Cabrillo College

Lab 7: Input and Output

The goal of this lab is to gain proficiency in using I/O redirection to perform tasks on the system. You will combine commands you have learned in this course using shell redirection, pipes and tees to perform a variety of tasks on the system.

Preparation

Everything you need to do this lab can be found in the Lesson 8 materials on the CIS 90 website: http://simms-teach.com/cis90calendar.php. Review carefully all Lesson 8 slides, even those that may not have been covered in class.

Check the forum at: http://oslab.cishawks.net/forum/ for any tips and updates related to this lab. The forum is also a good place to ask questions if you get stuck or help others.

If you would like some additional assistance come to the CIS Lab on campus where you can get help from instructors and student lab assistants: http://webhawks.org/~cislab/.

Procedure

Log on to Opus so that you have a command line shell at your service. Be sure you are in your home directory to start this lab. We are going to experiment with how commands get their input and what they do with their output. Then we will perform a series of tasks by combining commands together and saving the output to a file.

The find command

The syntax of the find command is:

```
find <search-directory> -name <filename> -user <username>
```

When the **-name** option and its argument are omitted; all files are displayed.

1. Find all the files under your home directory by issuing the command:

   ```
   find $HOME
   ```

2. Find all the files named *old* that are somewhere in or below your parent directory using the command:

```
find .. -name old
```

   Were there any error messages?

3. Filter out the error messages by redirecting *stderr* to a file called *errors* in your home directory:

```
find .. -name old 2> errors
```

4. Another useful option to the find command is **-user** which takes an argument of a user's name or id #. With this command you can find all the files that you own on the entire system and save them in a text file. Since we may get some error messages for directories we don't have permission for, let's also redirect the errors to the "bit bucket". This command may take a minute or so.

```
find / -user $LOGNAME > myfiles 2> /dev/null
```

The grep command

The syntax of the **grep** command is:

```
grep "search-string" <filenames...>
```

1. Find out how many of the sonnets contain the string "love" by changing your directory to *Shakespeare* and entering the command:

```
grep "love" sonnet*
```

   Does **grep** find just the words "love" or the string of letters: l,o,v,e?

2. One of the nice things about **grep** is that it will read its input from *stdin* if it is not specified on the command line. Change back to your home directory and try this command:

```
who | grep $LOGNAME
```

   What command does this remind you of?

3. Run the above command again, but this time save the output to a file called *whoami* in your home directory.

4. Can you combine the **file** command with **grep** to list all text files in your home directory?

```
file * | grep text
```

The wc command

This command will count characters, words and lines in a text file.
Often we are just interested in the number of lines in a file, so we use the -l option.

1. Let **wc** count the number of lines in Shakespeare's sonnets:

   ```
   wc  -l  poems/Sha*/son*
   ```

   Notice they all have the same number of lines?

2. Use word count to count all the files that you own on the system:

   ```
   wc -l myfiles
   ```

3. Count the number of files there are underneath your parent directory, */home/cis90* :

   ```
   find  /home/cis90 | wc -l
   ```

The spell command

Can be used to check the spelling in text files.

1. Let's find out how many misspelled words are in the file *small_town*.
   Where is *small_town*? Change to that directory and type:

   ```
   spell small_town
   ```

   Notice that some words may be spelled correctly but aren't in UNIX's dictionary.

2. Change to the *Shakespeare* directory and find how many misspellings there are in all the sonnets.

   ```
   spell sonnet* | wc -l
   ```

   What if you wanted to see these misspelled words?

The sort command

1. Change to your *misc* directory and display the file *fruit.*

   ```
   cat fruit
   ```

2. Sort the contents of this file using the command:

   ```
   sort fruit
   ```

   Note: the contents of the *fruit* are unchanged; only the output is sorted.

3. Sort the *fruit* file in reverse order and save the results to *tiurf*

   ```
   sort -r fruit > tiurf
   ```

The tee command

1. At times, you may want to see the results of a command on your screen as well as saving those results to a file. This may be accomplished using the **tee** command which takes one source of input (stdin) and writes that input to two outputs: stdout and to a file named as a command line argument. Change to the *Shakespeare* directory and run the command:

```
spell sonnet1 | tee words
```

Notice how the misspelled words came to the screen and also went to the file *words*.

2. Now let's use the tee command to get a sorted list of the misspelled words in all of Shakespeare's sonnets and count how many there are all at the same time. Change to your home directory and use the **tee** command to collect the intermediary results:

```
spell poems/Shakespeare/son* | sort | tee words | wc -l
```

Display the file *words* to see all the misspelled words.

Putting Commands Together

For your lab07, we are going to analyze your past 125 commands.

1. Create the file, *lab07*, by redirecting the output of the date command:

```
date > lab07
```

2. Create a file that lists your past 125 commands:

```
history 125 > cmds
```

3. How many times have you used the **cd** command? Send the results to the file *lab07*: (Note: the following two lines represent two distinct commands.)

```
echo -n "#Times I have used the cd command: " >> lab07
grep "cd" cmds | wc -l >> lab07
```

Verify your results by displaying the file *lab07* to the screen.

4. Repeat step three but count the number of times you have used the **clear** command.

5. Repeat step three but count the number of times you have used the **grep** command.

6. Add the sorted list of misspelled words from Shakespeare's sonnets to your *lab07* file:

```
cat words >> lab07
```

7. Now tack on a list of all the files you own on opus in alphabetic order.
First update your list of files with:

```
find / -user $LOGNAME > myfiles 2> /dev/null
```

Sort the updated file, *myfiles* in dictionary order and append it to your lab file:

```
sort -d myfiles >> lab07
```

8. Add the commands you used in this lab to your *lab07* file:

```
cat cmds >> lab07
```

9. Review your *lab07* file:

   ```
   less lab07
   ```

   Do you see the date, the 3 command counts, the misspelled words, the files you own, and your last commands issued? If not you should repeat the steps above.

10. You are almost done with this lab. Congratulate yourself by mailing the banner message, GOOD WORK to your mailbox:

    ```
    banner Good Work | mail -s "Pat on the Back" $LOGNAME
    ```

    Notice how the -s option to the **mail** command allows you to specify a subject for your message.

Submittal

You have now finished this lab. All you need left to do is copy it to me. The command to do that is given below:

```
cp lab07 /home/rsimms/turnin/cis90/lab07.$LOGNAME
```

Be sure to submit before the deadline. Remember, late work is not accepted.

Grading Rubric (30 points total)

30 points for successfully completing all steps
Less 1 point for each step not completed correctly