

Central Package Management System

Authors: Jason Herne, Kevin Roberts, Chris Johnson, Aaron Hughes

Introduction

Currently, each Linux distribution has its own way of managing software updates and installations. If you run multiple Linux distributions you must familiarize yourself with many different methods for installing and upgrading software. Furthermore, if a certain software package is not supported by your distribution, you must find the package and all of its dependencies and do the install on your own. Often times you find yourself waiting weeks or even months for newly released versions of your favorite programs to make their way into your package management system's update list. Furthermore, many package management systems do not automatically handle dependencies for you. For example, if you try to install a package with RedHat's RPM tool, it will alert you if that particular package has a dependency that is not installed, but you must track down the correct RPM to fill the dependency and download and install it yourself.

We propose a distribution independent package management system which provides a more effective solution for software updating and installation.

Our System

We model our system after the Gentoo Linux Portage Tree, which currently provides a central location for all packages for users of Gentoo Linux. This provides global access to the source code of all available packages, along with the means of building and installing the package on the user's system. The system has dependencies built in. You only have to specify the package you want to install, and it will find all of the dependencies and install them as well. All this is done automatically, and all software is downloaded, compiled from source, and installed without user intervention.

We feel that the Gentoo Linux Portage Tree is a good start at a unified package management system, however it can be extended to further enhance its functionality, ease of use, and rate of adoption. We build off of the Gentoo Linux Portage System in many ways.

We will support source code packages. Since source code packages are supported, users can rest assured that, worst case, they will be able to compile a version of any program that will work on their specific architecture and operating system combination. The use of patches and special installation meta-data and scripts will automatically be used to handle any architecture or operating system specific installation requirements. This means that we could

conceivably create such meta-data that would allow for the installation of most packages (including X windows and Gnome/KDE) in a Windows operating system environment. More to the point, any combination of software/hardware can easily be supported provided that it is possible to make the code compile and run on the hardware and operating system.

Binary packages will also be supported for every package in the management system. This means that pre-built binary packages could be made available for popular packages so users do not have to spend time compiling everything themselves. For example, someone running a software development shop could pre-compile the gcc compiler for Linux and AIX with optimizations for the hardware in use. This would generate several binary packages that the end users could download and install in a matter of seconds. No compilation would be necessary and no extra steps would need to be taken by the end users.

Centralized and distributed operation will both be possible with this system. In a centralized environment, a single entity (a distributor like RedHat, or a service provider like IBM) would host and maintain the central package update server. Users would configure their clients to get software updates from this server. This is exactly like current Microsoft Windows Update technology. In a distributed environment, each package maintainer (Gnome Foundation, KDE, Sun, IBM) could host a server that offers update and install packages for their products only. Gnome's server would allow people to install Gnome, Gimp, Abiword, Gnumeric, and all of the other Gnome related programs. Adobe might wish to host a server that would allow users to update and install Acrobat Reader. Clients could then be pointed at many different servers to gain their updates. Ideally, a mixed mode system would prevail. A centralized system containing most software would be maintained by a company like IBM or RedHat and then companies that develop other software products (like Adobe, Network Associates and Norton) could have their own servers that manage just their products. The client could then be configured to check all of these servers for updates. A priority system would ensure that Gnome updates (for example) would be taken from the most reliable source in the event that multiple servers provide the Gnome packages. The distributor's server could also redirect package requests for the Gnome packages (again using Gnome as an example) to official Gnome servers. A network of packages update servers could be formed. On a side note, wide spread adoption of this system could require companies and open source projects to maintain an update server, but it would also alleviate the need to maintain a public FTP server.

Rollback provides a way to undo changes caused by a malformed update. This ensures system integrity and allows a user to

return to a stable system at the push of a button. This also allows users to experiment with unstable packages without having to worry about damaging the host operating environment.

Package Collections allow easy installation of mission critical application environments. To illustrate the usefulness of this, consider installing the Gnome graphical desktop environment. Gnome itself is not a package but is made up of about 70 different packages. Not many people would be able to find and install all the correct packages to get gnome working. Install a virtual package titled "Gnome" would be created which would list all 70 of the required packages as dependencies. This way, the user only needs to say "I want to install Gnome" and all of the dependencies would be automatically downloaded, built, and installed.

Different packages can be marked with different stability levels. Stability Levels give users the option of installing the latest versions of software, the most stable versions, or somewhere in between. This gives users greater control of their system. Power users that like to run and test bleeding edge software have that opportunity while users who need to assurance of quality tested and controlled software will also be satisfied.

A Graphical User Interface will be constructed to make it easy for both novice and experienced users to install and update packages on their system. This could also allow for other features such as automatic updating to ensure that your system is always up to date, similar to that of other operating systems.

No physical media is necessary to keep systems up to date. The use of the Internet and automatic updating speed software updates to users in an extremely efficient manner. This will ensure that critical security related updates are installed as rapidly as possible, alleviating the need for many security e-mail alerts and the need for users to manually patch and protect their system against software related security threats.