**03 – Unix Tools and Filters**

**Activities**

COMP190 – Tools and Techniques for Software Development

Dickinson College

Fall 2021

**Name:**

In the article *Linux vs. Unix: What’s the Difference*, Phil Estes summarized the Unix philosophy as “utilizing small, purpose-built programs in combination to do complex overall tasks.” In this assignment you’ll learn about a number of these “purpose-built” programs that come on every Unix/Linux system. You will get some practice with them and also learn about techniques including *piping* and *redirection* that allow you to combine them to complete those more complex tasks.

As with A02, you will be following along some sections of Ryan’s Tutorials. Each of the sections below points you to the relevant tutorial and I have included headers within each section to connect them to the sections of the tutorial. Use these to clue you in to the sections of the tutorial where the material relevant to the question is. That doesn’t mean the other sections are not important or can be skipped. You should work each tutorial from the start to get a full picture of how to use the Unix/Linux tools. As with A02 I highly recommend that you try each of the things in the tutorial as you read through it, play around and try some other things and then answer the questions that I pose.

**Text Editors:**

There are a lot of different text editors available for Unix/Linux. You found a nice GUI editor at the end of A01 to open and type your name into to show that you had completed the assignment. However, many times a Unix/Linux machine is accessed only through a terminal (e.g. connecting to a cloud server through Amazon Web Services (AWS), Google Cloud, Microsoft Azure, Heroku or some other service). In those instances, you’ll need to know about command line editors that work entirely in text mode. There are several of these. The section of Ryan’s Tutorials introduces you to vi, which the most common and is the only one guaranteed to be on every single Unix/Linux machine you encounter. I will point you toward a few others that are a little easier to pick-up as well.

Open the “Vi Text Editor” section of the tutorial:

* <https://ryanstutorials.net/linuxtutorial/vi.php>

*A Command Line Editor*

1. The Vi editor has two modes of operation. What are these modes? What is each used for?

2. How do you switch from edit mode to insert mode?

3. How do you switch from insert mode to edit mode?

*Saving and Exiting*

4. Which mode do you need to be in to save a file?

5. What sequence of three keys must be pressed to save a file without exiting?

6. What are two different ways to save the file and exit Vi?

*Navigating in Vi*

7. What happens if you press the arrow keys while in insert mode?

8. What two Vi commands could you use to move to the first line of a file? Use google to find the second one.

*Other Ways to View a File*

9. What does the cat command do?

On all Unix/Linux machines there is a directory named etc in the root (/) directory. The etc directory contains files that hold information that the operating system uses for configuration and operation.

10. Give a cat command that uses an absolute path to display the contents of the file named passwd that is in the etc directory.

11. The passwd file contains a list of all of the users that are defined for the system. You should see your username near the bottom of the file. Next to it you will see information about your user including the path to your home directory and your default shell. Copy the line of text for your user from the cat output as the answer to this question.

*Deleting Content*

12. In Vi, the x command deletes the character under the cursor. What does the x command do when the line the cursor is on is blank? Hint: Try it!

13. What is the Vi command to delete the current line of text? This is really useful when you start pressing the arrow keys in the insert mode (like I do all the time!).

**Okay that’s Enough!**

As Ryan suggests, if you really train yourself to use Vi it can be extremely efficient because you can do everything without your fingers leaving the keyboard. I know a number of very good computer scientists / programmers that use Vi exclusively. I however am not one of them. I can get a file edited with it when I need to but generally prefer Pico or Nano. Which are other, easier to use, command line editors. One of them is *almost guaranteed* to be on every Linux/Unix system. These two are very similar and often times a system will only have one or the other, but either command will run it. That is the case on LinuxLite, which has the Nano editor but typing either pico or nano will run it.

14. Run nano from the command line. The menu of commands appears at the bottom of the screen. The ^ character indicates that the control key should be held down while typing the next character. What are the commands to save a file (i.e. write it out to disk)? Open a file (i.e. read it in from disk)? And exit nano?

Play around with nano a bit. Notice that it behaves kind of like you would expect. You still can’t use the mouse, but at least it doesn’t have separate insert and edit modes and you can use the arrow keys to move around anytime. This is my go-to editor when I’m stuck on the command line and need to get something done. That is unless it doesn’t exist… then knowing that Vi is out there and knowing you can look up its obtuse editing commands can be a life saver!

**Filters:**

*Filters* is another name for the “purpose-built” programs mentioned in the Unix philosophy. These programs come with every Linux/Unix system and form the basis for getting complex work done efficiently from the command line. In this section of the tutorial you will get familiar with some of the filters. Then in later sections you’ll see how to combine them to do complex sequences of operations.

Open the “Filters” section of the tutorial:

* <https://ryanstutorials.net/linuxtutorial/filters.php>

This section of the tutorial runs you quickly through a bunch of the different Unix/Linux filters that can be used to manipulate data. The point is not to become an expert at any of them, but rather to introduce you to a bunch of the most useful ones so that you know they are there. Then later when you have a real problem to solve, you’ll know what tools are available. You can then look up the details of how to use the one(s) that you need to solve your problem. So, don’t get too hung up on trying to memorize all of the details here. Just gain some experience and try to remember what tools are available and what they do.

*So What Are They?*

15. What type of data is processed by filters? What does a filter do to that data?

One thing that is important to know is that filters do not change the files. As we will see later, there are ways to change the contents of files using the output of filters. But when just running the filters, they will simply display the output in the terminal without modifying the file. This means that it is safe for you to experiment with the filters and you do not need to worry about corrupting anything in the system.

*Head*

In your home directory there is a hidden file named .bash\_history. This file contains a list of the commands that you have used. It is where the commands come from when you use the up and down arrows at the command line. Note however, that it is only updated when the shell is closed so it will not contain the commands you have used in the current session.

16. Give a head command that displays the first five lines of your .bash\_history file.

*Tail*

17. Give a tail command that displays the last ten lines of your .bash\_history file.

18. Are more recent commands at the top or bottom of your .bash\_history file? Hint: Quitting the terminal and restarting it will put your most recently used commands into the bash history.

*Sort*

19. Give a command that will display the passwd file used earlier in reverse dictionary order (i.e. z’s at the top of the output). Hint: Use the man page for sort to figure out what command line flag you need to include to reverse the sort order.

Remember that using filters does not modify the files. So, if you use cat to look at the passwd file after doing your sort, it will still be in its original order.

*WC*

20. Give a command that will display the number of different user accounts that exist on the system. Hint: There is one line for each user in the passwd file.

*Cut*

21. Give a command that will display just the username and home directory for each of the user accounts that exist on the system. Hint: This information is in the passwd file and the different fields are separated by ‘:’ characters.

*Sed*

22. The etc directory contains a file named “legal”. Use cat to have a look at that file. Then give a command that will display the contents of the legal file where the name of the Linux distribution (i.e. “Ubuntu”) is replaced by “LinuxLite”.

Again, remember that using filters (e.g. sed here) does not modify the files. So, if you use cat to look at the legal file after doing the find and replace with sed, it will still have the original content. We’ll see later how we could replace the contents.

**Grep and Regular Expressions:**

The Unix/Linux grep program (i.e. filter) is one of the most powerful command line tools you can learn. Like the other filters, don’t try to master or memorize everything about grep. If I haven’t used it in a while, I still have to look it up to remember just exactly how to use it. So, the point here is to get some practice using it so that you are familiar with the types of things that grep can do. That way later, when you have a problem for which grep will help, you’ll know it exists and have a basis for learning more about it.

Open the “Grep and Regular Expressions” section of the tutorial:

* <https://ryanstutorials.net/linuxtutorial/grep.php>

*So What Are They?*

23. What is a *regular expression* and what are they typically used for?

*eGrep*

Note that egrep is an extended version of an older program named grep. Many Linux/Unix systems (including LinuxLite) now combine grep and egrep and either command will run a program compatible with either. The syntax for using grep and egrep are slightly different and egrep is able to do more than grep. On Linux Lite, you can run the original grep using the command grep or egrep. To run egrep on Linux Lite you can use either the command grep -E or egrep -E.

I recommend for this section that you create the sample file that he uses so that you can try out the commands for yourself. I also suggest experimenting with variants on the examples. Think up something similar to what the given command does and then see if you can get it to do what you’ve thought up. For example, he searches for ‘mellons’ see if you can predict what happens if you search for ‘ears’ or how you could search for all the fruits that begin with ‘p’.

24. Give a command that displays the lines from the passwd file for users that have /bin/bash as their default shell.

25. Give a command that displays the number times you have used the ls command recently. Hint: Use the .bash\_history file mentioned earlier.

*Some Examples*

Be sure to skim the Regular Expression Overview section before going onto this section. Don’t try to memorize all of the building blocks in that section, but know they are there and generally what characters they use. Then you can return to the Overview section to use it as a reference at any time. Review the examples given and be sure you understand why they do what they do. Give them a try using the sample data and experiment with them.

26. Consider each of the following regular expressions. Each expression is followed by a number of lines of text. For each line of text, indicate whether the line of text will be matched by the regular expression or not. Hint: Create test data files and try the commands to check your answers. Also, you will need to use egrep (i.e. the -E flag) for these exercises. Finally, Word uses ‘smart quotes’ that are angled because they look better, but the linux command line requires simple normal ﻿'single quotes﻿'. So, if you copy and paste the regular expressions be careful to correct the smart quotes ( ‘ ) to normal single quotes ( ﻿' ) or you will get errors.

a. ‘.+[0-9]’

i. 7 is a digit.

ii. 77 is a number.

iii. This line has 4 words.

b. ’^[01]{8}$’

i. 01001110

ii. 0000000000

iii. 01010101

iv. Binary 00001010 is 10 decimal

v. 00001010 is decimal 10

vi. 10 decimal is binary 00001010

c. ‘ [A-Z|a-z]\*[@#$&%][A-Z|a-z]\* ’ (Note the spaces.)

(Note: There is no space at the start or end of the following lines)

i. &$%

ii. Does T#is match

iii. What about Th@t

iv. Or maybe T#i$ does

v. Just @ne more

27. The file /etc/shells contains a list of all of the shell programs that are valid to be listed as the default shell for a user. Give a grep command that selects only the shells that reside in the /bin directory (i.e. the bin subdirectory of the root directory). Hint: Use cat to look at the contents of the /etc/shells file to get started.

28. Give a grep command that selects all of the lines from the passwd file for users with usernames (the first field) that have less than 5 characters.

**Piping and Redirection:**

Now that you have some basic Unix/Linux filters in your toolbox, this section of the tutorial is when you really get to really put the Unix philosophy of combining small purpose-built programs (i.e. filters) together to accomplish more complex tasks into practice.

Open the “Piping and Redirection” section of the tutorial:

* <https://ryanstutorials.net/linuxtutorial/piping.php>

*So What Are They?*

29. What are the three data streams that are connected to every process? What is the purpose of each?

*Redirecting to a File*

30. Give a command that will write the first 10 lines of your bash history to a new file named MyCommands.txt. Hint: Check your answer by opening your .bash\_history and the new file in an editor MyCommands.txt.

31. Give a command that will append the last 10 lines of your bash history to the end of your MyCommands.txt file from the previous question. Be sure to check your answer.

*Redirecting from a File*

32. Give a command that *redirects* your MyCommands.txt file into grep and displays on the screen only the cd commands that you have used.

33. Give a command that does the same thing as question 32 but also redirects the output into a new file named MyCDcmds.txt.

*Piping*

34. Give a command that outputs a reverse sorted list of the usernames of all of the users on the machine. Hint: Use the /etc/passwd file with cut and sort. Check your answer by using cat to display the passwd file.

35. The file /usr/share/dict/words is known as the Unix dictionary file. Really it is just a long list of words in a text file. This file is often used for things like spell checking on Unix/Linux systems. Give a command that will display the second 10 words in this dictionary that start with the letter m and contain 8 or more letters. Hint: use grep and head and tail. Hint: Check your answer by opening the words file in an editor.

**Challenges:**

The following several questions ask you to produce a command line using filters, piping and redirection to accomplish a specific task. They use the programs that appeared in the tutorials but also require that you learn a little bit more about them. For each challenge the solution can be accomplished with a single command line statement. If you have already spent more than 3 hours on this assignment, skip these challegnes.

36. Give a command that creates a file named Emails.txt that will contain a sorted list of email address that have been generated for each user on your machine. Each individual’s email will be their username (from /etc/passwd) with the string @comp190.org appended to the end of it. So, if your machine has a user named ahmad then there should be an e-mail ahmad@comp190.org in the file. Hint: The sed command that we used earlier for find and replace can also append to the end of strings. Use your favorite search engine to find an example or two.

37. Give a command that creates a file named UniqDirs.txt that will contain a sorted list of all of the directories that you have used cd to change into recently. Each directory should only appear in the list once. Hint: You’ll need to understand the uniq command. There is some useful information about it in its man page.

**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.