

In this homework, you will need to work on the eniac-l.seas.upenn.edu server. If you need help logging in to this machine see the following webpage: <http://www.seas.upenn.edu/cets/answers/remote.html> (note: you need to swap eniac-l.seas.upenn.edu for eniac.seas.upenn.edu).

In some questions below, you will be asked to enter some command with some set of arguments. Depending on the command, there may or may not be output generated. For each command, you should use your mouse to highlight the command (including the prompt) and the output (if there is any) and copy it to a separate text or word document. Obviously, when you make mistakes, you should not copy the text. You should also copy the question to the text or word document as well.

When you are finished with the assignment, attach the text or word document to an email and send to *mcorliss@cis.upenn.edu*. The subject of the email should be "cse399 - hw2" (without the quotes). The homework is due by the beginning of Monday's class. Total points: 50.

1. [14 Points] Piping and redirection.

- (a) Change your location to the 'cse399' directory you created in homework 1. Copy the directory /home1/m/mcorliss/teaching/cse399/hw2 and all its subcontents to your current location. Change your location to this new directory (hw2).

Answer:

```
bash$ cd cse399
bash$ cp -r /home1/m/mcorliss/teaching/cse399/hw2 .
bash$ cd hw2
```

- (b) Below is one approach to sorting input from a file and printing the first 10 lines of the sorted input.

```
prompt$ sort file > newfile
prompt$ head -n 10 newfile
```

Write one command to do this (hint: using piping).

Answer:

```
bash$ sort file | head -n 10
```

```
%!Adobe-FontList 1.01
```

- (c) Print lines 50-75 of the file 'file' and sort them.

Answer:

```
bash$ head -n 75 file | tail -n 26 | sort
```

```

%BeginFont
%BeginFont
%EndFont
%EndFont
DataFormat:Plain
DataFormat:Plain
FamilyName:Helvetica
FamilyName:Symbol
FontName:Helvetica
FontName:Symbol
FontType:Type1
FontType:Type1
FullName:Helvetica
FullName:Symbol
Handler:DirectoryHandler
Handler:DirectoryHandler
OutlineFileName:/usr/openwin/lib/X11/fonts/Type1/Helvetica.pfa
OutlineFileName:/usr/openwin/lib/X11/fonts/Type1/Symbol.pfa
UsesStandardEncoding:no
UsesStandardEncoding:yes
WritingScript:Roman
WritingScript:Roman
isCFF:no
isCFF:no

```

(d) Use 'cat' to send input from the keyboard to the file 'foo' (using one command).

Answer:

```
bash$ cat > foo
```

(e) Redirect all output (including errors) from the command 'ls *.txt *.jpg' to files.

Answer:

```
bash$ ls *.txt *.jpg 1> out.txt 2> err.txt
```

(f) Why is there no output from the command 'head -n 20 file | tail -n 10 > newfile | sort'?

Answer:

Because the output from the 'tail' command is redirected to 'newfile'. Consequently, the 'sort' command receives no input and generates no output.

(g) Why is there output from the command 'head -n 20 file | tail -n 10 > newfile | sort < file'? Why is the output more than 10 lines?

Answer:

Because even though the output from tail is redirected to 'newfile', input is redirected to sort from 'file', which contains more than 10 lines of text.

2. [20 Points] Process management.

(a) In these exercises, we'll use a new command 'sleep'. What does 'sleep' do?

Answer:

Delay for a specified amount of time (in seconds).

(b) Type 'sleep 300' and hit the enter key. Now move this process to the background. Describe your solution.

Answer:

First, suspend the process using <ctrl>-z then move it to the background using the 'bg' command.

(c) Kill the process from the previous exercise.

Answer:

```
bash$ ps
PID TTY TIME CMD
10826 pts/0 00:00:00 bash
12616 pts/0 00:00:00 sleep
12617 pts/0 00:00:00 ps
bash$ kill 12616
```

(d) Again, sleep for 300 but this time alter the command so that the process immediately goes to the background.

Answer:

```
bash$ sleep 300 &
[1] 12627
```

(e) Print out the process information for the command in the previous exercise. Your command should not print out process information for any other command.

Answer:

```
bash$ ps 12627
PID TTY STAT TIME COMMAND
12627 pts/0 S 0:00 sleep 300
```

(f) Print out the job information for all currently running jobs, including the process from exercise (d). Kill the process from exercise (d).

Answer:

```
bash$ jobs
bash$ kill 12672
```

(g) Start up three different sleep processes in the background. Then print the job information for all three of them. Next, move the second sleep process to the foreground. Kill the process without suspending it. Kill the other two processes using the 'kill' command. Describe your solution.

Answer:

```
bash$ sleep 300 &
[1] 12638
bash$ sleep 300 &
[2] 12639
bash$ sleep 300 &
[3] 12640
bash$ jobs
[1] Running sleep 300 &
[2]- Running sleep 300 &
[3]+ Running sleep 300 &
bash$ fg %2
sleep 300
```

```
bash$ kill 12638
bash$ kill 12640
```

The solution is illustrated above with one exception. After typing 'fg %2', I used '<ctrl>-c' to kill the process.

- (h) Again, start up three sleep processes, but this time suspend all three of them. Move the first job to the background. Kill all three processes using the 'kill' command. Describe your solution.

Answer:

```
bash$ sleep 300
```

```
[1]+ Stopped sleep 300
```

```
bash$ sleep 300
```

```
[2]+ Stopped sleep 300
```

```
bash$ sleep 300
```

```
[3]+ Stopped sleep 300
```

```
bash$ bg %1
```

```
[1] sleep 300 &
```

```
bash$ ps
```

```
PID TTY TIME CMD
```

```
10826 pts/0 00:00:00 bash
```

```
12697 pts/0 00:00:00 sleep
```

```
12698 pts/0 00:00:00 sleep
```

```
12699 pts/0 00:00:00 sleep
```

```
12708 pts/0 00:00:00 ps
```

```
bash$ kill -9 12697 12698 12699
```

```
[1] Killed sleep 300
```

```
[2]- Killed sleep 300
```

```
[3]+ Killed sleep 300
```

The solution is illustrated above. Notice that I need to use the 'kill -9' for all suspended processes.

- (i) Start up a sleep process. Kill the process, but using the job number rather than the PID.

Answer:

```
bash$ sleep 300 &
```

```
bash$ kill %1
```

- (j) Start up a sleep process. Kill the process using the highest kill level.

Answer:

```
bash$ sleep 300&
```

```
bash$ kill -9 12709
```

3. [10 Points] New command: *top*.

- (a) Describe what the command 'top' does.

Answer:

'top' interactively allows users to manage processes. It allows users to view all processes (like 'ps') as well as to kill any of their own processes.

(b) What key do you use to view processes sorted by CPU utilization?

Answer:

Entering the 'c' key will sort by CPU utilization.

(c) What key do you use to view processes sorted by memory usage?

Answer:

Entering the 'm' key will sort by memory usage.

(d) How do you view only one user's processes in top?

Answer:

You can view one user's processes by entering the 'u' key, typing the username in, and hitting enter.

(e) How do you kill a job in top?

Answer:

To kill a process you type 'k', typing in the PID of the process you wish to kill. After entering the PID you enter in a kill level (e.g., 9).

4. [6 Points] Using google. Google (like man) is an important resource for finding information about Unix and Linux commands. Use google to solve the following exercises. For each problem, answer the question and also list the search key words you used as well as the web page you found that provided the question.

(a) How can redirect standard error to standard output?

Answer:

You can redirect standard error to standard output using '2>&1'. For example:

```
bash$ ls *.txt *.jpg 2>&1
```

Google keywords: bash redirecting standard error to standard output

Website: http://theory.uwinnipeg.ca/localfiles/infofiles/bash/bashref_37.html

(b) After redirecting standard error to standard output, how do you redirect the output to a file?

Answer:

To redirect to a file, you can use '>' with '2>&1'. Note: the '2>&1' must follow the redirection to the file. For example:

```
bash$ ls *.txt *.jpg > foo 2>&1
```

This convention is somewhat unfortunate since it doesn't match our intuition: first we combine the streams ('2>&1') then we send the one stream to a file (> foo).

A better strategy is to use parentheses. Using parentheses you can put the redirection to the file last:

```
bash$ (ls *.txt *.jpg 2>&1) > foo
```

Google keywords: bash redirecting standard error to standard output

Website: http://theory.uwinnipeg.ca/localfiles/infofiles/bash/bashref_37.html

(c) How do you pipe standard error to a unix command?

Answer:

To pipe standard error you need to first get rid of standard out, then send standard error to standard out,

and then pipe standard out to the next command. To get rid of standard out, we can redirect it to a (scratch) file. To send standard error to standard output we use '2>&1'. Then we can simply pipe the output to 'head'. Again, we use parentheses. For example:

```
bash$ (ls *.txt *.jpg > scratchfile) 2>&1 | head -n 3
```

In the next lab, we'll discuss '/dev/null', which could be used in place of 'scratchfile' to avoid creating a new file.

Google keywords: bash redirecting standard error to standard output

Website: http://theory.uwinnipeg.ca/localfiles/infofiles/bash/bashref_37.html

5. About this assignment.

- (a) Approximately, how long did it take you to complete this homework?
- (b) Would you classify this assignment as easy, straight-forward, or difficult?