Lab 4

Software Management

Outcomes

- Be able to manage software packages on a system running CentOS via YUM.
- Be able to compile and install software from source as root.
- Be able to compile and install software from source with a regular user account.

Rubric (10 points total):
Each of these items requires a checkoff from the TA. Unlike most other labs, there is no electronic submission of this lab; only checkoffs from the TA.

- Add a repository to your system(2 points)
- Compile and install emacs from source as a root user(3 points)
- Compile and install emacs from source as a non-root user(5 points)

Procedure

Most of you are familiar with software package installation. You've almost certainly installed software on your personal laptops or desktops before. If you are a windows user, you probably installed the aforementioned packages using a 'exe' file, or if you are a Mac user, you installed them using a 'dmg' or the app store. Regardless of the installation file used, the installation process is usually pretty straight forward; you interact with a graphical interface and answer a few questions during the installation process. Removing, or uninstalling, software is similar to the installation process.

Traditionally, installing and managing software on a Linux system was a little more involved than downloading a 'exe' and just clicking on it . However, over time distributions like CentOS and Ubuntu, have made managing software as easy as in the previous examples. This is fine for a desktop system, but not for a server system, since you will most likely not have a graphical interface and you will want to automate installations processes for large number of computers. So, you will have to manage software packages using command line tools.

On a Linux server you will use package-management tools like YUM (Yellowdog Updater, Modified; for Red Hat based systems), and APT (Advanced Packaging Tool; for Debian based systems) to install and remove software packages. These tools download software packages from repositories(more on repositories later) and install them on your system. However, there will be times when the software package you need is not in any of the repositories or the latest version of the software available in the repositories is not the one you need. In such cases you would either tell the package management tool to look in other repositories or you would compile the software from source (more on compiling and installing software from source later).

YUM

Since we are on a Red Hat based system we will be working with YUM. Since Yum is so critical to managing systems, the very first command you should use is:

```
man yum
```

to see what sorts of options are supported. If man is not installed, you will have to install it:

```
yum install man
```

Everything you are about to see/learn in this lab you can find on the man page. However, if you are still unclear about how to use a command you can always do a google search for YUM. Before you continue, you should boot up your virtual machine.

Installing Software Packages with YUM

You probably already installed a few packages on your virtual machine using the YUM package-manager. However, lets go over installing packages anyway. The command to install packages with YUM is of the format:

```
yum [options] package1 [package2 package3 ...]
```

The stuff between the brackets is optional, everything else is required. In the above command you need to specify at least one package.

For practice, lets install two packages:

```
yum -y install emacs wget
```

Notice the 'y' option. After you specify the install command, YUM usually shows you a summary of the packages that is about to install and asks you if you want to continue with the installation. With the 'y' option, YUM does not ask you any questions, it just installs the packages you specify.

'wget' is a very useful tool, that allows to download files from the web from the command line(we are going to use it later).

Suppose that you want to install all the packages necessary for general software development (Java/C++/Fortran/Perl ... etc). This means you will need to install gcc, make (for make files), gdb, glibc, javac, java, ... etc. You can do it with the following command

```
yum -y install gcc glibc make gdb ... etc
```

However, what if the list is incomplete? This is why YUM provides software groups. Software groups just refer to a set of related packages. So, alternatively you can install the tools necessary for a software development environment with the 'groupinstall' command:

```
yum groupinstall 'Development Tools'
```

We will discuss, how you can know to use the name 'Development Tools' in the 'Searching for Software with YUM' section of the lab.

Searching for Packages with YUM

Other than knowing how to install packages with YUM, you also need to know how to search for packages. Let's say that you want to install an enhanced version of the vi text editor but you don't remember the exact name of the package, but you do remember that the title of the package has the word 'enhanced' somewhere. Either

```
yum search editor
```

or

```
yum search enhanced
```

would help you locate the package. The output of both commands would be all packages whose title, description, or summary contain the keyword 'editor' or 'enhanced'. If you search through the results of the queries above, you would find that the package you want is 'vim-enhanced'.

Recall software groups from the previous section. You really can't search for groups per se, but you can list all of the software groups available by using the 'grouplist' command:

```
yum grouplist
```

The names of the groups are pretty self explanatory, and the list generated by previous command is not too long, so you can easily look through it(so it is kind of like a search function). If you want more details on a particular software group, you can use the ' groupinfo' command. For example, if we want more details on the 'Development Tools' software group, we can issue the following command:

```
yum groupinfo 'Development Tools'
```

Other than searching for packages to install, you can also search for packages that are already installed on your system with the 'list' command. The 'list' command can do the same thing as the 'search' command, but you need to know the exact name of the package. As an example, say we want to find out what version of emacs is installed on our system, we would simply type:

```
yum list emacs
```

Now try

```
yum list
```

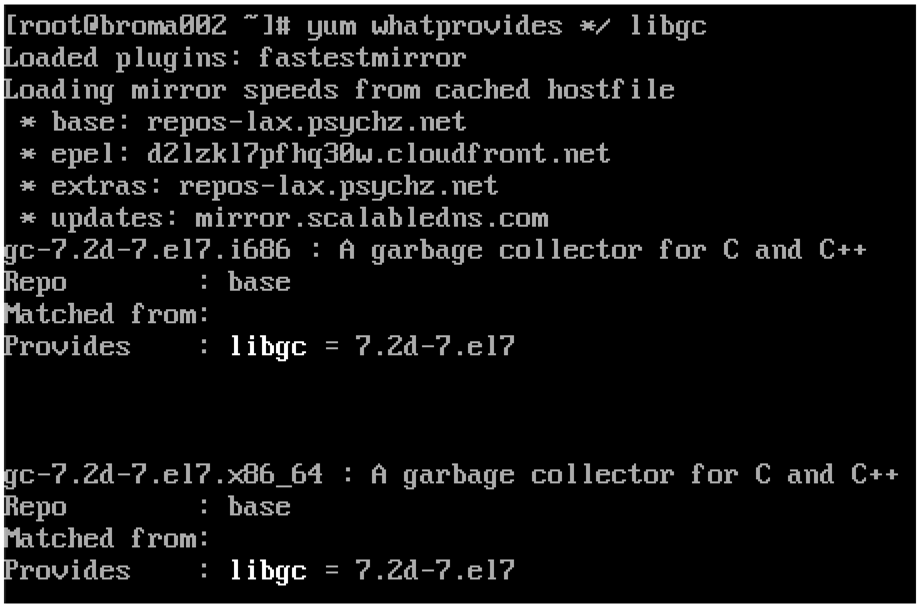
What do you think all the output of the previous command means?

YUM can also help you locate packages that provide a certain file or feature. This is useful when you need to locate a shared object file. For example if an application was giving you the following error:

```
 "error while loading shared libraries: libmysqlclient_r.so: cannot
open shared object file: No such file or directory ".
```

Using the 'whatprovides' command, we can locate the package we need to install in order to get rid of the aforementioned error:

```
yum whatprovides */ libgc
```



```
[root@broma002 ~]# yum whatprovides */ libgc
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: repos-lax.psychz.net
 * epel: d2lzkl7pfhq30w.cloudfront.net
 * extras: repos-lax.psychz.net
 * updates: mirror.scalabledns.com
gc-7.2d-7.el7.i686 : A garbage collector for C and C++
Repo        : base
Matched from:
Provides    : libgc = 7.2d-7.el7



gc-7.2d-7.el7.x86_64 : A garbage collector for C and C++
Repo        : base
Matched from:
Provides    : libgc = 7.2d-7.el7
```

From the output you will see which package provides 'libgc' (don't install the package though).

Updating packages with YUM

Fixes for security issues as well as new program features, will make updating packages necessary. YUM provides you with the functionality to do this with the update command. You can update all the packages, you installed with yum, at once, or just a few. As an example, if you wanted to update the kernel on your system you would type:

For Centos 6:

```
yum update kernel
```

For Centos 7:
https://www.howtoforge.com/tutorial/how-to-upgrade-kernel-in-centos-7-server/

Just type enter when prompted. you wanted to update the entire system, you would type:

```
yum update
```

Don't update the system though, just hit enter.

Removing packages with YUM

Removing packages with YUM is straight forward. The format of the command for removing packages is:

```
yum remove package1 [package2 pacakge3 ...]
```

For example if you want to remove emacs you can type:

```
yum remove emacs
```

Note that you may run into situations where removing a particular package might also remove many of the other important packages on the system! If you see this, it's a sign that the particular package probably is critical to the operation of the system and should not be uninstalled.

YUM Repositories

As mentioned in the introduction to the lab, tools like YUM, download software packages from repositories. However, how does YUM know where the repositories are? YUM consults the files under '/etc/yum.repos.d' to know where repositories are located. With vi or emacs, open up '/etc/yum.repos.d/CentOS-Base.repo'. The 'mirrorlist' variable tells YUM where to look (it is just a URL).

You add repositories to your system so that you can have access to more software packages or more recent versions of them. One example of this is the epel repository. The process for adding this repository is described in the following paragraph.

To add a repository we first need to create a file under '/etc/yum.repos.d/'; the file will contain all the information for the repository that we want to add. Then we need to import the repositories GPG keym(more on GPG keys here). Sites like epel that have automated the process of adding their repository to your system, by providing you with an RPM (BTW, YUM is really just a tool that manages RPM's). So, lets go ahead and add the epel repository. Type the following command:

```
rpm -Uvh https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
```

The previous command installs the RPM from the given URL.

To see whether the repository was successfully added you can check the '/etc/yum.repos.d/' directory and look for a file titled ' rpmforge.repo'. Next, try(notice that there is no 'y' option):

```
yum install freenx-server nxagent
```

you will see that the package comes from the 'rpmforge' repository(look under the 'Repository' column). Just hit enter when prompted, as YUM's default behaviour is not to install packages. At this point, call the TA over to your computer and show him the output of the previous command.

Compile and install software from source

There will be times when you will need to compile software from source. Sometimes the version of the a particular package in any of the repositories is not is too old and is missing features that you or one of your users need. Or, sometimes a package might contain a serious security vulnerability and there is no sign of updates in the repositories.

Installing emacs from source as a root user

For this part of the lab, we are going to compile and install emacs from source. On your virtual machine, make sure you are logged in as root.

We first need to get our hands on the emacs source code. First change directory (cd) to '/usr/local/src'. Next we grab the emacs source code from the GNU site with wget:

```
wget http://ftp.gnu.org/gnu/emacs/emacs-24.4.tar.gz
```

Once the download completes, untar/unzip the file; if you don't know how to this, open the man page for tar. After unzipping the downloaded file, change directory to '/usr/local/src/emacs-24.4'. The first we need to do is to configure the software with the configuration script in the current directory:

```
./configure --prefix=/usr/local
```

The 'prefix' variable tells the configuration script where it should install emacs. In this case we are telling it to install it under '/usr/local'. Note that if we did not specify the 'prefix' option, emacs would still install under '/usr/local' as this is the default. However, I am specifying it here because you will need to use this option later in the lab.

If you get an error about 'Xorg' or 'X', during the configuration phase, read the message carefully, it will tell you how to resolve the issue. If you still can't figure out how to resolve the issue, run:

```
./configure --help
```

and read the options.

After you figure out how to resolve the error, type

```
make
```

to compile the emacs code. Once the compilation phase completes, type

```
make install
```

to install the package. After the 'make install' phase completes, you are done. You just compiled and installed a software package from source. Change directory into '/usr/local/bin' and run 'ls'. You should see an emacs binary. Try running it by typing:

```
./emacs
```

Compiling and installing other software from source, is pretty much identical to what you did here. You always configure a package, then compile it, and then install it. There may be necessary options when you do so - for example, which libraries to exclude, the paths to libraries that the 'configure' script could not find, and so on. To see a list of configuration possibilities for a package, you can run:

```
./configure --help | less
```

Installing emacs from source as non-root user

Now log into your virtual machine with the account you created for yourself:

- Make a directory called 'usr' under your users home directory.
- Follow the instructions from the previous section of the lab and install emacs under '/home/username/usr'(where username is your users home directory), specifying an appropriate prefix option on the configure command.
- Now modify the PATH environment variable to use the version of emacs you just installed. Show the output of `which emacs` to the TA.