Package Management – Overview Sheet

Installers

On any personal computer system there will be a way to install new programs and services. The most common and oldest method is to use an installer provided by the same people who provide the target software. An installer is a program which when executed, guides the administrator through various questions to get the program installed correctly.

It usually doesn't know a lot about the target system and the variety and quality of each installer can vary quite a bit. There is also not often any checks against the installer before executing to make sure it came from a trusted source, allowing installers to be bundled with viruses and other malware.

Under Gnu/Linux based computers an installer will normally come in the form of a .bin or .run file, this file needs to have execution permissions added to it, before the user can execute it. Use of installers in Ubuntu is **heavily discouraged,** as the system will be unable to keep track of the files or even what the installer is doing.

This is how you can use a binary installer on the command line:

```
chmod 755 install.bin
sudo ./install.bin
```

Source Packages

An archive or compressed file is a tar or zip file that contains a directory full of files. These often contain source distributions which need to be compiled. It's recommended that dealing with source archives be left to packagers and developers unless no other options are available, as they can present the same loss of tracking as an installer.

This is how you can typically install a source package on the command line:

```
tar xzvf source-package.tar.gz
cd source-package-dir
./configure && make && sudo make install
```

Packages

A package is the more modern managed way of installing all software. It keeps track of dependencies, installation and uninstallation tasks as well as following strict guidelines and best practices for compatibility. Under Ubuntu and Debian a package will have a .deb extension and can be installed by double clicking on it or by using the command below, under red hat based systems such as fedora or openSuSE you will see packages with an rpm extension (not covered in this course).

It is recommended that all software be installed via packages as you get the benefit of organised, know packages, fully verified sources and reliable uninstallation:

```
dpkg -i package.deb
```

Repositories

A repository is a location on the internet or local network which contains a list of all packages that it is possible to install on the computer. On Ubuntu by default there is the apt sources system which contains a number of official repositories that are used to install applications via apt-get, it's possible to add custom repositories and specify new locations to get things. Most often this will be in the form of a PPA (Personal Package Archive) and normally we do this to install software which has not been made available in the main repositories.

The official repositories are managed by the MOTU (Mast of the Universe) team, and packages are only allowed into the main system once they've passed strict rules and guides.

The advantages of using repositories is that your computer can keep every single installed application up to date, and easily install a complete upgraded system without loosing compatibility or functionality.

Finding and Installing Packages

The simplest way to install applications is to use apt-get, this tool can quickly install any required library or program and all it's dependencies, you must be root in order to install programs system wide:

```
sudo apt-get install package_name
```

You can also use a program called apt-cache to search for available packages in the available repositories as well as getting details about a package version:

```
apt-cache search term
apt-cache showpkg package_name
```

You can get much more detailed information from the apt-file program, this will allow you to find the package which contains the specified file, as well as listing all the files in a given package:

```
sudo apt-file update
apt-file search filename
apt-file list package_name
```

If you want to automatically install any required packages that will be used by a running process, you can use auto-apt and it will prompt to install packages which contain the files that the running program required, this is very useful for compiling new programs or broken packages:

```
auto-apt run program_name
```

Package Management

You can remove a package using apt-get, as well as reinstall, reconfigure and fix issues using the following commands:

```
sudo apt-get remove package_name
sudo apt-get purge package_name
sudo apt-get install -reinstall package_name
sudo dpkg-reconfigure package_name
```

Downloaded packages all end up cached in the directory /var/cache/apt/, you can use apt-get to clear the cache of removed packages as well as clearing the entire cache and save space on the computer:

```
sudo apt-get autoclean
sudo apt-get clean
```

You can upgrade packages using apt-get, this will download all updates which will fix security and other problems. You can also upgrade the entire distribution, which are more pronounced updates and thus carry more risk of breaking:

```
sudo apt-get upgrade
sudo apt-get dist-upgrade
```

Repository Management

All repository configurations are stored in /etc/apt/source.list and sources.list.d/ directory. Each section must be formatted correctly and it's best practice to copy and paste from the form shown in documentation, the source for all packages can be set using the graphical tool, or manually by editing the list:

```
[type] [location] [version] [repository names]
deb http://us.archive.ubuntu.com/ubuntu/ hardy main restricted
```

Once you've edited the configuration you have to update the repository cache in order to install new programs:

```
sudo gedit /etc/apt/source.list
sudo apt-get update
```

Each repository needs to have a public key available which signs the packages. This allows the local computer to confirm that the package downloaded is the package requested and that it hasn't been tampered with. You can add keys using the apt-key program but you need to know the key number shown in the example as 12345678:

```
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 12345678
```

Creating a Local Repository

You may want to create a local network repository for all the local computers, this enables you to provide your network with all the approved company wide packages or custom packages and any updates in a controlled fashion. This is simply done by providing an Apache based web server with a specific directory structure and a package index which specifies the available packages and their dependencies.

Creating an Apt-Mirror

An apt mirror is a locally provided service which allows all computers on the network to download packages from a local apt server, this server will periodically download/sync with a central database and only contains packages from the official repositories. Though an apt-mirror does take a great deal of disk space (15GB per architecture, per release).

```
sudo aptitude install apache2 apt-mirror
sudo gedit /etc/apt/mirror.list
sudo apt-mirror
```

Each mirror will need configuration in the mirror.list file, detailing each of the source repositories to mirror. The apache2 http server will then serve the contents to the local network and to make this possible we create a sym-link to the /var/www directory.

Each client that uses the mirror should be configured to now use the mirror instead of the internet based repository. This is similar to the PPA method above, but should contain your local machine's address instead of the external ppa server.

Creating an Apt-Cache Service

An apt cache service is similar to an apt-mirror, but instead of syncing all packages and having a local collection of everything. The service only downloads packages on request and caches them locally. This has the benefits of saving lots of disk space but has the disadvantage of not allowing new packages to be installation without an internet connection.

```
sudo aptitude install apache2 apt-cacher
sudo gedit /etc/default/apt-cacher
sudo /etc/init.d/apt-cacher start
```

The cacher is a running service and similar to the mirroring above, each client should be configured to use the local machine as it's repository.

Creating a Custom Apt Repository

So far the two solutions above only allow access to packages on existing systems, but what if you want to allow your systems to access your own deb packages locally? In this case you use the same apache technology used above, but you create your own directories and package indexes, the dpkg-dev package contains all the required tools to build the indexes.

Create a new directory in your configured Apache installation, for a user apt repository, use ~/public_html and for system wide use a /var/www/ directory.

```
sudo apt-get install dpkg-dev
mkdir -p ~/public_html/myrepository
cd ~/public_html/myrepository
```

This directory should contain two sub-directories, one called binary and one called source, inside each of these directories you should copy your binary and source deb packages respectively.

```
mkdir binary
mkdir source
cp my-downloaded-package-i386.deb binary/
cp my-downloaded-package-src.deb source/
```

Once all the packages are in place, you use a program called dpkg-scanpackages to generate the package indexes:

```
dpkg-scanpackages binary /dev/null | gzip -9c > binary/Packages.gz
dpkg-scansources source /dev/null | gzip -9c > source/Sources.gz
```

Configure each client to use your new repository as in the previous two examples.