

Please complete the following questions. If you need help, feel free to ask questions both from other students and from me. If you cannot finish during the class time then you should finish this lab at home. After you finish, feel free to clean up (*i.e.*, remove) any files or directories that were created, you won't need them in the future. But be careful not to remove any other important files you may have in your home directory.

1. Piping and redirection.

- (a) Do the following commands on one line (*i.e.*, only hit the enter key once): 1) change your location to the 'cse399' directory you created in previous homeworks or labs (create it again if you deleted it), 2) copy the directory /home1/m/mcorliss/teaching/cse399/lab2 and all its subcontents to your current location, 3) change your location to this new directory (lab2).

Answer:

```
bash$ mkdir cse399; cd cse399; cp -r /home1/m/mcorliss/teaching/cse399/lab2 .; cd lab2
```

- (b) Below is one approach to finding the size in bytes of the last 10 lines of file 'file'.

```
prompt$ tail -n 10 file > tmp
```

```
prompt$ wc -c tmp
```

Write one command to do this that doesn't require using a temporary file (hint: using piping).

Answer:

```
bash$ tail -n 10 file | wc -c
```

- (c) Print lines 45-50 of file 'file' in sorted order.

Answer:

```
bash$ head -n 50 file | tail -n5 | sort
```

- (d) Type 'cat | head -n3' and hit enter. Type '1' and enter, '2' and enter, and so on. What happens after entering each line of text? Why does the command quit prematurely?

Answer:

After entering each line of text, the shell prints out the same line. After typing the fourth line of text, the command quits prematurely since the head process has finished.

- (e) Type 'cat | sort' and hit enter. Begin entering lines of text as in the previous exercise. What happens in this case? Notice that the command does not return to the prompt. Use '<ctrl>-c' to quit the command. What happens? Unfortunately, '<ctrl>-c' kills the command before any sorting occurs. Therefore, no text is printed to the screen. Repeat the command, but this time after you're finished entering text, hit '<ctrl>-d' (which terminates the cat command rather than killing it). What happens in this case?

- (f) Type 'cat < infile | sort'. What happens? Now type 'cat | sort < infile'. What happens? What causes the two commands to act differently?

Answer:

'cat < infile | sort' sorts and prints the contents of file 'infile'. 'cat | sort < infile' sorts and prints infile, but also does not return from the prompt until one line of text is entered. The difference in behavior is due to the location of the redirection. In the first case, it applies to 'cat'. In the second case, it applies to the 'sort'. In the second case, 'cat' has no input so it waits for input from the keyboard. After entering a line of text, the 'cat' process wakes up and discovers the 'sort' process has terminated and so terminates itself.

(g) Type `'sort < infile1 < infile2'`. What happens?

Answer:

Only one file can be redirected to sort, and in bash it selects the last redirection (`'infile2'`). The command sorts the contents of `'infile2'`.

(h) Type `'cat file > outfile1 > outfile2'`. What happens? Now set the contents of `'outfile1'` to "abc" using the echo command. Redo the cat command. What happened to `'outfile1'`?

Answer:

Only one file can be written and bash chooses the last redirected file. However, each file's contents are emptied even though the output is redirected to only one file.

(i) List all files matching `'*.jpg'` and `'*.txt'`. Redirect standard error to `err.txt`. The old contents of `'err.txt'` should not be overwritten.

Answer:

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

2. 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) Print all processes belonging to `'root'` only, showing all fields, and displaying as a process tree.

Answer:

```
bash$ ps -fH -U root
```

(c) Type `'cat'` and hit enter. Now move this process to the background.

Answer:

First, suspend the process using `'<ctrl>-z'`. Then, type `'bg'`.

(d) Print out all jobs.

Answer:

```
bash$ jobs
```

(e) Suspend the `'cat'` process (hint: requires multiple steps).

Answer:

First, use `'fg'` to move the process to the foreground. Then suspend it using `'<ctrl>-z'`.

(f) Enter a command to sleep for 300 that automatically goes to the background.

Answer:

```
bash$ sleep 300 &
```

(g) Type `'sleep 300 &'` and hit enter. Then type `'sleep 400 &'` and hit enter. Now move the `'sleep 300'` job to the foreground. Then kill this job **without** suspending it first. Finally, kill the `'sleep 400'` job using the `'kill'` command.

Answer:

To move the 'sleep 300' job to the foreground, we can use 'fg %1'. To kill it, we can use '<ctrl>-c'. To kill the 'sleep 400' job, we can use the 'kill' command.

- (h) Enter three sleep commands, all on the same line: 'sleep 5', 'sleep 5', and 'sleep 5'. The first sleep command should execute and finish before the second sleep command begins. The second and third sleep commands should execute concurrently.

Answer:

```
bash$ sleep 5; sleep 5 & sleep 5
```

3. New commands.

- (a) **clear.** Type 'clear'. What does 'clear' do?

Answer:

'clear' clears the screen.

- (b) **date.** Type 'date'. What does 'date' do?

Answer:

'date' returns the current date.

- (c) **time.** Type 'time ls'. What does 'time' do?

Answer:

'time' times the command passed as an argument.

- (d) **lpr.** Although, we won't test it out, 'lpr' can be used to print documents. For instance, 'lpr -P ex_printer hw1.pdf' would print 'hw1.pdf' to the printer whose name is 'ex_printer'.

- (e) **scp.** We have seen already how to remotely log in to another machine using 'ssh'. 'scp' allows you to transfer files to and from a remote machine. Open up a new terminal (leaving the other one open). Now use 'scp' in this terminal to get the file 'file' in the lab1 directory (hint: look at 'man' for help). Then create a new file 'foo' on the local machine and transfer this to eniac-1. 'scp' can also act like 'cp'. If you don't give it a remote hostname, it will copy files locally. After you're finished close this terminal and go back to the previous one.

Answer:

```
bash$ scp eniac-1.seas.upenn.edu:cse399/lab1/file .
bash$ echo "abc" > foo
bash$ scp foo eniac-1.seas.upenn.edu:.
```

4. Bash editing.

- (a) Begin typing an arbitrary command. Before hitting enter, type '<ctrl>-a'. What does this do? Now type '<ctrl>-e'. What does this do?

Answer:

'<ctrl>-a' brings the cursor to the beginning of the line. '<ctrl>-e' brings the cursor to the end of the line.

- (b) Begin typing an arbitrary command. Make sure the command is a legal one. Before hitting enter, type left arrow repeatedly. What does this do? Now type right arrow repeatedly. What does this do? Again, hit the left arrow repeatedly and then hit the enter key. What happens?

Answer:

Left arrow moves the cursor one spot to the left in the command text. Right arrow moves the cursor one

spot to the right in the command text. Hitting enter will execute the command, no matter where the cursor is within the command text.

- (c) Begin typing an arbitrary command. Make sure your command is using at least two separated arguments (e.g., “-a -b -c” rather than “-abc”). Before hitting enter type ‘<alt>-b’. What does this do? Now type ‘<alt>-f’. What does this do?

Answer:

‘<alt>-b’ moves the cursor back one word and ‘<alt>-f’ moves the cursor forward one word.

- (d) Begin typing an arbitrary command. Before hitting enter, type ‘<ctrl>-a’ and then ‘<ctrl>-k’. What happens? Now type a new command using a number of separated arguments. Then type ‘<alt>-<backspace>’ a few times. What does this do?

Answer:

‘<ctrl>-k’ cuts the text to the right of the current position of the cursor.

- (e) Type ‘<ctrl>-y’. What happens? Now move the cursor to the middle of the command text. Hit ‘<ctrl>-k’. Then hit ‘<ctrl>-y’ followed by ‘<alt>-y’. Continue hitting ‘<alt>-y’. What do ‘<ctrl>-k’, ‘<ctrl>-y’, ‘<alt>-y’ all do?

Answer:

‘<ctrl>-y’ pastes the most recently cut text. Hitting ‘<alt>-y’ after ‘<ctrl>-y’ allows you to scroll through the history of cut text.

5. Bash history.

- (a) At the prompt hit the up arrow repeatedly. What does this do? Now hit the down arrow. What does this do?

Answer:

Up arrow goes back through previous commands, while the down arrow goes forward.

- (b) Use ‘man’ to find what the command ‘history’ does. Notice that ‘man’ brings you to a generic bash manual. Use the search feature in man (which is the same as in less) to find the documentation on the history command.

Answer:

‘history’ displays the command history with reference numbers.

- (c) To re-execute a command, you can use ‘!<num>’ where <num> is the reference number displayed by ‘history’. Use this to re-execute the command from exercise 1(b).

Answer:

```
bash$ !1001
```

- (d) At the prompt, type ‘<ctrl>-r’. Begin typing “ps -f”. What happens? Try typing ‘<ctrl>-r’ and typing other text. What does ‘<ctrl>-r’ do? Type ‘<ctrl>-r’, type ‘cd’ and then repeatedly hit ‘<ctrl>-r’. What happens?

Answer:

‘<ctrl>-r’ allows users to search backwards through the history. By typing it once, you can enter in a search string. It will automatically show you the first matched command. If you type it again, after entering a search string, it will search backward for the next command that matches the string.