HW-3-C-UNIX

CS 503 Systems Basics

100 Points

What To Do

This assignment has two parts: Part 1 asks you to use simple UNIX commands and Part 2 asks you to use the debugger.

Part 1 (75 points)

Part 1-a: Using the man Command

1. Use the man command to get help for the "ls" command. What command did you type?

   man ls

2. What ls option displays the size of files in blocks (other than the -l option, for long listing)?

   ls -s

3. Issue the command:

man -k date | grep "(1)" to list user commands having to do with dates. Select one of the listed commands and use it to get the current date and time according to the computer. Which command did you use to get the system date and time?

   date

4. Find commands that show who is logged onto the system you are on. What commands will do this?

   who
   w
   last (retrieves users who were last logged on)

Part 1-b: Dealing with Files 1. Issue the command: cd to make sure you are in your home directory. Then issue the command:
cd CS503 to go to the CS503
subdirectory.

2. In the current directory, list all files in long format. What different types of files do you have?

drwx--x--x 2 alc466 domain users 8 Jan 23 19:35 HW2
drwx--x--x 2 alc466 domain users 2 Jan 23 20:25 HW3

I have 2 folders, one for my homework 2 and another for my homework 3.

3. The touch command can be used to create new, empty files. Use the following command to create a file named sky:

touch sky

What command will show you if you succeeded and the file sky now exists?

ls

4. Attempt to create three empty files with the commands:
touch baseball bat
touch -bigfile
touch chili&beans

What error messages occur and are any files created?

1. No errors. Baseball and bat were created as separate files

2. touch: invalid option -- 'b'
   Try 'touch --help' for more information.

3. [1] 1068739
   chili: command not found
   Command 'beans' not found, did you mean:
   command 'beads' from deb beads (1.1.20-3)
   Try: apt install <deb name>
   [1]+ Exit 127 chili

5. When would you use "cat"? When would you use "more"?

"cat" stands for concatenate and is used when you wish to write content to std.out and to files. "more" is a paging utility that is used to display a file one screen at a time. This is useful when working with large files.

6. How many lines in big.file contain the character string "tcsh"? What command did you use?

   I counted 0. grep -c "tcsh" big.file

7. Check your quota status. What command did you use? Are you close to either your quota or your limit?

   quota -v

   I am not close to my quota because there weren't any quotas set.

8. What is your disk usage in the current directory and below? What is the command you used to check it?

   My disk usage is 3.7 MB. I used du -sh to check it.

Part 1-c: The File System

1. Return to your home directory. What command will do this?

   cd

2. What is the absolute path to your home directory? What command displays this?

   pwd

3. Create three new directories in the CS503 directory: "Letters", "Programs", and "Misc". What command(s) did you use?

   mkdir Letters Programs Misc

4. List the files in the directory "/bin" that end in "sh". What command did you use?

   ls /bin/*sh

5. What command lists the files in the current directory that begin with upper case letters?

   ls [A-Z]*

6. Copy all files in the current directory whose names contain the character string "let" into the subdirectory "Letters". What command did you use?

   cp *let* Letters

7. Copy all files in the current directory whose names end in ".c" or ".h" into the subdirectory "Programs". What command did you use?

   cp *.[ch] Programs

8. Copy all files in the current directory whose names contain the character strings "notes" or "misc" into the subdirectory "Misc". What commands did you use?

   cp *notes* *misc* Misc

9. Change the following files to have the specified permissions (use chmod with symbolic permission specs and use ls to check your success):

File	Permissions	Symbolic Mode Command
pp1	rwxrwxrwx	chmod 777 pp1
pp2	rwxrwxr-x	chmod 775 pp2
pp3	rwxr-xr-x	chmod 755 pp3
pp4	r-x------	chmod 500 pp4
pp5	r--r-----	chmod 440 pp5
pp6	rw-r--r--	chmod 644 pp6
pp7	r--r--r--	chmod 444 pp7
pp8	rw-rw-rw-	chmod 666 pp8
pp9	rwx------	chmod 700 pp9

Part 1-d: Redirection

1. Execute the following command:

ls > output.1

What are the contents of "output.1"?

output.1 will display the names of all the files and folders listed in the current directory.

2. Execute the following commands:

ls > output.2
who > output.2
ps > output.2
Use more to check the contents of "output.2".
What is there? Why?

output.2 will contain only the processes that are running because here, the > redirect operator is writing the operation to the file. If the file already exists, any subsequent operations will be overwritten. So, here it only stores the result of the ps command.

3. Now execute the following commands:

ls > output.3
who >> output.3
ps >> output.3
What is the contents of output.3?

output.3 will contain the list of files and folders in the directory, the users that are currently logged on, and running processes. Here, the >> operator is appending the operation to the end of the file file. Even if the file already exists, any subsequent operations will just get added.

4. Begin with a single command, "ps", with the options for a full listing of every process.

ps -fe

5. Use more, to display the output one screenful at a time.

   ps -fe | more

6. Add grep root to filter for the string root. The standard output of ps goes into the standard input of grep. The standard output of grep goes into the standard input of more.

   ps -fe | grep root| more

7. Add sort so the output gets sorted.

   ps -fe | grep root| sort| more

8. Are all of the above processes owned (run) by root? Why?

   The above processes are not all run by root because grep root will display all processes as long as the name contains “root.”

9. Find a "ps" option that limits the display to one user and use it to re-write the pipeline to display only processes owned by root.

ps -u root | sort | more

Part 2 - GDB (25 points)

1. Access the GDB manual or download the pdf file and read it, starting from this page
   a. https://www.gnu.org/software/gdb/documentation/

2. Program etox.c should approximate e^x but has an error that makes it execute incorrectly
   a. First, simply compile and run the program to observe the incorrect behavior.
   b. Second, recompile it to support debugging with gdb and use the gdb commands to identify the source of the incorrect behavior.
   c. Third, fix the errors and run the program to observe the correct behavior.
   d. Since the program is very simple, you may be able to identify the errors just by looking at the .c source code. Still, go through the process of debugging to verify that you are familiar with gdb.
   e. Record all the gdb activity and include it in your lab report together with the corrected .c program.

```
(gdb) next
40          return(value);
(gdb) next
41        }
(gdb) next
main () at etox.c:22
22          printf("e^x = %14.10lf\n",series);
(gdb) print series
$34 = 18.5
(gdb) next
e^x =  18.5000000000
24          return(0);
(gdb) next
25        }
(gdb) next
```

```
This program calculates e^x
using sum of (x^k)/k!
Enter x, n : 5 2
x,n =   5.0000    2
e^x =  18.5000000000
```

```
Enter x, n : 5 4
x,n =   5.0000    4
e^x =  65.3750000000
[Inferior 1 (process 1373159) exited normally]
(gdb) exit
```