25D Linux Foundation Course

07 – Managing Linux Software

Overview

- Managing software with RPM
- Installing software from source code
- Managing Debian software packages
- Managing shared libraries

Managing Software with RPM

- Two choices for installing software on a Linux system:
  - Install from source code
    - Must have a compiler installed (gcc)
  - Install from a package
    - Must have a package manager installed
      - Debian (Ubuntu)
      - Red Hat Package Manager (openSUSE and Red Hat)
        - Maintains an RPM database file (/var/lib/rpm)

Managing Software with RPM

- **All package manager tasks regardless of Distro:**
  - **Install software**
  - **Update software that's already been installed**
  - **Uninstall software**
  - **Query installed software**
  - **Verify the integrity of installed software**

Obtaining Linux Software

Notable Sources

- Installation CD or DVD
- Open source project websites
- www.rpmfind.net
- www.sourceforge.net
- linux.tucows.com
- www.freshmeat.net
- www.linux.org

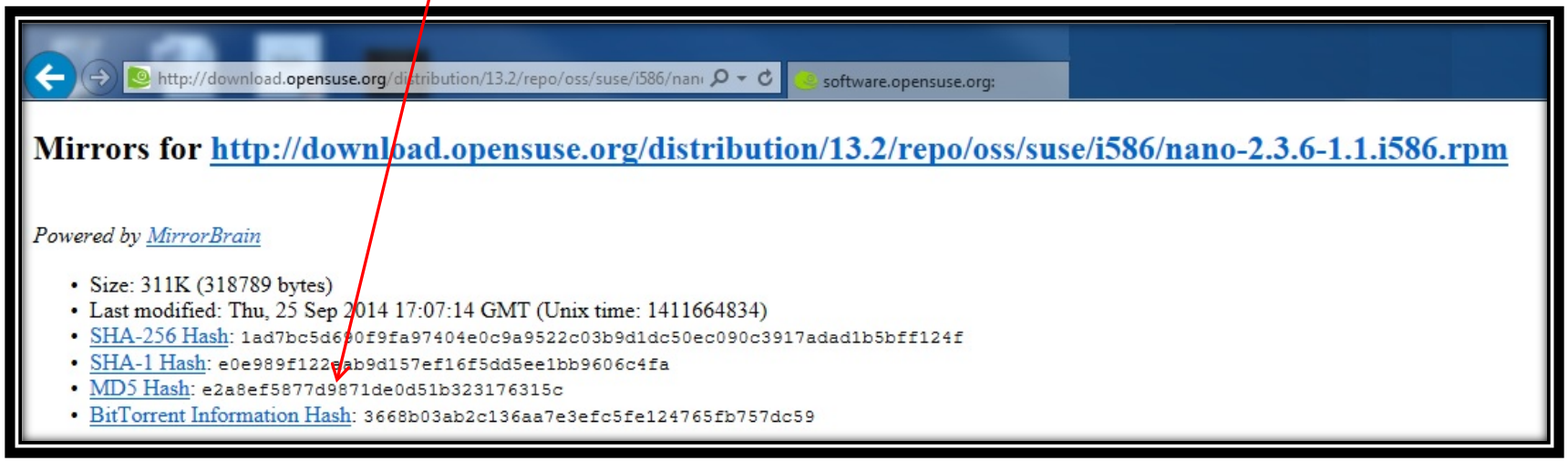
Obtaining Linux Software

❑ Verifying Downloads

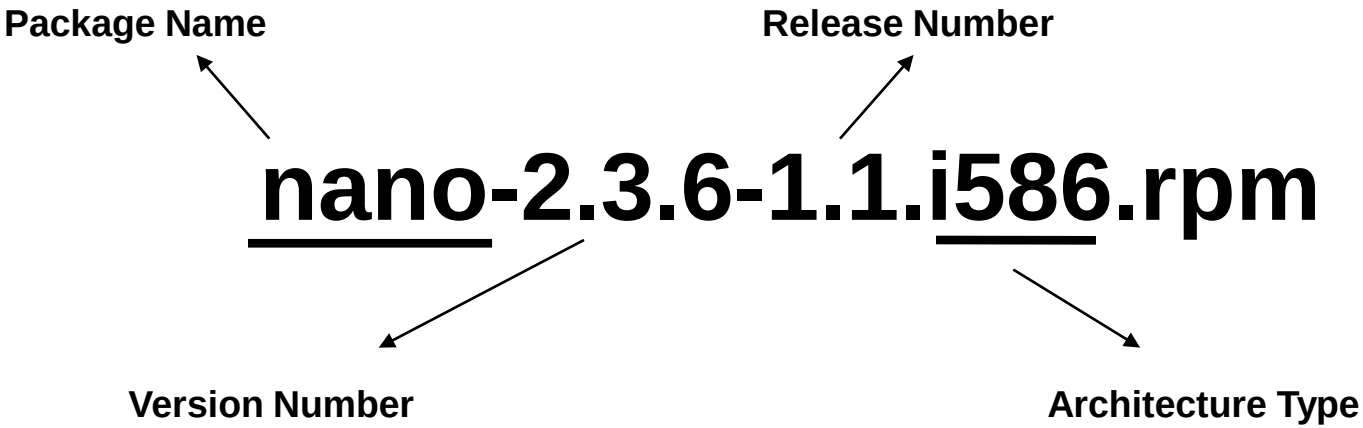
– Checksum

- value generated by calculating the contents of a file using a hashing algorithm

```
student@openSUSE:~> md5sum nano-2.3.6-1.1.i586.rpm
e2a8ef5877d9871de0d51b323176315c  nano-2.3.6-1.1.i586.rpm
```



Installing Software Packages



Installing Software Packages

- Check authenticity of package

```
student@openSUSE:~> rpm --checksig nano-2.3.6-1.1.i586.rpm
nano-2.3.6-1.1.i586.rpm: rsa sha1 (md5) pgp md5 OK
```

- Install package (local machine)

```
openSUSE:~ # rpm -i /home/student/nano-2.3.6-1.1.i586.rpm
openSUSE:~ # whereis nano
nano: /usr/bin/nano /usr/share/nano /usr/share/man/man1/nano.1.gz
```

- Install package (directly from internet)

```
openSUSE:~ # rpm -ihv "http://download.opensuse.org/repositories/openSUSE:/13.2/
standard/i586/nano-2.3.6-1.1.i586.rpm"
Retrieving http://download.opensuse.org/repositories/openSUSE:/13.2/standard/i58
6/nano-2.3.6-1.1.i586.rpm
Preparing...                          ########################################## [100%]
Updating / installing...
   1:nano-2.3.6-1.1                   ########################################## [100%]
openSUSE:~ # whereis nano
nano: /usr/bin/nano /usr/share/nano /usr/share/man/man1/nano.1.gz
```

- Install package (with “progress” bar)

```
openSUSE:~ # rpm -ihv /home/student/nano-2.3.6-1.1.i586.rpm
Preparing...                          ########################################## [100%]
Updating / installing...
   1:nano-2.3.6-1.1                   ########################################## [100%]
```

Exercise 7-1: Installing RPM Packages

Please open your Practical Exercise book to Exercise 7-1.

Time to Complete: 5 Minutes

Uninstalling Software

- Uninstall package

```
openSUSE:~ # rpm -e nano
openSUSE:~ # whereis nano
nano:openSUSErpm -ihv /home/student/nano-2.3.6-1.1.i586.rpm
```

- The rpm utility will perform a dependency check before installation and upon uninstallation of packages.
  - If other software is installed on the system that is dependent on the package you are trying to uninstall:
    - an error message will be displayed listing the dependent packages

Exercise 7-2: Uninstalling RPM Packages

Please open your Practical Exercise book to Exercise 7-2.

Time to Complete: 5 Minutes

Updating Software and Querying Packages

- Update a package
  - If the –U option is used and the package is not installed, it will automatically install the package

```
openSUSE:~ # rpm -U /home/student/nano-2.3.6-1.1.i586.rpm
        package nano-2.3.6-1.1.i586 is already installed
```

- Query a package
  - Adding the “a” option in place of a package name will generate a listing of all packages installed on the system
  - This list printed to the screen will be long, you can add the specific package name, or pipe to the more utility

```
openSUSE:~ # rpm -q nano
nano-2.3.6-1.1.i586
```

Updating Software and Querying Packages

Additional query options

- -i – displays summary information

```
openSUSE:~ # rpm -qi nano
Name        : nano
Version     : 2.3.6
Release     : 1.1
Architecture: i586
Install Date: Thu Dec  8 13:20:31 2016
Group       : Productivity/Editors/Other
Size        : 1004298
License     : GPL-3.0+
Signature   : RSA/SHA256, Thu Sep 25 11:07:14 2014, Key ID b88b2fd43dbdc284
Source RPM  : nano-2.3.6-1.1.src.rpm
Build Date  : Thu Sep 25 11:07:01 2014
Build Host  : cloud108
Relocations : (not relocatable)
Packager    : http://bugs.opensuse.org
Vendor      : openSUSE
URL         : http://www.nano-editor.org/
Summary     : Pico Editor Clone with Enhancements
Description :
GNU nano is a small and friendly text editor. It aims to emulate the
Pico text editor while also offering a few enhancements.
Distribution: openSUSE 13.2
```

Updating Software and Querying Packages

Additional query options

- **--whatrequires – displays a list of packages that require specified packages**

```
openSUSE:~ # rpm -q --whatrequires nano
no package requires nano
openSUSE:~ # rpm -q --whatrequires postfix
no package requires postfix
openSUSE:~ # rpm -q --whatrequires pptp
NetworkManager-pptp-0.9.8.4-1.3.i586
```

Updating Software and Querying Packages

Additional query options

- **-l – displays a list of files that are included in an RPM package**

```
openSUSE:~ # rpm -q -l nano
/usr/bin/nano
/usr/bin/rnano
/usr/share/doc/packages/nano
/usr/share/doc/packages/nano/AUTHORS
/usr/share/doc/packages/nano/COPYING
/usr/share/doc/packages/nano/COPYING.DOC
/usr/share/doc/packages/nano/ChangeLog
/usr/share/doc/packages/nano/ChangeLog.pre-2.1
/usr/share/doc/packages/nano/NEWS
/usr/share/doc/packages/nano/README
/usr/share/doc/packages/nano/THANKS
/usr/share/doc/packages/nano/TODO
/usr/share/doc/packages/nano/UPGRADE
/usr/share/doc/packages/nano/faq.html
/usr/share/doc/packages/nano/man-html
/usr/share/doc/packages/nano/man-html/nano.1.html
/usr/share/doc/packages/nano/man-html/nanorc.5.html
```

Updating Software and Querying Packages

❑ Additional query options

- **--provides – displays the functionality the specified package supplies**

```
openSUSE:~ # rpm -q --provides nano
nano = 2.3.6-1.1
nano(x86-32) = 2.3.6-1.1
openSUSE:~ # rpm -q --provides pptp
config(pptp) = 1.7.2-42.1.2
pptp = 1.7.2-42.1.2
pptp(x86-32) = 1.7.2-42.1.2
openSUSE:~ # rpm -q --provides postfix
config(postfix) = 2.9.6-7.4.1
dict_ldap.so
dict_pcre.so
dict_tcp.so
libpostfix-dns.so.1
libpostfix-global.so.1
libpostfix-master.so.1
libpostfix-milter.so.1
libpostfix-tls.so.1
libpostfix-util.so.1
postfix = 2.9.6-7.4.1
postfix(x86-32) = 2.9.6-7.4.1
smtp_daemon
sysvinit(postfix)
sysvinit(sendmail)
sysvinit(smtp)
```

Updating Software and Querying Packages

Additional query options

- --requires – displays the functionality required by the specified package
- --whatprovides – displays name of the package that provides specified program

```
openSUSE:~ # rpm -q -requires postfix
/bin/awk
/bin/bash
/bin/grep
/bin/sed
/bin/sh
/bin/sh
/bin/sh
/bin/sh
/bin/sh
/usr/bin/getent
/usr/sbin/groupadd
/usr/sbin/useradd
/usr/sbin/usermod
config(postfix) = 2.9.6-7.4.1
coreutils
diffutils
fileutils
fillup
grep
```

```
openSUSE:~ # rpm -q --whatprovides fillup
fillup-1.42-269.1.2.i586
openSUSE:~ # rpm -q --whatprovides fileutils
coreutils-8.21-7.1.3.i586
openSUSE:~ # rpm -q --whatprovides coreutils
coreutils-8.21-7.1.3.i586
```

Verifying Packages

- **Verify a package**
  - **rpm –V package_name**
    - **Error messages may appear during the verification process. The following is the syntax:**

SM5DLUGT c *filename*

```
openSUSE:~ # rpm -V -a
/usr/bin/wodim: cannot verify root:root 0755 - not listed in /etc/permissions
.M.......    /var/lib/ca-certificates/openssl
.M.......    /var/lib/ca-certificates/pem
S.5....T.  c /usr/share/fonts/encodings/encodings.dir
S.5....T.  c /usr/share/fonts/misc/fonts.dir
S.5....T.    /usr/share/fonts-config/conf.avail/17-suse-bitmaps.conf
.......T.  c /usr/share/fonts/100dpi/fonts.dir
S.5....T.  c /usr/share/fonts/Speedo/fonts.dir
S.5....T.  c /usr/share/fonts/Speedo/fonts.scale
S.5....T.  c /usr/share/fonts/Type1/fonts.dir
S.5....T.  c /usr/share/fonts/Type1/fonts.scale
.......T.  c /usr/share/fonts/cyrillic/fonts.dir
S.5....T.  c /usr/share/fonts/truetype/fonts.dir
S.5....T.  c /usr/share/fonts/truetype/fonts.scale
SM5....T.  c /etc/default/grub
....L....    /usr/lib/libblas.so.3
```

Exercise 7-3: Managing RPM Packages

Please open your Practical Exercise book to Exercise 7-3.

Time to Complete: 5 Minutes

Extracting Files from Packages

- At some point you may need to extract files from a package
- Using rpm2cpio
  - Similar to tar
  - rpm2cpio package_name > archive_filename
  - Converts an rpm to a cpio archive file

Extracting Files from Packages

- In this example an archive file of nano is created. We then move to a directory and extract the archived file:

```
openSUSE:/home/student # rpm2cpio nano-2.3.6-1.1.i586.rpm > nano.cpio
openSUSE:/home/student # ls
.Xauthority       .local                        Downloads
.bash_history     .mozilla                      Music
.bashrc           .profile                      Pictures
.cache            .rcc                          Public
.config           .skel                         Templates
.dbus             .thumbnails                   Videos
.dmrc             .vboxclient-draganddrop.pid   bin
.emacs            .viminfo                      docs
.esd_auth         .xim.template                 nano-2.3.6-1.1.i586.rpm
.fonts            .xinitrc.template             nano.cpio
.gstreamer-0.10   .xsession-errors              public_html
.gtkrc-2.0        .xsession-errors-:0           test.txt
.inputrc          Desktop
.kde4             Documents
openSUSE:/home/student # cd Downloads
openSUSE:/home/student/Downloads # cpio -idv < /home/student/nano.cpio
```

- The –i option will do the extraction, the –d will create directories as needed and the –v will display files extracted

Extracting Files from Packages

- The files are extracted from the package and subdirectories are placed in the current directory:

```
./usr/share/nano/sh.nanorc
./usr/share/nano/spec.nanorc
./usr/share/nano/tcl.nanorc
./usr/share/nano/tex.nanorc
./usr/share/nano/texinfo.nanorc
./usr/share/nano/xml.nanorc
1980 blocks
openSUSE:/home/student/Downloads # ls
nano-2.3.6-1.1.i586.rpm  usr
openSUSE:/home/student/Downloads # ls /home/student/Downloads/usr
bin  share
openSUSE:/home/student/Downloads # ls /home/student/Downloads/usr/bin
nano  rnano
openSUSE:/home/student/Downloads # ls /home/student/Downloads/usr/share
doc  info  man  nano
```

- If nano were to be installed by the rpm utility the listed files would have been installed in the listed directories instead of the usr shown

Extracting Files from Packages

- Another extraction example with cpio:

```
openSUSE:~/Output # find . -name "*.txt" | cpio -o > txt.cpio
1 block
openSUSE:~/Output # cd /home/student/Pictures
openSUSE:/home/student/Pictures # cpio -idv < /root/Output/txt.cpio
file1.txt
file2.txt
file3.txt
1 block
openSUSE:/home/student/Pictures # ls
file1.txt  file2.txt  file3.txt
```

- In this example a cpio archive file was created using find to identify all .txt files in the current directory and piping those to cpio and creating an archive file named txt.cpio.

- We then move to another directory and extract the three text files.

Using yum to Install RPM Packages

- Yellowdog Updater Modified (yum)
  - Allows installation of a package and all of its dependencies
  - can have issues with this using rpm….

```
openSUSE:~/Downloads # rpm -ivh yum-3.4.3-14.2.2.i586.rpm
error: Failed dependencies:
        python-gpgme is needed by yum-3.4.3-14.2.2.i586
        python-iniparse is needed by yum-3.4.3-14.2.2.i586
        python-yum = 3.4.3 is needed by yum-3.4.3-14.2.2.i586
```

  - It locates packages for you by searching one or more repositories on the Internet
  - Not in all Linux distros by default

Using yum to Install RPM Packages

- Can be downloaded and installed via command line or graphical interface

```
openSUSE:~/Downloads # ls
python-gpgme-0.1-118.1.4.i586.rpm
python-iniparse-0.4-18.1.4.noarch.rpm
python-urlgrabber-3.9.1-12.2.2.noarch.rpm
python-yum-3.4.3-14.2.2.i586.rpm
yum-3.4.3-14.2.2.i586.rpm
yum-metadata-parser-1.1.2-129.1.3.i586.rpm
```

- Syntax for yum
  - yum option command *package_name*
    - yum install rubygem-rake-compiler-0.9.2-2.1.4.i586.rpm
- Configuration files
  - /etc/yum.conf
  - /etc/yum.repos.d/

yum Commands

Command	Description
`yum install` *`package_name`*	This command installs the specified package.
`yum remove` *`package_name`*	This command uninstalls the specified package.
`yum list all`	This command lists all packages in a repository and packages installed on your system.
`yum list installed`	This command lists all packages installed on your system.
`yum list installed` *`package_name`*	This command checks to see if the specified package is installed on your system.
`yum list` *`package_name`*	This command searches the configured repositories for the specified package.

yum Commands

Command	Description
`yum list available`	This command displays a list of all packages available for installation in the configured repositories.
`yum list updates`	This command generates a list of updates for all installed packages.
`yum list update` *`package_name`*	This command checks for updates for the specified package.
`yum info` *`package_name`*	This command displays information about the specified package, including its version and dependencies.
`yum whatprovides path/filename`	This command identifies the RPM package that provides the specified file.
`createrepo` *`/path`*	This command defines the path to a package repository.

yum Repository Configuration File Information

Element	Description
[repo_name]	Defines a repository.
name=	Defines a name for the repository.
baseurl=	Defines a URL where the file is located. In the examples given in the text, the repositories are located on the Internet. However, you can also define a local repository either on a repo server on your network or even in your local file system. The syntax for baseurl= in these situations is protocol:///path_to_repo. For example, if the repository is located on the local hard drive in the /var/repos directory, you would use a baseurl of file:////var/repos.
enabled=	Enables (1) or disables (0) the repository.
gpgcheck=	Enables (1) or disables (0) GPG security key checking to validate repository files.
gpgkey=	Specifies the location of the GPG security key.

yum Repository Configuration File Information

- Example repository file entry with minimum configuration settings:

```
openSUSE:/etc/yum/repos.d # ls
testrepo.repo  testrepo1.repo
openSUSE:/etc/yum/repos.d # cat testrepo.repo
[testrepo]
name=testrepo
baseurl=http://download.opensuse.org/distribution/13.2/repo/oss/
enabled=1
```

- The files above had to be created and were not created/added with the install of yum

Installing Software from Source Code

- Preparing the installation files
- Compiling the executable
- Installing the executable
- Uninstalling software compiled from source code

Installing Software from Source Code

- We have seen the installation of applications via packages but software can also be installed via source code:
  - Advantage to source code method
    - Developers do not have to create a package/executable for each architecture and platform
  - Disadvantage
    - More complex process
    - Users must have a compiler and know the procedures for compiling the source code and installing the resulting executable

Preparing the Installation Files

- Installation files used to install source code is typically distributed as a tarball file
- Use the tar command to extract files from tarballs
  - tar –zxvf *filename*
    - z – use gzip to decompress
    - x – extract the files from the decompressed archive file
    - v – verbose
    - f – name of file to extract
- After files have been extracted, they must be prepared to be compiled. This is done by using the configure command
  - compiles the program and creates the Makefile files

Preparing the Installation Files

- Tar extracts the file to a directory:

```
openSUSE:~ # tar -zxvf pure-ftpd-1.0.36.tar.gz
pure-ftpd-1.0.36/m4/ax_check_compile_flag.m4
pure-ftpd-1.0.36/m4/ax_check_link_flag.m4
pure-ftpd-1.0.36/m4/getloadavg.m4
pure-ftpd-1.0.36/m4/Makefile.am
pure-ftpd-1.0.36/m4/Makefile.in
pure-ftpd-1.0.36/README
pure-ftpd-1.0.36/configure.ac
pure-ftpd-1.0.36/aclocal.m4
pure-ftpd-1.0.36/Makefile.am
pure-ftpd-1.0.36/Makefile.in
pure-ftpd-1.0.36/config.h.in
pure-ftpd-1.0.36/pure-ftpd.spec.in
pure-ftpd-1.0.36/configure
pure-ftpd-1.0.36/AUTHORS
pure-ftpd-1.0.36/COPYING
pure-ftpd-1.0.36/ChangeLog
pure-ftpd-1.0.36/INSTALL
pure-ftpd-1.0.36/NEWS
pure-ftpd-1.0.36/THANKS
pure-ftpd-1.0.36/compile
pure-ftpd-1.0.36/depcomp
pure-ftpd-1.0.36/install-sh
pure-ftpd-1.0.36/missing
pure-ftpd-1.0.36/HISTORY
```

```
openSUSE:~ # ls
.bash_history   .gnupg   .local     Output     pure-ftpd-1.0.36
.config         .kbd     .viminfo   bin        pure-ftpd-1.0.36.tar.gz
.dbus           .kde4    Downloads  inst-sys
```

```
openSUSE:~/pure-ftpd-1.0.36 # ls
AUTHORS                        README.Donations      depcomp
CONTACT                        README.LDAP           gui
COPYING                        README.MacOS-X        install-sh
ChangeLog                      README.MySQL          m4
FAQ                            README.PGSQL          man
HISTORY                        README.TLS            missing
INSTALL                        README.Virtual-Users  pam
Makefile.am                    README.Windows        pure-ftpd.png
Makefile.gui                   THANKS                pure-ftpd.spec.in
Makefile.in                    aclocal.m4            puredb
NEWS                           compile               pureftpd-ldap.conf
README                         config.h.in           pureftpd-mysql.conf
README.Authentication-Modules  configuration-file    pureftpd-pgsql.conf
README.Configuration-File      configure             pureftpd.schema
README.Contrib                 configure.ac          src
README.Debian                  contrib
```

Preparing the Installation Files

- After files have been extracted, they must be prepared to be compiled. This is done by using the configure command

```
openSUSE:~/pure-ftpd-1.0.36 # ./configure_
```

  - Make sure you are in the directory tar extracted the files to

```
config.status: creating man/pure-pwconvert.8
config.status: creating man/pure-authd.8
config.status: creating config.h
config.status: executing depfiles commands
configure: +-------------------------------------------------------+
configure: | You can subscribe to the Pure-FTPd users mailing-list |
configure: | to ask for help and to stay informed of new releases. |
configure: | Go to http://www.pureftpd.org/ml/ now!                |
configure: +-------------------------------------------------------+
```

  - The configure file is a script that does two things:
    - checks your system to make sure everything to compile the program is available
    - compiles the program and creates the Makefile files

Compiling and Installing the Executable

- Makefile files:
  - contain specific instructions for how the executable should be compiled to run on that platform
- A C compiler (such as gcc) is necessary in order to complete the compiling process
- Execute the make command to read the source code files and generate a compiled executable file
- To install the program, the make command is run a second time using the install option

Compiling and Installing the Executable

- Run the make utility in the directory where the files where extracted:

```
openSUSE:~/pure-ftpd-1.0.36 # make
```

- The make utility will call the systems C compiler (in this case gcc) and direct it to read the source code files using the specification listed in the Makefile files

- This process with create an executable to be installed on the system

```
Making all in m4
make[2]: Entering directory `/root/pure-ftpd-1.0.36/m4'
make[2]: Nothing to be done for `all'.
make[2]: Leaving directory `/root/pure-ftpd-1.0.36/m4'
make[2]: Entering directory `/root/pure-ftpd-1.0.36'
make[2]: Nothing to be done for `all-am'.
make[2]: Leaving directory `/root/pure-ftpd-1.0.36'
make[1]: Leaving directory `/root/pure-ftpd-1.0.36'
```

Compiling and Installing the Executable

- To actually install the program on the system run the make utility again with and specify install

```
openSUSE:~/pure-ftpd-1.0.36 # make install_
```

- Let's verify Pure-FTPd installed:

```
openSUSE:~/pure-ftpd-1.0.36 # /usr/local/sbin/pure-ftpd &
[1] 8992
openSUSE:~/pure-ftpd-1.0.36 # ftp localhost
Trying ::1:21 ...
Connected to localhost.
220---------- Welcome to Pure-FTPd [privsep] ----------
220-You are user number 1 of 50 allowed.
220-Local time is now 16:17. Server port: 21.
220 You will be disconnected after 15 minutes of inactivity.
Name (localhost:root): anonymous
230 Anonymous user logged in
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

- It did as we can start the service, connect and log on anonymously

Exercise 7-4: Building Software from Source Code

Please open your Practical Exercise book to Exercise 7-4.

Time to Complete: 5 Minutes

Uninstalling Software Compiled from Source Code

- Typically the same process used to compile and install software compiled from source is used to uninstall programs
- During the compile process, an UNINSTALL target may be present in the Makefile file

```
openSUSE:~/pure-ftpd-1.0.36 # make uninstall_
```

- Run the configure command in the same directory that the original source code tarball was extracted and then run the make uninstall command

```
openSUSE:~/pure-ftpd-1.0.36 # /usr/local/sbin/pure-ftpd &
[2] 9134
openSUSE:~/pure-ftpd-1.0.36 # -bash: /usr/local/sbin/pure-ftpd: No such file or
directory
```

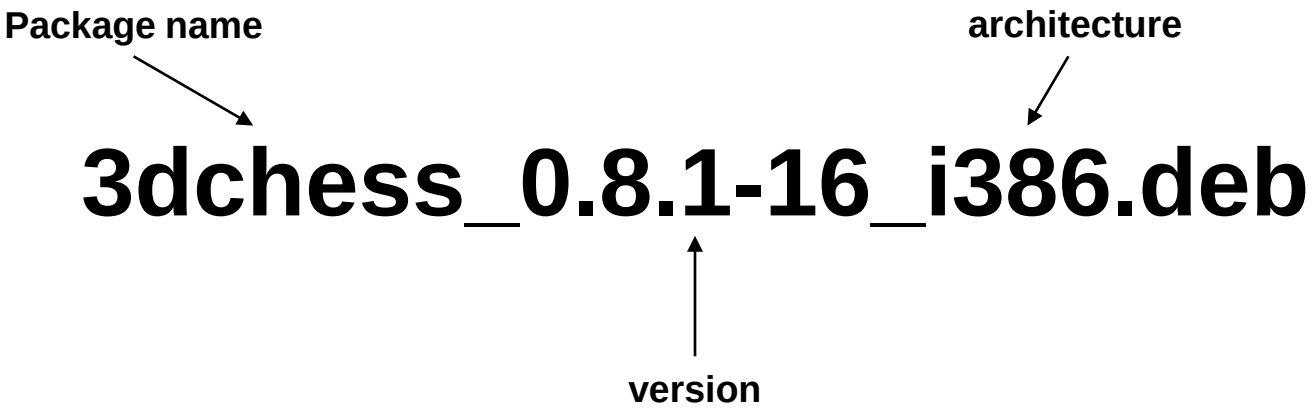
Managing Debian Software Packages

- Not all distributions use RPM to manage software packages
  - Some like Ubuntu and others use Debian Package Manager (dpkg)
- Packages are similar to RPM in functionality but completely different in implementation
  - RPM and Debian packages are not cross platform compatible
    - RPM can only install on RPM-based systems same for debian packages
    - There are utilities available that will allow you to convert from one to the other (alien is one utility)

Managing Debian Software Packages

- Debian package naming
- Installing packages with dpkg
- Viewing package information with apt-cache
- Installing packages with apt-get
- Using aptitude

Debian Package Naming



Installing Packages with dpkg

- The key command-line utility used to manage Debian packages is dpkg. The syntax for dpkg is:
  - dpkg options action package_name or package_filename
    - dpkg –i 3dchess_0.8.1-16_i386.deb
    - the above example would install a debian package named 3dchess
  - After the install of the package you can refer to the application by name and not the package name
    - For instance to display information about 3d chess you could enter the following:
      - dpkg –p 3dchess
    - Or uninstall it with this command
      - dpkg –r 3dchess

dpkg Command Actions

Action	Description
`-i`	Installs the specified package
`-r`	Uninstalls the specified package but does not delete its configuration files
`-P`	Uninstalls the specified package and deletes all of its configuration files
`--configure`	Reconfigures the specified package (can also be done with dpkg-reconfigure)
`-p`	Displays information about the specified package. The package must already be installed
`-I`	Displays information about a package that isn't currently installed on the system
`-l`	Lists all installed packages on the system
`-L`	Lists all files that were installed by the specified package on the system
`-S` *`filename`*	Identifies the package that installed the specified file

dpkg Command Options

Option	Associated Action	Description
`-B`	`-r`	When you're uninstalling a package that other packages are dependent on, this option disables those packages
`-G`	`-i`	This option tells dpkg to not install the specified package if a newer version of the same package is already installed
`-E`	`-i`	This option tells dpkg to not install the specified package if the same version of that package is already installed
`--ignore`	`-i or -r`	This option causes dpkg to ignore dependency information when installing or removing a package
`--no-act`	`-i or -r`	This option tells dpkg to check for problems, such as unresolved dependencies, when installing or removing a package
`--recursive`	`-i`	This option allows you to install multiple packages at once using * in the package filename part of the command. All matching packages in the current directory as well as subdirectories will be installed

Viewing Package Information with apt-cache

- In addition to dpkg there are several other Advanced Package Tools (apt) to manage packages on Debian-based systems
  - apt cache
    - Comparable to rpm –q command as it queries package information from the Debian package database
    - apt-cache showpkg 3dchess would display information about the 3dchess program
      - Version number
      - Description language
      - Dependencies
  - apt-get
    - Equivalent to the yum utility on an RPM system
    - Automatically downloads packages and dependant packages

Viewing Package Information with apt-cache

Common apt-cache Command	Description

`apt-cache showpkg package_name`
or
`apt-cache show package_name`
	Displays information about the package

`apt-cache stats`	Displays the number of packages installed, dependency information, and other package cache statistics
`apt-cache unmet`	Reports any missing dependencies in the package cache
`apt-cache depends package_name`	Displays all of the package's dependencies
`apt-cache pkgnames package_name`	Checks to see whether or not a package is installed on the system. Leaving out the package name displays a list of all the packages installed on the system
`apt-cache search keyword`	Searches package descriptions for the specified keyword

/etc/apt/sources.list

- The etc/apt/sources.list file defines the repositories from where apt-get can get packages
- Very similar to yum and it's conf files, can be optical disk as well (installation CD/DVD)

```
root@Ubuntu-Desktop:/etc/apt# cat sources.list
#deb cdrom:[Ubuntu 9.10 _Karmic Koala_ - Release i386 (20091028.5)]/
karmic main restricted
deb http://us.archive.ubuntu.com/ubuntu/ karmic main restricted
deb-src http://us.archive.ubuntu.com/ubuntu/ karmic main restricted
deb http://us.archive.ubuntu.com/ubuntu/ karmic universe
deb-src http://us.archive.ubuntu.com/ubuntu/ karmic universe
deb http://us.archive.ubuntu.com/ubuntu/ karmic-updates universe
deb-src http://us.archive.ubuntu.com/ubuntu/ karmic-updates universe
deb http://us.archive.ubuntu.com/ubuntu/ karmic multiverse
deb-src http://us.archive.ubuntu.com/ubuntu/ karmic multiverse
deb http://us.archive.ubuntu.com/ubuntu/ karmic-updates multiverse
deb-src http://us.archive.ubuntu.com/ubuntu/ karmic-updates multiverse
deb http://security.ubuntu.com/ubuntu karmic-security main restricted
deb-src http://security.ubuntu.com/ubuntu karmic-security main restricted
deb http://security.ubuntu.com/ubuntu karmic-security universe
deb-src http://security.ubuntu.com/ubuntu karmic-security universe
deb http://security.ubuntu.com/ubuntu karmic-security multiverse
deb-src http://security.ubuntu.com/ubuntu karmic-security multiverse
```

Common apt-get Commands

apt-get Command	Description
`install`	Installs the latest version of a specified package
`remove`	Removes the specified package
`update`	Displays updated information about all packages available in your configured package repositories
`upgrade`	Upgrades all installed packages to the newest version
`dist-upgrade`	Upgrades all installed packages to the newest version, but avoids upgrading packages if the upgrade would break a dependency
`check`	Verifies the integrity of installed packages as well as the package database
`clean`	Removes outdated information from the package database

Common apt-get Options

apt-get Option	Associated Command	Description
`-d`	`Upgrade install`	Downloads the specified package but doesn't install it
`-s`	`All commands`	Simulates the actions associated with the specified command but doesn't actually perform them
`-f`	`install remove`	Checks for unmet dependencies and fixes them, if possible
`-q`	`All Commands`	Suppresses progress information
`-y`	`All Commands`	Sends a default yes answer to any prompts displayed in the action
`--no-upgrade`		Tells apt-get not to upgrade a package if an older version of the package has already been installed

How Shared Libraries Work

- On Linux, applications running on the system can share code elements called shared libraries
- Considered code reuse and makes it so software developers don't have to reinvent every time they write a new program
  - Usually common functions
    - Opening a file
    - Saving a file
- With shared libraries developers can focus on code elements that are unique to an application
- For common elements they can link to prewritten code in the shared library

How Shared Libraries Work

- **Dynamic**
  - dynamic shared libraries exist as files in the Linux file system
  - programmers insert links to the functions in these shared libraries in their program code
  - functions are called from the dynamic shared libraries when the program is run and not integrated into the program itself
  - smaller applications but dependent on the shared library
- **Static**
  - linked statically into the program when it's compiled
  - in essence the actual code elements for the functions called are integrated directly into the application itself
  - results in larger applications but does make the application independent

Managing Shared Library Dependencies

- There is a configuration file to tell applications running on the system where they can find the dynamic shared library files
  - /etc/ld.so.conf

```
openSUSE:/etc # cat ld.so.conf
/usr/local/lib
include /etc/ld.so.conf.d/*.conf
# /lib64, /lib, /usr/lib64 and /usr/lib gets added
# automatically by ldconfig after parsing this file.
# So, they do not need to be listed.
```

Managing Shared Library Dependencies

- ldconfig –p
  - Lists all shared libraries available on the system
- ldd –v *executable_filename*
  - Lists shared libraries required by a specific application
- LD_LIBRARY_PATH
  - Lists directories to look in when trying to locate a shared library
- /etc/ld.so.cache
  - File that contains a list of all the system libraries; refreshed when system is initially booted and checked by apps on startup
  - ldconfig command can be used to rebuild library cache manually

Exercise 7-5: Working with Shared Libraries

❑ In this exercise, you will practice managing shared libraries. You can perform this exercise using the virtual machine that comes with this book. Run snapshot 8-3 for the correctly configured environment.

Complete the following:

1. With your system running, open a terminal session.
2. If necessary, change to your root user account by entering su – followed by your root user's password.
3. View the shared libraries used by the ping executable on your system by entering ldd –v /bin/ping at the shell prompt. You should see that ping requires the libc.so.6 shared library.
4. Find the location of the lib64/libc.so.6 library file on your system by entering find / –name libc.so.6 at the shell prompt. On a 32-bit system, you should see that the file resides in /lib. On a 64-bit system, it probably resides in /lib64.
5. View your system's library cache by entering ldconfig –p at the shell prompt.
6. Rebuild your library cache by entering ldconfig –v at the shell prompt.

Summary

- Managing software with RPM
- Installing software from source code
- Managing Debian software packages
- Managing shared libraries

Questions

Questions?

Check on Learning

Question 1

You've just downloaded a file named 3dchess-1.2.34.i586.rpm to the /home/student directory on your Linux system. Which of the below commands would you use to generate a checksum value? (Choose two.)

A. checksum /home/student/3dchess-1.2.34.i586.rpm

B. sum /home/student/3dchess-1.2.34.i586.rpm

C. md5sum /home/student/3dchess-1.2.34.i586.rpm

D. verify /home/student/3dchess-1.2.34.i586.rpm

E. rpm –V /home/student/3dchess-1.2.34.i586.rpm

Check on Learning

Question 2

You've just downloaded a file named FC-6-i386-DVD.iso and have generated a checksum value. The value generated is slightly different from that shown on the download website. What does this imply?

A. The downloaded copy is different from the original, but the download is still usable as long as the differences are minor.

B. The version number is incremented by 1 when the file was downloaded.

C. The downloaded copy is different from the original copy and shouldn't be used.

D. The downloaded copy is exactly the same as the original copy.

Check on Learning

Question 3

You've just downloaded a file named BitTorrent-5.0.1.tar.gz to your home directory. Assuming the current directory is ~, what command would you enter at the shell prompt to extract all the files from this archive?

A. gzip –d ./BitTorrent-5.0.1.tar.gz

B. tar –bxrf ./BitTorrent-5.0.1.tar.gz

C. tar –vf ./BitTorrent-5.0.1.tar.gz

D. tar –zxvf ./BitTorrent-5.0.1.tar.gz

Check on Learning

Question 4

Where does the RPM store its database of installed packages?

A. /var/lib/rpm

B. /etc/rpm

C. /var/rpmdb

D. /tmp/rpm

Check on Learning

Question 5

You've just downloaded an RPM package file named evolution-2.6.0-41.i586.rpm to your home directory. Assuming the current directory is ~, what command could you use to install the package on your system, displaying a progress indicator as the installation is completed? (Choose two.)

A. **rpm –i evolution-2.6.0-41.i586.rpm**

B. **rpm –ihv evolution-2.6.0-41.i586.rpm**

C. **rpm –U evolution-2.6.0-41.i586.rpm**

D. **rpm –install --progress evolution-2.6.0-41.i586.rpm**

E. **rpm –Uhv evolution-2.6.0-41.i586.rpm**

Check on Learning

Question 6

You need to uninstall the Pure-FTPd service from your Linux system. You've switched to the directory where the original installation files are located. What's the command you need to enter to uninstall this package?

A. ./configure

B. make

C. make remove

D. make uninstall

Check on Learning

Question 7

You've installed an RPM package file named evolution-2.6.0-41.i586.rpm on your Linux system. What command would you use to uninstall this package?

A. rpm –U evolution

B. rpm –U --remove evolution

C. rpm –i --remove evolution

D. rpm –e evolution

Check on Learning

Question 8

You need to install the GNU C Compiler (gcc) package on your system. Which yum command will do this?

A. yum gcc

B. yum install gcc

C. yum update gcc

D. yum installpkg gcc

Check on Learning

Question 9

What does the configure script do in an application's installation directory? (Choose two.)

A. It compiles the source code into a binary executable.

B. It checks the local system to verify that the necessary components are available.

C. It copies the binary executable and other files, such as documentation, to the appropriate directories in the file system.

D. It creates the Makefile file.

E. It verifies that the installation files haven't been corrupted or tampered with.

Check on Learning

Question 10

Which action, when used with the dpkg command, uninstalls a specified package and deletes all of its configuration files?

A. –r

B. –p

C. –P

D. –U

Check on Learning

Question 11

You want to use apt-get to download and install the 3dchess package on your Linux system. Which command can you use to do this?

A. apt-get install 3dchess

B. apt-get –d install 3dchess

C. apt-get upgrade 3dchess

D. apt-get –s install 3dchess

Check on Learning

Question 12

Which file is checked by applications on startup for the location of shared libraries on the Linux system?

A. /etc/ld.so.conf

B. /etc/ld.so.cache

C. /lib/ld.so

D. /usr/lib/ld.so

Check on Learning

Question 13

Which type of shared library is integrated directly into an executable file when it is initially compiled?

A. Dynamic

B. Shared

C. Static

D. Linked