Computer Science 433
Programming Languages
The College of Saint Rose
Fall 2014

Program/Problem Set 1: Unix Commands

Due: 11:59 PM, Wednesday, September 3, 2014

In this assignment, you will learn (or refresh your memory about) some Unix commands, then implement a few of them in the language of your choice (I assume this will be Java for most of you). You may work alone or with a partner on this assignment.

The Unix Command Line

To get started, log into `mogul.strose.edu`. In your favorite editor, create a document in which you will answer the questions you find in this section. Start by putting your name at the top of this document.

GUIs are nice, but they can be slow to navigate and too restrictive for some purposes. For some of our examples and assignments, you will be working in a Unix environment and interacting with the system by typing commands at the Unix *shell*, or *command line*. When you log in, you will be presented with a prompt. This is your direct interface to issue commands to the operating system. When you type a command here, the shell will execute the command on your behalf, print out any results, then reissue the prompt.

Of course, the command line is useless if you don't know what commands it understands. You will learn about several important commands in this lab and many more throughout the semester. One of the most important is `man` – the Unix manual. Every Unix command has a manual page, including `man`. To see the manual page about `man`, type the command:

```
man man
```

Navigating the Directory Structure

You have very likely used systems where files can be organized into *folders*. When accessing these kinds of structures from the command line, we usually refer to them as *directories*. Each program in a Unix system, including your shell, maintains the notion of a *working directory*. That is where the program will look for files unless instructed to do otherwise. You'll hear Unix users asking a question like "What directory are you in?" and the answer to this is your working directory.

When you first open a shell, your *home directory* is your working directory. The command `pwd` will instruct the shell to print your working directory.

? Lab Question 1:

What is your home directory on `mogul.strose.edu`? (use `pwd`) ($\frac{1}{2}$ point)

You can also list the contents of your working directory with the command `ls`.

Lab Question 2:
What output do you see when you issue the `ls` command on `mogul.strose.edu`? ($\frac{1}{2}$ point)

Other important operations to navigate and modify the directory structure are changing your working directory (`cd`), creating a new directory (`mkdir`), and removing a directory (`rmdir`).

Create a directory in your account for your work for this course (`cs433` might be a good name), and a directory within that directory for this assignment (`ps1` might be a good name).

Lab Question 3:
Change your working directory to the one you just created and issue the `pwd` command. What does this show as your working directory? ($\frac{1}{2}$ point)

In your shell window and in your home directory (note: you can always reset your working directory to be your home directory by issuing the command `cd` with no parameters), issue this command:

```
uname -a > linux.txt
```

This will execute the command `uname -a`, which prints a variety of information about the system you are on, and "redirects" the output, which would normally be printed in your terminal window, to the file `linux.txt`.

Unix Commands

Identify the function of and experiment with these Unix commands (a few of which you have already used):

```
ls     cd       cp       mv      rm      mkdir    pwd
man    chmod    cat      more    grep    head     tail
ln     find     rmdir    wc      diff    tar      touch
```

Lab Question 4:
Give a one sentence description of each command. (3 points total)

Using appropriate commands from the above list, move the `linux.txt` file you created in your home directory into the directory you created for your work for this assignment.

Show that this has worked by issuing the following command from inside of your course directory:

```
ls -laR > ls.out
```

Then move the file `ls.out` into your directory for this assignment.

? Lab Question 5:
Class examples are in `/home/cs433/examples` on mogul. Give the command that would copy the entire directory of the "`late`" example into your directory for this assignment. ($\frac{1}{2}$ point)

All of these programs (except `late.bas`) can be run on mogul. Run each program at the command line and redirect its output to an appropriately-named file (*e.g.*, "`late.c.out`" might be a good name for the C program's output). Hints: the C program can be compiled with `gcc` and the executable run directly, the Java program can be compiled with `javac` and run with `java`, the Perl and Python programs can be sent directly to their interpreters, which are installed as `perl` and `python3`, and the scheme program can be loaded into the MIT Scheme interpreter (`scheme`) by giving the command-line parameters "`-load late.scm`".

Create a "tar file" that includes your `linux.txt`, `ls.out`, and all of the "late" output files by issuing this command while in your directory that contains those files:

```
tar cvf unix.tar linux.txt ls.out ...
```

(Of course, you will replace the "..." with the list of files you created when you redirected the output of each of the "late" programs.)

This will create a file `unix.tar` in your directory. Transfer this to the computer you're working on with a secure copy program like WinSCP or FileZilla.

Implementing Some Unix Commands

Your programming tasks are to implement clones of 4 Unix commands in the language of your choice:

File Copy

`cp filenameA filenameB` - copy the contents of `filenameA` to a file named `filenameB`; see the Unix `cp` command.

Note: if you write this in Java, you would instead issue the command

```
java cp filenameA filenameB
```

but it should behave otherwise like the Unix `cp` command.

Display File Contents

`cat [OPTION] filename` - display the contents of `filename`, where the options can be either (or neither) of:

- `-n` - number all output lines
- `-b` - number nonempty output lines

See the Unix `cat` command.

Display Initial Lines of File

`head [-`n`] filename` - display the first n lines of `filename`, default is to display 10 lines if the flag is not specified. If the file has fewer than the requested number of lines, the entire file should be displayed.

See the Unix `head` command.

Word Count

`wc filename` - find the number of lines, words, and bytes (characters) in `filename`.

See the Unix command `wc`.

Write a separate program for each of the above (that is, do not have a single program that tries to perform all of the functionality).

Your code should be commented appropriately throughout. Please also include a longer comment at the top of your program describing your implementation. And, of course, it should include your name.

Submission

Before 11:59 PM, Wednesday, September 3, 2014, submit your work for grading. Create and submit a single archive file (a `.7z` or `.zip` file containing all required files) using Submission Box at `http://sb.teresco.org` under assignment "PS1".

Grading

This assignment will be graded out of 40 points.

Feature	Value	Score
`linux.txt` file	1	
`ls.out` with correct directory structure	1	
Unix command descriptions	3	
Other question responses	2	
"late" program output files in `unix.tar`	3	
`cp` program correctness	6	
`cat` program correctness	6	
`head` program correctness	6	
`wc` program correctness	6	
Program documentation	3	
Program efficiency, style, and elegance	3	
Total	40	