Assignment 3

Chapter 4: Q8, Q9, Q10

8. Suppose a user belongs to a group that has all permissions on a file named **jobs_list**, but the user, as the owner of the file, has no permissions. Describe which operations, if any, the user/owner can perform on **jobs_list**. Which command can the user/owner give that will grant the user/owner all permissions on the file?
Initially the user/owner cannot perform any operations involving the file, other than using ls to list it. When the user/owner gives the following command, the user/owner can perform any operation involving the file:

```
$ chmod u+rwx jobs_list
```

9. Does the root directory have any subdirectories you cannot search as an ordinary user? Does the root directory have any subdirectories you cannot read as a regular user? Explain.
Generally you cannot search or read the **lost+found** directory in any filesystem, including root (**/lost+found**), because its permissions are **rwx------**(700). Other answers are system dependent. Refer to "Directory Access Permissions" on page 109 for more information.

10. Assume you are given the directory structure shown in Figure 4-2 on page 83 and the following directory permissions:

```
d--x--x--- 3 zach pubs 512 2013-03-10 15:16 business
drwxr-xr-x 2 zach pubs 512 2013-03-10 15:16 business/milk_co
```

For each category of permissions—owner, group, and other—what happens when you run each of the following commands? Assume the working directory is the parent of **correspond** and that the file **cheese_co** is readable by everyone.
a. **cd correspond/business/milk_co**
owner: OK; group: OK; other: **Permission denied**
b. **ls –l correspond/business**
owner, group, and other: **Permission denied**
c. **cat correspond/business/cheese_co**
owner and group: **OK**; other: **Permission denied**

Chapter 5: Q1, Q2, Q3, Q4, Q5, Q9, Q16

1. What does the shell ordinarily do while a command is executing? What should you do if you do not want to wait for a command to finish before running another command?
The shell sleeps while a command is executing in the foreground. When you want to keep working while a command is running, run a command in the background by ending the command line with an ampersand (**&**).

2. Using sort as a filter, rewrite the following sequence of commands:

```
$ sort list > temp
$ lpr temp
$ rm temp
```

```
$ cat list | sort | lpr
```

3. What is a PID number? Why are these numbers useful when you run processes in the background? Which utility displays the PID numbers of the commands you are running?
PID stands for *process identification.* A PID number uniquely identifies the process running a command. When you run a command in the background, you can use its PID number as an argument to kill to stop the command from running. The ps utility displays PID numbers.

4. Assume the following files are in the working directory:

```
$ ls
intro notesb ref2 section1 section3 section4b
notesa ref1 ref3 section2 section4a sentrev
```

Give commands for each of the following, using wildcards to express filenames with as few characters as possible.

a. List all files that begin with **section**.

```
$ ls section*
```

or

```
$ ls sec*
```

b. List the **section1**, **section2**, and **section3** files only.

```
$ ls section[1-3]
```

or

```
$ ls section[123]
```

c. List the **intro** file only.

```
$ ls i*
```

d. List the **section1**, **section3**, **ref1**, and **ref3** files.

```
$ ls *[13]
```

5. Refer to Part VII or the info or man pages to determine which command will

a. Display the number of lines in its standard input that contain the *word* **a** or **A**.

```
$ cat file | grep -wci a
```

or

```
$ cat file | grep –wc [aA]
```

or

```
$ cat file | grep –c [aA]
```

b. Display only the names of the files in the working directory that contain the pattern **$(.**

```
$ ls *$\(*
```

or

```
$ ls | grep '$('
```

c. List the files in the working directory in reverse alphabetical order.

```
$ ls -r
```

d. Send a list of files in the working directory to the printer, sorted by size.

```
$ ls -S | lpr
```

9. Explain the following error message. Which filenames would a subsequent ls command display?

```
$ ls
abc abd abe abf abg abh
$ rm abc ab*
rm: cannot remove 'abc': No such file or directory
```

The shell expands the asterisk wildcard character before it passes a list of filenames to rm. As a result rm receives a list of files that includes **abc** twice. After rm removes **abc**, it generates an error message when it is asked to remove **abc** again. After giving the preceding rm command, ls does not list any files.

16. Create a file named **answer** and give the following command:

```
$ > answers.0102 < answer cat
```

Explain what the command does and why. What is a more conventional way of expressing this command?

Reading the command line from left to right, it instructs the shell to redirect standard output to **answers.0102**, redirect standard input to come from **answer**, and execute the cat utility. More conventionally, the same command is expressed as

```
$ cat answer > answers.0102
```

or simply

```
$ cp answer answers.0102
```