

CIS121 - Spring 2012

Homework 3 Friday, February 3rd

Due: Friday, February 10 at 5:00pm

Goals:

- Improve and practice basic Unix command line skills
- Improve understanding of Unix `WC` Utility
- Improve understanding of how to compile and run programs from the command line
- Improve understanding of asymptotic runtime analysis

Overview:

The following problems require you to demonstrate your ability to use the Unix command line, ability to implement a real Unix command in Java, familiarity with the Java classpath and compiling from the command line, and knowledge of asymptotic notation and Big-O notation. For the Unix command line section, you will have to demonstrate your ability to perform certain functions through the command line. For the Unix `WC` utility section, you will be asked to implement the Unix `WC` utility. For the Asymptotic Notation section, you will be asked questions designed to test your familiarity with asymptotic growth rates as well as your ability to use the Big-O definition and theorems in formal proofs.

Problems:

Unix Command Line: 35 points

Note: This portion of the homework is expected to be completed after Unix is covered in lab on Monday/Tuesday, so we recommend you start with the asymptotic analysis problems first. If you'd like to prepare for the lab and this homework, you can browse the Unix tutorials here: <http://www.ee.surrey.ac.uk/Teaching/Unix/>

Below is a list of operations that should be completed from the Unix command line. While most of the operations listed below can be executed in 1 Unix command, some may take more. For some operations, we will specify a number of commands necessary and ask that you perform these operations with exactly this many commands. Also note that many operations will require you to execute commands from remote directories.

As you work through the assignment you are likely to make some mistakes, and we understand that.

The minimum set of instructions that are necessary to complete this assignment is around 35 or 40. The last command in the series outputs a file that will contain the last 120 commands that you typed in the terminal. This will be included with your submission. We give you this freedom because we expect you to check your work along the way, by changing or listing directories, printing files to the screen, etc. However, in your set of 120 instructions, you must have the minimum set of 35-40 that we will be grading against.

For technical reasons, we recommend you do this portion of the assignment on ENIAC machines. You may also use your own computer if you have Mac or Linux installed as your operating system. Unfortunately, Windows users cannot do this assignment from their native Command Prompt as it does not support the same set of commands that Unix provides.

On Eniac / Mac:

1. Log in with your username and password
2. Open the terminal (Search for Terminal if you cannot find it)
3. You are ready to go

If for some reason, you are a Windows user and cannot go to the Eniac machines, you should read the following articles. Again, we do not recommend this option.

1. How to be able to access Eniac remotely:
<http://www.seas.upenn.edu/cis1xx/resources/remoteLogin.shtml>
2. How to transfer a file from Eniac to my computer:
<http://www.seas.upenn.edu/cis1xx/resources/transferringFiles.shtml>
3. More information on the CIS 121 resources page:
<http://www.seas.upenn.edu/cis1xx/resources.shtml>

Assignment

Note: [username] denotes your PennKey and should be replaced by your PennKey. For example, cis121hw3_[username] for user stz would be cis121hw3_stz

1. Make a folder called cis121hw3_[username] on Eniac and navigate to this directory.
2. Create an empty file unix-[username].txt in that directory
3. Insert your name, email address, and major in the file (use `cat` and a file redirection. Do not attempt to open the file locally and insert information manually)
4. Insert 2-3 sentences about yourself [your hobbies/interests/why youre taking this course] (again using the `cat` command). Hint: Make sure that you **append** to the file

5. Create the directories `history`, `test`, and `foo`
6. Rename `foo` to `os`
7. From the present working directory (`cis121hw3-[username]`) create a new file `os.txt` in the `os` directory (**1 cmd**)
8. Save the output of the command `uname -a` into `os.txt` (**1 cmd**)
9. From the current directory (`cis121hw3-[username]`) copy `os.txt` to the `test` directory with name `os-copy` (**1 cmd**)
10. In the current directory (`cis121hw3-[username]`) create a new directory `copies`
11. Move the file `os-copy` into `copies`
12. Remove the directory `copies` (Hint: The directory is not empty!)
13. Go to `test` and redirect the output of the command that prints the present working directory into a file called `[name of command].txt`
14. Go to your home directory
15. From the home directory, create a file `numwords` in your `cis121hw3-[username]` directory that contains the number of words in `unix-[username].txt` (**1 cmd**)
16. From the home directory, create a file `books.txt` in your `cis121hw3-[username]` directory and enter a list of your 3 favorite books. (**1 cmd**)
17. From the home directory, create a file `movies.txt` in your `cis121hw3-[username]` directory and enter a list of your 5 favorite movies. (**1 cmd**)
18. Sort the movies and save the sorted movies into `sorted-movies.txt` in your homework directory (**1 cmd**)
19. List all files from your `cis121hw3-[username]` directory that end with `.txt` (excluding textfiles from subdirectories)
20. From your home directory, type:
`history | tail -n 120 > cis121hw3-[username]/history/cmds` (**1 cmd**)
Linux saves all commands you type in a history file. This command copies the last 120 lines from your history and saves them in a file called `cmds`. This is the file that the TAs are going to look into.

This is it! To test your work, type this command in the terminal and make sure you are in your home directory:

```
grep 'mkdir[[:space:]]\+cis121hw3' cis121hw3-[username]/history/cmds
```

This command checks if the first command you were required to enter for this homework exists in the history of the last 120 commands.

If this command produces output, you have successfully attempted to complete the UNIX part of the homework. If you don't see anything, make sure you type the command again very carefully! If you see no output, you have probably used more than 120 commands and should try to do it again. Again, when you're not sure, ask the friendly TAs.

In the end, archive your directory by typing:

```
tar cvfz cis121hw3_[username].tar.gz cis121hw3_[username]
```

You should see the output of the operation. Then submit the archive on the cis121 home page.

Unix WordCount Utility: 30pts

Unix provides a standard word count utility `wc`. This command is used to count the lines, words and characters in a text file. As part of this homework, you will implement a `wc`-like tool in a class named `WordCount`.

Similarly to the UNIX word count utility, your program should take a list of filenames as a command-line argument. If two or more files are supplied – their names are separated by spaces. For each file, your program should output the number of lines, words, and characters in that file. If no filename is provided, your program should report that a filename argument is required. You are not responsible for reading input from the commandline, as the real Unix `wc` does. You may assume that your utility will only be invoked on ASCII files, hence, you are not required to handle binary files. (ASCII files are typical human-readable files like `.txt` and `.java` files. Binary files are encoded in binary.)

You do not need to implement any of the invocation options that the `wc` utility on Unix provides. The only way to invoke your utility should be with one or more filenames as an argument.

Format your output the same way as is done by the Unix `wc` utility. So, if `WordCount` is invoked with 1 filename, you should print the following output:

```
> WordCount And_Thou_Art_Dead.txt
85  475 3409 And_Thou_Art_Dead.txt
```

If multiple filenames are given, you should print statistics for each file followed by a summary line, like so:

```
> WordCount And_Thou_Art_Dead.txt Darkness.txt
85  475 3409 And_Thou_Art_Dead.txt
88  639 4662 Darkness.txt
173 1114 8071 total
```

In order to test your code, use the `javac` and `java` commands (covered in Lab 3) from the Unix command line. Some sample text files have been provided to use as input to test.

Please Note: Do not name your class `WC`, as when you test from the command line it will simply run the built in `wc` instead of your own.

Asymptotic Analysis: 30pts

1. Order the following functions by asymptotic growth rate indicating when two or more are big-Theta of each other: $n^2\sqrt{n}$, 2^{2n-3} , $\sqrt{n^3}$, $4n\log n + 2n$, 3^{n+3} , $1 + 2^2 + \dots + n^2$, $\log n^n$, $\frac{\log n^2}{\log n}$
2. Prove or Disprove: If $f(n)$ is $O(h(n))$, $g(n)$ is $O(h(n))$, and $z(n)$ is $O(g(n))$, then $f(n) + z(n)$ is $O(h(n))$. Show your work.
3. Prove or Disprove: If $f(n)$ is $O(g(n))$, $g(n)$ is $O(h(n))$, then $f(n)\log(g(n))$ is $O(h^2(n))$
4. In this problem you are NOT allowed to use the theorems about Big-Oh stated in the lecture notes. Your proof should follow just from the definition of Big-Oh. Prove that for any $f(n)$ (as usual mapping nonnegative reals to strictly positive reals) if $f(n)$ is $O(n^n)$ then $\log f(n)$ is $O(n \log n)$.
5. **(Extra Credit: 5pts)** Compute the time complexity of the following binary search algorithm that takes a sorted array and key as input and outputs the index of the given key. Give your answer in Big-O notation.

```
int binarySearch(int[] in, int key) {
    int bot = 0;
    int top = in.length - 1;
    int mid = top / 2;

    while (bot <= top) {
        if (key == in[mid]) {
            return mid;
        } else if (key < in[mid]) {
            top = mid-1;
            mid = bot + ((top - bot) / 2);
        } else if (key > in[mid]) {
            bot = mid+1;
            mid = bot + ((top - bot) / 2);
        }
    }
    return -1;
}
```

Explain your answer.

Grading

What to turn in

You should submit the following:

Note: Please staple your sheets together and label them with your name and PennKey

- `cis121hw3-[username].tar.gz` (through online submission page)
- `WordCount.java` (through online submission page)
- Physical paper sheet(s) with your answers to the Asymptotic Analysis questions, to be handed in to Maggie in the Levine 302 **4pm on Friday**.

Unix

To get full credit for the Unix section, operations specified as being executed in a single command (**1 cmd**) must be executed as such.

Style and Documentation

The above parts together are worth a total of 95 points. The remaining 5 points are awarded for code style and sensible documentation. Style includes proper indentation, clear variable names, consistent naming conventions, and proper use of public and private, while sensible documentation includes proper use of comments to describe what functions do and the code internals. Please note, sensible documentation is not the same as large amounts of comments. Most lines of code do not need to be explained, especially if your variables have clear names. Use your own judgement for when comments are necessary.