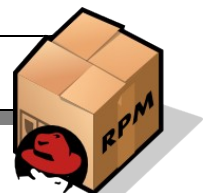
- Provide tools for installing, updating, removing and managing the software on your system.
- Allow you to install new or upgraded software directly across a network.
- Tell you what software package a particular file belongs to or what files a package contains.
- Maintain a database of packages on the system and their status, so that you can determine what packages or versions are installed on your system.
- Provide dependency checking, so that you don't mess up your system with incompatible software.
- Provide GPG, PGP, MD5, or other signature verification tools.
- Provide tools for building packages.

[Robbins A., 2005 p. 468]

Building an RPM package

The Red Hat Package Manager (RPM) is the most popular cross-platform package manager, originally developed by Red Hat but now widely adopted by several big distributions like SUSE  CentOS  and Mandriva 

An RPM package has the form `package-version-release.architecture.rpm`
`audacity-1.2.3-11.x86_64.rpm` for example is such a file.



An RPM package is consisted of three main parts:

- the header contains all information about the package
- the signature used to validate the source of package
- the archive composed by the actual files needed for the installation

If you are planning to create an RPM package, one of the requirements is to create a **spec** file. This file “contains all the information required to build a package, including a description of the software, instructions telling the **rpmbuild** command how to build the package, and a list of the files included and where they get installed.” [Siever E., 2009 p. 552]

To begin with, we have to install all needed tools for packaging, by executing the commands in an RPM-supporting PMS:

```
$ yum groupinstall "Development Tools"  
$ yum install rpmdevtools
```

yum is a meta-packager that gives you the ability to easily install a package (that might not exist in your hard drive) and its dependencies through the command

```
$ yum install [name of package]
```

*It also supports commands like **update**, **remove** and **search** for easier administration of rpm packages.*

Now we should create a user for building the packages under his directory. Never try to create packages under root user, so if something goes wrong your computer will be invulnerable. You can execute as superuser

```
$ useradd [user name] and then to change his password you need to run  
$ passwd [user's password]
```

Log into this user by **\$ su [user name]** and execute in your home directory **\$ rpmdev-setuptree**. This script will create a **~/rpmbuild** directory which we will use for creating the packages. Copy the original source .tar file in **~/rpmbuild/SOURCES** folder. Be sure to remove any external libraries included, they should be packaged separately and added as dependencies. Also verify that you are allowed to package this software via reading its License (OSS licences allow you to do that, some of them under special premises).

Here comes the most interesting part, creating the spec file. Execute **\$ rpmdev-newspec [program name]** and you can find under **~/rpmbuild/SPEC** directory a template **[program name].spec** file. At appendix B there is a custom **audacity.spec** file, created specially for the project, of audacity's latest stable version (1.2.6), patched to support MP3 encoding by default. [I had to rename the compressed source folder from **audacity-src-1.2.6** to **audacity-1.2.6** in order to avoid adding more information on the .spec file, as during the **rpmbuild** execution there was an “invalid” directory].

Now we can proceed in creating the binary and source RPM packages:

```
$ rpmbuild -ba [program].spec  
$ rpmbuild -bs [program].spec
```


[www.debian.org]. The innovation APT brought to LINUX was the automatic dependency resolution. While dpkg can work with individual packages, the command **\$ apt-get install [name of package]** will search in the sources listed in /etc/apt/sources.list file, download the packages or source code you have requested and execute all needed actions for the successful installation of the program.

The sources.list file contains entries in the following format:

deb [server-type] [address] [directories] [areas]
deb-src [server-type] [address] [directories] [areas]

Most archives are FTP or HTTP servers, but you can also insert rsh or SSH server or even a CD or a directory from your computer. Be sure to execute as root
\$ apt-get update after modifying the sources.

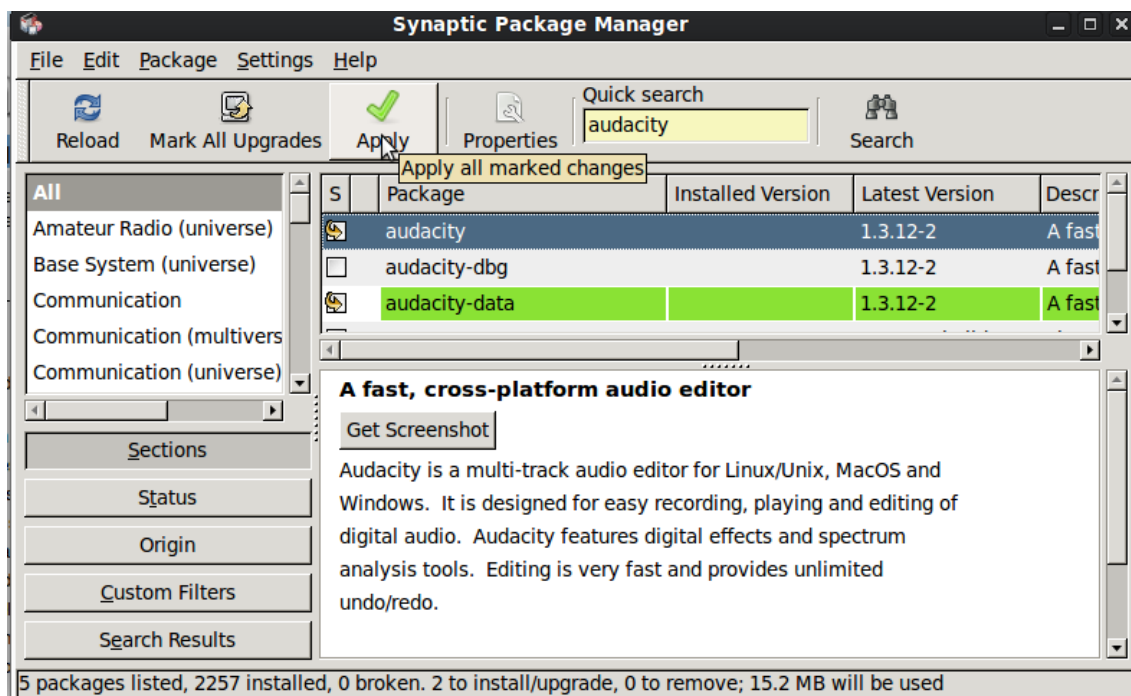
Here follow some of the basic commands APT supports. You need to be root to execute them:

\$ apt-get install [program]	APT installs the latest version of the package in all the available sources. In every following situation, can add the version you prefer by [package]=[version number / stable / unstable / testing]
\$ apt-get upgrade	APT will actually upgrade to the newest stable versions all packages installed in your system
\$ apt-get build-dep [program]	build-dep causes apt-get to install/remove packages in an attempt to satisfy the build dependencies for a source package.
\$ apt-get source [program]	APT will fetch source packages from the packages you requested. If followed by --compile it will create a .deb binay package using dpkg-buildpackage that you can later install
\$ apt-get check	check is a diagnostic tool; it updates the package cache and checks for broken dependencies.
\$ apt-get remove [program]	APT removes the packages, though leaving it's configuration files on system
\$ apt-get clean	clean clears out the local repository of retrieved package files


```
marios@marios-laptop:~$ sudo apt-get install audacity
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  audacity-data
Suggested packages:
  ladspa-plugin
The following NEW packages will be installed:
  audacity audacity-data
0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
Need to get 5,121kB of archives.
After this operation, 15.2MB of additional disk space will be used.
Do you want to continue [Y/n]? █
```

APT waiting authorization to install audacity

The Advanced Packaging Tool can automate the installation from source code, or even upgrade of the whole system, in only one command. Things become even better, as there are GUI front-ends of APT, like **synaptic**.



Installing audacity with synaptic

Furthermore, there are tools like **Ubuntu's Software Center** which offer a user-friendly environment, categorize applications by their use and allow you to install a program only by mouse clicks.

One to rule them all...



Studying the advance in PMS during the last years, you can notice a tendency to create interfaces more friendly to users, while keeping the same principles as defined time ago independent for each one. The big fight between RPM and dpkg for dominance doesn't help users who usually have to search in internet for optimized packages, missing dependencies or incompatibilities among libraries, even more when they try to combine packages from both systems. Also, although LINUX is called the universal OS available for many architectures, most packages are compiled only for i386 or i686 architectures.

These remarks gave the spark for creation of new PMS, **Portage** and **Conary**. Portage as mentioned before, which follows the Ports system used by *BSD distributions, downloads the source code from Gentoo's Portage tree and compiles it using users' CFLAG environment variables.

During Free Open Source Developers' European Meeting 2011, many LINUX contributors were discussing this topic. The time when there were conflicts between distributions seem to have passed, as most welcomed the idea of creating a single unified package management system, something like a “cross-distro App-Store [...] The power in Linux's package management comes from three primary things: repository flexibility, dependency management, and complete file system access” [Damien Radtke]. In my opinion, such an accomplishment is to be expected in the next versions. But we can be sure for one thing, that this freedom LINUX offers to users, to totally have control of their systems and parametrize it according to their needs, will never extinct.



Ubuntu Software Center offering 35707 packages categorized in "Departments"



References

- Turnbull J., Lieverdink P., Matotek D., **Administración de sistemas Linux**, Anaya Multimedia, Madrid 2010
- Krafft, Martin F., **The Debian system : concepts and techniques**, No Starch, San Francisco 2005
- Rodríguez de Sepúlveda Maillo, **Manual imprescindible de Linux**, Anaya Multimedia, Madrid 2009
- Morrill, Daniel L., **Tuning and customizing a Linux system**, Berkeley Apress, 2002
- Anguiano, Eloy, **Linux**, Pearson Educación, Madrid 2008
- Stanfield, Vicki, **Linux System Administration**, Sybex, San Francisco 2002
- Roderick W. Smith, **Linux administrator street smarts : a real world guide to Linux certification skills**, Wiley, Indianapolis 2007
- Roderisk W. Smith, **LPIC-1 Linux Professional Institute Certification – Study Guide**, Wiley, Indianapolis 2009
- Siever E., Figgins S., Love R., Robbins A., **Linux in a nutshell – A Desktop Quick Reference**, O'Reilly, USA 2009
- Robbins A., **UNIX in a nutshell**, O'Reilly, USA 2005
- <http://www.linux.com> , <http://www.debian.org> , <http://fedoraproject.org> , <http://ianmurdock.com> , <http://wikipedia.org>

Appendix: Custom audacity.spec File

```
Name:                audacity
Version:             1.2.6
Release:             1%{?dist}
Summary:             Audio editor
Group:               Applications/Multimedia
License:             GPLv2
URL:                 http://audacity.sourceforge.net
Source0:
http://sourceforge.net/projects/audacity/files/audacity/1.2.6/audacity-src-
1.2.6.tar.gz

Patch1:              audacity-1.3.7-libmp3lame-default.patch

BuildRoot:           %{_tmppath}/%{name}-%{version}-%{release}-root-%(%{__id_u} -n)

BuildRequires:      desktop-file-utils
BuildRequires:      expat-devel
BuildRequires:      flac-devel
BuildRequires:      gettext
BuildRequires:      jack-audio-connection-kit-devel
BuildRequires:      ladspa-devel
BuildRequires:      libid3tag-devel
BuildRequires:      taglib-devel
BuildRequires:      libogg-devel
BuildRequires:      libsamplerate-devel
BuildRequires:      libsndfile-devel
BuildRequires:      libvorbis-devel
BuildRequires:      soundtouch-devel
BuildRequires:      vamp-plugin-sdk-devel
BuildRequires:      zip
BuildRequires:      zlib-devel
BuildRequires:      wxGTK-devel
BuildRequires:      libmad-devel

%description
Audacity is a cross-platform multitrack audio editor. It allows you to
record sounds directly or to import files in various formats. It features
a few simple effects, all of the editing features you should need, and
unlimited undo. The GUI was built with wxWidgets and the audio I/O
supports OSS and ALSA under Linux.

%prep
%setup -q
```

```
%build
%configure
make %{?_smp_mflags}

%install
rm -rf %{buildroot}
make install DESTDIR=%{buildroot}

%clean
rm -rf %{buildroot}

%files
%defattr(-,root,root,-)
%doc

%changelog
```