**04 – Installing Software**

**Activities**

COMP190 – Tools and Techniques for Software Development

Dickinson College

Spring 2023

**Name:**

Over the last several classes you have learned to navigate the Unix/Linux command line interface and to use a number of the common tools and filters. In this set of activities, you will again use all of those skills and add to them some new ones that allow you to add new software to a Linux installation. Along the way you’ll learn about the PATH environment variable and the administrative privileges that are necessary to manage a Unix/Linux system.

**Environment Variables:**

As we saw in class, Linux shells use a collection of *environment variables* to maintain information that helps them to execute programs and that can be used to customize their behavior. This section explores those variables just a little more.

1. The env command will display “the environment” (i.e. all of the environment variables). Each environment variable has a name, which by convention will typically be in all capital letters (e.g. PATH). There will be an = sign, followed by the value of the variable.

a. Run the env command. Look over the output of the env command and find the environment variable that stores the time zone for the machine (Hint: It’s the one with the value that is a location in the world). What is the name of the environment variable that holds the time zone?

b. Use an echo command to display just the value of the time zone environment variable. Paste a screenshot showing your echo command and its output below. Please do not show the full output of the env command as it will be quite long.

**The Linux/Unix PATH variable:**

2. The PATH is an important environment variable that is used on all Linux/Unix systems. We mentioned the PATH variable in class and the following questions explore the PATH environment variable a little more.

a. Describe the purpose of the PATH environment variable in a sentence or two of your own words.

b. Use an echo command to display the value of your PATH environment variable. Paste a screenshot of the command you used and its output below.

c. Based on your answer to part b, indicate which of the following directories are “on the path” (i.e. listed in the PATH environment variable) on your Linux system?

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | **Directory** | **On PATH (Yes/No)?** |  |
|  | /usr/bin |  |  |
|  | /home/student/scripts |  |  |
|  | /usr/local/bin |  |  |
|  | /usr/local/games |  |  |
|  | /sbin |  |  |
|  |  |  |  |

**Extending the PATH:**

The value of PATH that you saw above is the default value that is set by your Linux system. Sometimes you will want to add additional directories to the PATH, so that your shell will be able to find programs or scripts that you place in that directory. For example, in HW03 you created a scripts directory within your home directory and wrote your ./countdown.sh script in that directory.

3. In this question you’ll create a new script and then you’ll extend the PATH so that you can run this script from any working directory.

a. Environment variables and the echo command can be used in shell scripts as well as on the command line. Use an editor (nano, Mousepad, VSCodium) to create a new file in your scripts directory named myenv.sh and place the following content into it:

#!/bin/bash

echo "My home is "$HOME"."

echo "My working directory is "$PWD"."

echo "My PATH is "$PATH"."

*Nothing is required here. You just need to be sure to create the script.*

b. Set the permissions on the myenv.bash file so that the user (i.e. the owner), the group and all others can all execute this script. Use the ls -l command to verify that you have set the permissions correctly. Paste a screenshot showing the commands that you used and their output below

c. Try to run the script that you just created using the command myenv.bash. Your script will not run and a “command not found” error will be displayed. Briefly explain why the myenv.bash command was not found.

Note: When you ran your scripts in HW03, you used something like ./myenv.bash. The ./ at the beginning of the command is a relative path that tells the shell to look in the current directory for the command. So if you use ./myenv.bash here it will work too.

d. Let’s now update the PATH environment variable so that it knows where to find the command myenv.bash. The following command will add your scripts directory to the PATH environment variable:

 PATH="$PATH:/home/student/scripts"

Use this command and then use echo to check that the value of the PATH has been updated. Paste a screenshot of the commands you used and their output below.

e. Now try the command myenv.bash again. It should work now. In fact, it should now work from any working directory. Try the command myenv.bash from your home directory instead of from the scripts directory. Give a screenshot of the commands you used and their output here.

It is important to note that the command you just used to add /home/student/scripts to your PATH only has an effect in the current terminal window. To see that this is true, try opening another terminal window and using echo to display the PATH variable. Notice that it has the original value.

If you want to make permanent changes to the PATH variable you have to change some of the Linux configuration files. It is not required viewing, but if you are interested in how you can permanently change your PATH You can check out the short video *LINUX BASICS IN 3 MINS : $PATH ENVIRONMENT VARIABLE:*

* <https://www.youtube.com/watch?v=abN6bvyPRxQ> (3:00)

**The Super User:**

Because Unix/Linux is a multiuser operating system it is necessary to ensure that not every user is able to modify the system configuration. In Unix, the user named root is the *super user* and has full administrative privileges to the machine. That is root may perform any operation on any file anywhere on the system. Thus, the root user has incredible power, but like any such power, it should be used sparingly. To facilitate the use of root powers only when they are needed, Linux/Unix systems provide the sudo command. The sudo command grants a user temporary administrative (i.e. root) privileges on a command-by-command basis.

It is not required viewing, but you might find the video *How to use the sudo command: 2-Minute Linux Tips* helpful in getting a feel for how sudo is used.

* <https://www.youtube.com/watch?v=msTXNYhFtTU> (2:27)

4. In order to run commands as root using sudo, your user (i.e. student) must be a member of the sudo group. The groups <username> command will display the groups to which the specified <username> belongs. Use this command to display the groups that your user belongs to. Give a screenshot of your command and its output.

5. Now that we know you can run programs as root using sudo, let’s do a little experiment to confirm that sudo does in fact run commands as the super user (i.e. as root).

a. Use the command man whoami to display the man page for the whoami command. What does the whoami command do?

b. Run the whoami command twice. Once normally at the command prompt as your student user and once as the root user by using sudo whoami. When running the command using sudo you will be prompted for your password. The password for the student user is also student. Paste a screenshot of the commands you used and their output below.

c. Explain in a sentence or two how the output of your commands in part b demonstrates that the sudo command runs other commands as the super user (i.e. as root).

6. The /etc directory on a Linux/Unix machine contains many of the configuration files that control how the system operates and who is allowed to do what. Many of these files contain sensitive system wide configuration information. To protect the system, only authorized system administrators (i.e. root) should be allowed to modify these files. Normal users (i.e. anyone who is not root) should not be allowed to modify (or sometimes even see) these files.

This question explores how a user with the right privileges can use sudo to read and (possibly) edit files that their normal user (e.g. student) is not able to.

a. Make /etc your working directory and use the ls -l command. Find a file for which your user (student) does not have read permission. There are 5 of them… just pick one. If you are not finding any, try reviewing the meaning of the permissions we learned about last week.

Give a screen shot of the line of output for the file you have chosen. Please do not paste the entire ls -l output.

b. Try to use cat to display the file that you chose in part a. This should result in a “Permission denied” error. Give a screenshot of your command and its output.

c. Now use sudo to run cat as the root user to display the file you chose in part a. Briefly explain why the contents of the file are now displayed.

**Installing Software with a Package Manager:**

You now know that the sudo command allows you to run commands with super user privileges. For example, as you just did above. Or if you were to need to edit a file that requires root privileges for writing you can simply run the editor using sudo (e.g. sudo nano). However, probably the most common use of sudo is to install new software. This section shows you how to install software on a Debian Linux system using a package manager with sudo.

While many Linux distros come with a lot of good software, anytime you are doing any serious development work you are going to need to regularly install new software and update old software. This section looks at how to install new software using a *package manager*, which is the primary mechanism by which software is installed and updated in Linux.

In this section, you’ll use the command line Advanced Packaging Tool (apt) to install some software. The apt is used by Debian (and many of its derivatives, e.g. Ubuntu) as their package manager. Other distros will used different package managers (e.g. yum, apk, snap, etc.). These all do the exact same types of things, they just use slightly different commands. So, the types of things you learn to do in apt can also be done in any of the other package managers as well. So, if you come across them in the future, you’ll just have to learn their specific commands.

It is not required viewing, but if you would like to hear more about package managers and the apt before you get started, you can watch the video *Package Management Basics* from LinkedIn Learning:

* <https://www.linkedin.com/learning/linux-tips-weekly/package-management-basics> (6:59)

7. The apt relies on external sites called *package repositories*. These package repositories provide *installation* *packages (or just packages)* for all of the different software that can be installed. These packages also contain information about the dependencies that are necessary for the software to work correctly. Package repositories are updated frequently with new versions of software and new dependency information. So, the first step in using apt should always be to update it with the latest information about the packages that are available for installation. The following command updates apt with the latest information from the package repositories:

sudo apt update

Run this command. Give a screenshot of the command and its output below.

8. Now that our package information has been updated we can install some software. Our Linux system does not currently have Java installed, but it might be nice to have it. So, let’s fix that by installing Java.

a. Type the command java --version on the command line. This should confirm that Java is not installed. Give a screenshot of your command and its output.

 b. The command to install a new piece of software with the apt is:

 sudo apt install <package>

Where <package> is the name of the installation package for the software. The package for the Java Development Kit (JDK) is named default-jdk. Use apt to install the Java JDK package. Give the full command that you used.

You will notice that when you use your command, apt will give you a lot of information about all of the additional and new software packages that are to be installed. All of these are packages contain dependencies that are required by the default-jdk package. This is one of the great things about using a package manager, it ensures that all of the necessary dependencies are also installed. Just imagine the misery of having to install all of those dependencies one-by-one by hand before you could install Java!

c. When the install has completed use the java --version command again to confirm that Java has been installed. Take a screenshot of this command and its output and paste it here.

d. Let’s check that your new JDK works by making a simple Java program, compiling it and running it as follows:

* + Use a text editor and enter a basic “Hello World” program:

class HelloWorld {

 public static void main(String[] args) {

 System.out.println("Hello comp190!");

 }

}

* + Save the program into a file named HelloWorld.java.
	+ Compile the program: javac HelloWorld.java
	+ Run the program: java HelloWorld

Give a screenshot showing the commands that you used to compile and run the program and their output here.

**Finding Packages:**

Often the trick with using apt is knowing the name of the package that you need to install to get the software that you want. Fortunately, there are a few ways to search for packages that contain what you want.

* You can browse the packages by category on-line:
	+ <https://packages.debian.org/bullseye/>
* You can search using an on-line search feature.
	+ <https://www.debian.org/distrib/packages#search_packages>
* You can use your favorite search engine and search for something like:
	+ Debian apt install <what I want to install>
* You can search using the apt program.
	+ We’ll learn how to use this one here, because I know you can use a search engine, and this one can be quite useful.

9. The apt search <string> command will search for packages that contain <string> in their description. There is a classic Unix program called cowsay that is silly but fun (at least for a few minutes).

a. Use the apt search command to find the packages related to cowsay. Give a screenshot of the command you used and its output here.

b. In your output above the package names will be highlighted in green. What three packages are related to the cowsay program?

c. Use the apt to install the package for the “graphical” version of the cowsay program. What command did you use? Do not give the output that is generated.

10. The apt will install the program into the /usr/games directory.

a. Check the files in that directory. What are the names of the five programs that were installed?

b. The xcowsay "<string>" command will run the program where the string is what you would like the cow to say. Try to run the command:

xcowsay "Hey Comp190!"

This command will not work. Give a screenshot of your command and the output that is generated.

c. Briefly explain why you see the error that you do in part b. Hint: See question #3(c).

d. When a program is not on your path you can use absolute or relative paths to run the program. Make ensure that your home directory is the working directory. Then complete the following two questions.

i. Run the xcowsay program by using an absolute path and have the cow say hi to you by name. Cute and silly, right? ☺ Give the full command that you used below.

ii. Repeat part i, but using a relative path this time. Give the full command that you used below.

11. Using absolute or relative paths for frequently used programs can be annoying, and I know you want to use xcowsay all the time… so let’s make it better.

Give a command that will add /usr/games to your PATH environment variable (Hint: See #3). Use echo to display your PATH variable to ensure that /usr/games has been added. Finally, run the xcowsay program without using a path to test that it works.

Give a screenshot of all of the commands that you used and their outputs.

Disclaimer: The xcowsay program will only say what you tell it to say. However, that package also installs the fortune package and the xcowfortune program, which display fortune messages. The maintainers of that package have attempted to move fortunes containing potentially offensive material out of the main database for the program. Their judgement may not be perfect. To avoid any risk of offense, please do not run the fortune program or the xcowfortune program.

**The Synaptic Package Manager:**

As you know, usually when something can be done on the CLI it can also be done using a GUI. Installing software is no different. So, it won’t be surprising that it is possible to use the apt through a GUI as well. Here you’ll use the Synaptic Package Manager to install a game of your choice, just to get a feel for how it works.

12. The GUI for the apt is called the Synaptic Package Manager.

a. Launch the Synaptic Package Manager using the command: synaptic.

Note: A few WARNING messages will appear in the terminal. These are safe to ignore.

What does the dialog box that appears tell you?

b. The solution to the problem described in the dialog box is to run the Synaptic Package manager as root so that it has the privileges that it needs in order to install software - just like when you used apt install on the command line.

However, before that will work there is one little hack you have to perform because our machine was not quite configured correctly (sorry ☹). Run the following command in your Terminal:

export XAUTHORITY=$HOME/.Xauthority

Then run synaptic as root give a screenshot of the command that you used. Again, you can safely ignore the WARNING messages that appear.

c. Read the information in the dialog box that appears. If you don’t see a dialog or closed it, choose “Quick Introduction” from the “Help” menu. What is one way that you can mark a package for installation?

d. Browse the sections (i.e. categories) of packages (top left) and find the “Games and Amusement” category. Browse through the available game packages and find one that appeals to you. Note: The package names are not always that descriptive, but if you click on the package name, more information about the package appears below. Once you have chosen a game, mark its package for installation. Then click the “Apply” button to install it.

What game did you install and what is the name of its package?

e. All of the packages in “Games and Amusement” are installed into the /usr/games directory. Examine that directory to find the name of the program(s) that the package installed. Run the game that you installed.

* Note: there are a few of the “Games and Amusement” packages that do not work. If you are unlucky and have chosen one of those, go back to part d and try another.

Give a screenshot of the command that you used.

f. Give a screenshot of your game running.

**Summary:**

13. Complete the table below by filling in the Linux command or symbol that corresponds to each task. Use the <…> and […] notation as appropriate to make your commands general.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | **Command** | **Task** |  |
|  |  | Display all of the environment variables. |  |
|  |  | Write text to standard output. |  |
|  |  | Display the PATH environment variable. |  |
|  |  | Display the absolute path to a program on the PATH. |  |
|  |  | Display the groups in which your user is a member. |  |
|  |  | Update the package information for apt. |  |
|  |  | Install new software with apt. |  |
|  |  | Remove an installed package with apt. |  |
|  |  | Search for a package using apt. |  |
|  |  | Run the Synaptic package manager. |  |
|  |  | Compile a java program. |  |
|  |  | Run a java program. |  |
|  |  |  |  |

**Optional:** To help us improve and scope these activities for future semesters please consider providing the following feedback.

a. Approximately how much time did you spend on this activity outside of class time?

b. Please comment on any particular challenges you faced in completing this activity.

**Acknowledgements:**

Some materials, questions and resources have been adapted from activities posted on foss2serve.org.

* [http://foss2serve.org/index.php/Linux\_Package\_Management\_(Distribute\_Your\_App)](http://foss2serve.org/index.php/Linux_Package_Management_%28Distribute_Your_App%29)
* <http://foss2serve.org/index.php/Fedora_Install_Activity>