Linux #Course - 2

mtk-notes-bd.blogspot.com/p/linux-course-2.html

Mohammad Taseen Khan

Linux Installation And Package Management

#1 Linux Installation :

Understanding about it

Installing software on Linux involves package managers and software repositories, not downloading and running .exe files from websites like on Windows. If you're new to Linux, this can seem like a dramatic culture shift.

While you can compile and install everything yourself on Linux, package managers are designed to do all the work for you. Using a package manager makes installing and updating software easier than on Windows.

Linux vs. Windows

There are a wide variety of Linux distributions and a wide variety of package managers. Linux is built from open-source software, which means that each Linux distribution compiles its own software with its desired library versions and compilation options. Compiles Linux applications generally don't run on every distribution – even if they could, installation would be hindered by competing package formats.If you locate a Linux application's website, you'll likely see a variety of download links for different package formats and Linux distributions – assuming the application's website provides pre-compiled versions at all. The application may tell you to download the source code and compile it yourself.

Software Repositories

Linux users don't normally download and install applications from the applications' websites, like Windows users do. Instead, each Linux distribution hosts their own software repositories. These repositories contain software packages specially compiled for each Linux distribution and version. For example, if you're using Ubuntu 12.04, the repositories you use contain packages specially compiled for Ubuntu 12.04. A Fedora user uses a repository full of packages specially compiled for their version of Fedora.

Package Managers

Think of a package manager like a mobile app store – except they were around long before app stores. Tell the package manager to install software and it will automatically download the appropriate package from its configured software repositories, install it, and set it up – all without you having to click through wizards or hunt down .exe files on websites. When an update is released, your package manager notices and downloads the appropriate update. Unlike on Windows, where each application must have its own updater to receive automatic updates, the package manager handles updates for all installed software — assuming they were installed from the software repositories.

What's a Package ?

Unlike on Windows, where applications come in .exe installer files that can do anything they like to the system, Linux uses special package formats. There are a variety of package types – most notably DEB on Debian and Ubuntu and RPM on Fedora, Red Hat, and others. These packages are essentially archives containing a list of files. The package manager opens the archive and installs the files to the location the package specifies. The package manager remains aware of which files belong to which packages – when you uninstall a package, the package manager knows exactly which files on the system belong to it. Windows has no idea what files belong to an installed application – it lets application installers manage installation and uninstallation themselves.

Packages can also contain scripts that run when the package is installed and removed, although these are generally used for system setup and not moving files to arbitrary locations.

Installing Software on Linux

To install software on Linux, open your package manager, search for the software, and tell the package manager to install it. Your package manager will do the rest. Linux distributions often offer a variety of frontends to the package manager. For example, on Ubuntu, the Ubuntu Software Center, Update Manager, Synaptic application, and apt-get command all use apt-get and dpkg to download and install DEB packages. You can use any utility you like – they just provide different interfaces. You'll generally find a simple, graphical package manager in your Linux distribution's menus.

Update Delays

One thing new Linux users often notice with package managers and repositories is a delay before new software versions reach their systems. For example, when a new version of Mozilla Firefox is released, Windows and Mac users will acquire it from Mozilla. On Linux, your Linux distribution must package the new version and push it out as an update. If you open Firefox's preferences window on Linux, you'll note that Firefox has no ability to automatically update itself (assuming you're using the version of Firefox from your Linux distribution's repositories).

You can also download and install the application yourself – for example, downloading Firefox directly from Mozilla — but this may require compiling and installing the software from source and removes the benefits of package managers, such as automatic, centralized security updates.

While new versions of Firefox are a priority because they contain security updates, other applications may not be delivered as quickly. For example, a major new version of the LibreOffice office suite may not ever be released as an update for the current version of your Linux distribution. To avoid potential instability and allow time for testing, this version may not be available until the next major release of your Linux distribution – for example, Ubuntu 12.10 – when it becomes the default version in the distribution's software repositories.

To fix this problem, some Linux distributions, such as Arch Linux, offer "rolling release cycles," where new versions of software are pushed into the main software repositories. This may cause problems – while you may want new versions of desktop applications, you probably don't care about new versions of low-level system utilities, which could potentially introduce instability.

Ubuntu offers the backports repository to bring newer versions of significant packages to older distributions, although not all new versions make it into the backports repository.

Other Repositories

While Linux distributions ship with their own repositories pre-configured, you can also add other repositories to your system. Once you have, you can install software repositories from that repository and receive updates from it using your package manager. The repository you add must be designed for your Linux distribution and package manager.

For example, Ubuntu offers a wide variety of personal package archives (PPAs), which contain software compiled by individuals and teams. Ubuntu doesn't vouch for the stability or security of the packages in these repositories, but you can add PPAs from trusted individuals to download packages not yet in Ubuntu's repository – or download newer versions of existing packages.

Some third-party applications also use their own software repositories. For example, when you install Google Chrome on Ubuntu, it adds its own apt repository to your system. This ensures you receive updates to Google Chrome through Ubuntu's Update Manager and standard software installation tools.

#2 Package Management :

All the operating systems depend on a set of software applications to carry out user intended tasks. In the early days, applications were tested against bugs before release to provide a better user experience. Now the software application is released with the intent to apply bug fixes in new versions. Moreover, each application has its updater, or the user has had to figure out how to obtain the upgraded software release.
Linux adopted the timely software management practice by creating packaging formats, software packages, and unique installation tools. This article discusses how the software package installation process upgraded from tarball package installation to DEB and RPM package management.

Tarball

Earlier Linux systems software addition required the user to download the source code, compile it in binary files, and add it to the system. Sometimes the software was made available by some users in a compiled form known as the tarball. A tarball contains multiple files including, executables, configuration files, documentation, and libraries. Such that all the files are compressed into a single file for easy storage and distribution.

After software installation, the files spread across the system in relevant directories. However, the method of creating tarball may seem easy, but the installation process makes some tasks difficult, for instance:

It requires the user to independently/manually track down the dependencies for the installing software such that the dependent software itself has some dependencies.

Since tarball package installation spreads the files, it won't be easy to locate the package documentation and configuration files even if the user knows the commands.

It's hard to locate files to uninstall software.

The absence of metadata in tarballs leaves users confused about the version details after installation. That makes it difficult to track bugs and get new versions.

To overcome these problems, software packaging in the Linux distributions evolved into two packaging formats known as DEB and RPM packaging.

DEB Packaging

The Debian and Debian-based Linux distributions use DEB-base software packaging. The .deb files include all relevant files with metadata in a .ar archive format. The metadata contains all the relevant software details involving version, description, dependencies, licenses, etc. Debian distributions offer multiple graphical interfaces and terminal-based tools to manage .deb files. Some of them include:

1. apt: Ubuntu advanced packaging tool that provides an apt-get command to search and manage package installation.
2. aptitude: the command is a package management tool that provides a text-based interface to run inside the terminal. It performs package installation, removal, and upgrade by using arrow keys and highlighting the selected option.
3. Ubuntu Software Centre: It is an intuitive graphical user interface for beginning Linux users searching and installing packages.

Even though Ubuntu Software Center is intuitive, the advanced packaging management system outperforms all the other PMS for DEB packaging.

RPM Packaging

The RPM (.rpm) packaging format is the preference of SUSE, Fedora, and Red Hat, and RHEL-based Linux distributions. The RPM package is the amalgam of files to provide a photo viewer, word processor, or other software to RHEL distribution users. It also contains configuration files, metadata, and other required documents to create the software.

The RPM Package Manager combines binaries and all the required files available via upstream software providers into an RPM package. Before including packages into the repository, they are signed so the users can verify their validity. Now the user can access these packages for installation from repositories placed inside CDs or directories via NFS or FTP servers.

The RPM package name tells a lot about the software. For instance, type the following command to find out the details of the currently installed RPM package of firefox:

rpm -q firefox

Result comes firefox-<build version and release info>.x86_64

Dependency Hell:

The RPM package installation fails in the absence of dependencies while telling about the required components. Moreover, the dependent package itself has some needed

dependencies to get the work done.

RPMs Location:

The RPM Package manager expects to receive the package location before installation. If the package is available in the current folder, it requires an input of firefox-87.0-12.fc34.x86_64.rpm, if it's on the server, it requires http://example.com/firefox-87.0-12.fc34.x86_64.rpm.

Whereas at that time, DEB-based software packaging could automatically resolve the dependencies problem. However, after the increasing popularity of RPM packages, the issues have been resolved with the yum facility.

YUM Project

The Yellowdog Updater Modified (YUM) facility was introduced to manage RPM packages dependencies by considering each RPM package as part of a large software repository. Such that the problem of dealing with the dependencies is for the Linux distribution or third-party software.

It resolves the problems with the concept that repositories can build on each other. For instance, if a user installs some package from the rpmfusion.org repository, which requires a command/tool from the main Fedora repository, it also has access to that. Hence, it will be downloaded and installed in the meantime.