

## Assignment 0: emacs vs. vi, csh vs. ksh

**Complete assignment due: Tuesday January 29th, 1:59 p.m.**

### Problem 1 (20 points): Java and command-line arguments.

For this problem, you are to parse the UNIX command-line arguments in Java. This is a simple java program, but you will be interacting with it from the UNIX command-line to become familiar with you how UNIX deals with arguments to programs.

You are to write a Java text file called **args.java** on Lectura and run it from Lectura using the command-line tools we have been discussing in class (**javac** for the Java compiler and **java** to invoke the Java virtual machine). There is a script **:/home/cs352/spring08/assign0/test0.ksh** which will show you how we expect to compile and execute your program.

The Java program is pretty simple: It reads the command-line arguments, and depending on the type of argument, it will classify it either as (1) a plain argument to (2) a “short” option, specified with a single - (3) a “long” option, specified with two - (i.e., --) or (4) a named directory, where either a long or a short option ends with the 3 letters “dir”. In the case of a named directory, the next argument is the named directory. For example:

```
$ javac args.java # compile the program to an args.class
$ java args arg1 -shortoption arg2 --longoption -idir somedir
```

**Arguments:**

**arg1**

**arg2**

**Directories:**

**somedir**

**Short Options:**

**shortoption**

**Long Options:**

**longoption**

There are a few things to notice about the output:

The order of output is always Arguments, Directories, Short Options and Long Options. Within each heading, they should also be in the order they appeared on the command-line. In the example above, note that even though there was an intervening option, **arg1 arg2** were still listed in that particular order on the command-line on purpose.

For the short and long options, note that the dashes (-) were removed before output.

If a type of command-line argument does not appear, then it should not be listed in the output. For example, only the Arguments appear below:

```
$ java args arg1 arg2
```

**Arguments:**

**arg1**

**arg2**

If there are any further questions about what the output should look like, take a look at the

```
/home/cs352/spring08/assign0/test0.ksh
```

It will have all the test cases we expect, with the expected output in the

```
/home/cs352/spring08/assign0/test0.output
```

We will be using a program called **diff** to test your assignment (get comfortable with this program!) and we will discuss this in class. To try out your solution against our expected solution:

```
$ ksh /home/cs352/spring08/assign0/test0.ksh > myoutput.txt
```

```
$ diff myoutput.txt /home/cs352/spring08/assign0/test0.output
```

The diff will show you all the places you DO NOT match: if it matches exactly (which is what you want), there will be no output. (UNIX Philosophy: only report problems)

It is strongly recommended you do all the work on Lectura so you can get used to UNIX, especially since we will be grading it ON Lectura!! If you choose not to use **emacs/vi**, and you use a Windows computer, then there is an extra step needed. After you have edited the file on your Windows system, upload it to your CS account. Then, on lectura, use the **dos2unix** program on the file before you turn it in. There is a manpage for **dos2unix**. (If you use vi on lectura to complete this problem, you do not need to use the **dos2unix** program.)

**Turnin:** Use the **turnin** program to turn in your **args.java** file. The command is:

```
turnin 352assign0 args.java
```

See the man page for the **turnin** program for details on what turnin can do and how you can confirm that your file was turned in. The **turnin** program must be run ON THE COMMAND-LINE ON LECTURA (not from a windows icon!)

## **Problem 2 (20 points): emacs.**

Copy the file **/home/cs352/spring08/assign0/problem2.txt** to your directory. Answer the **emacs** questions that are contained in the file.

It is not required that you actually use **emacs** to answer the questions. It is recommended, so that you can gain some experience with **emacs**. If you choose not to use **emacs**, and you use a

Windows computer, then there is an extra step needed. After you have edited the file on your Windows system, upload it to your CS account. Then, on lectura, use the **dos2unix** program on the file before you turn it in. There is a manpage for **dos2unix**. (If you use vi on lectura to complete this problem, you do not need to use the **dos2unix** program.)

**Turnin:** Use the **turnin** program to turn in your **problem2.txt** file. The command is:  
**turnin 352assign0 problem2.txt**

See the man page for the **turnin** program for details on what turnin can do and how you can confirm that your file was turned in. The **turnin** program must be run ON THE COMMAND-LINE ON LECTURA (not from a windows icon!)

### **Problem 3 (20 points): vi.**

Copy the file **/home/cs352/spring08/assign0/problem3.txt** to your directory. Answer the **vi** questions that are contained in the file.

It is not required that you actually use **vi** to answer the questions. It is recommended, so that you can gain some experience with **vi**. If you choose not to use **vi**, and you use a Windows computer, then there is an extra step needed. After you have edited the file on your Windows system, upload it to your CS account. Then, on lectura, use the **dos2unix** program on the file before you turn it in. There is a manpage for **dos2unix**. (If you use vi on lectura to complete this problem, you do not need to use the **dos2unix** program.)

**Turnin:** Use the **turnin** program to turn in your **problem2.txt** file. The command is:  
**turnin 352assign0 problem2.txt**

See the man page for the **turnin** program for details on what turnin can do and how you can confirm that your file was turned in. The **turnin** program must be run ON THE COMMAND-LINE ON LECTURA (not from a windows icon!)

### **Problem 4 (20 points): Korn script.**

You will write a very simple Korn shell script named **problem4.ksh**. The script will consist of seven lines. The first line will be:

```
#!/bin/ksh
```

The second through seventh lines will contain your answers to parts a through e below. Each question is to be done with exactly one Unix command-line.

a.) Use **ls** to show the short listing format for all entries in the directory **~cs352/spring08/assign0/**, including the “hidden” files. All directories should show with a slash **/** at the end of the filename, and all symbolic links should show up with **@** at the end of the filename.

b.) Use **ls** to list all the entries in the directory **~cs352/spring08/assign0/** in the long listing format and arranged by the reverse modification time (most recently modified entry should be last). No hidden files should be shown and the file sizes should be in human readable form (showing the number of bytes (or Megabytes, depending on the size of the file) in the files, not the number of blocks).

c.) Use **fgrep** to print all the lines in the file `~cs352/spring08/assign0/README` that contain the part of the word Pick (case doesn't matter, so any lowercase variant such as pickling, PICKLINGTOOLS) but NOT the word pickle (case matters, so only exclude the word pickle, not PICKLE). (Hint: I found 22 lines)

d.) Use **fgrep** to print all lines in the file `~cs352/spring08/assign0/README` that contain both the words midas (all lowercase, case matters) and MidasServer (case matters). The order of the two names within a line does not matter. (Hint: I found 4 lines)

e.) Count the number of processes of `lectura` using `httpd` daemon from `apache`. Remember to exclude any `fgreps` looking for `httpd`! This is dynamic and will change depending on who is running on the machine at the time.

f.) Use **fgrep** to search two files (in the same command) for the word Tool (case-matters). The first file exists and is `~cs352/spring08/assign0/README` and the second file does NOT exist and is called **nothing**. Redirect the standard output (which shows all the matches) to a file called **stdout.txt** and redirect the standard error (which tells is that there is no file called ) to a file called **stderr.txt**.

It is not required that you use **vi** or **emacs** to write your script. It is recommended, so that you can gain some experience with **vi**. If you choose not to use **vi/emacs**, and you use a Windows computer, then there is an extra step needed. After you have edited the file on your Windows system, upload it to your CS account. Then, on `lectura`, use the **dos2unix** program on the file before you turn it in. There is a manpage for **dos2unix**. (If you use **vi** on `lectura` to complete this problem, you do not need to use the **dos2unix** program.)

**Turnin:** Use the **turnin** program to turn in your **problem4.ksh** script. The command is:  
**turnin 352assign0 problem4.ksh**

See the man page for the **turnin** program for details on what **turnin** can do and how you can confirm that your file was turned in. The **turnin** program must be run ON THE COMMAND-LINE ON LECTURA (not from a windows icon!)

### **Problem 5 (20 points): C-Shell script.**

You will write a very simple C-shell script named `problem5.csh`.

Redo problem 3, but make this be a C-Shell script instead.

It is not required that you use **vi** or **emacs** to write your script. It is recommended, so that you can gain some experience with **vi**. If you choose not to use **vi/emacs**, and you use a Windows computer, then there is an extra step needed. After you have edited the file on your Windows system, upload it to your CS account. Then, on `lectura`, use the **dos2unix** program on the file before you turn it in. There is a manpage for **dos2unix**. (If you use **vi** on `lectura` to complete this problem, you do not need to use the **dos2unix** program.)

**Turnin:** Use the **turnin** program to turn in your **problem5.csh** script. The command is:  
**turnin 352assign0 problem5.csh**

See the man page for the **turnin** program for details on what turnin can do and how you can confirm that your file was turned in. The **turnin** program must be run **ON THE COMMAND-LINE ON LECTURA** (not from a windows icon!)

### **Final Summary:**

There are five files that you will need to turnin for full credit on this assignment:

- args.java**
- problem2.txt**
- problem3.txt**
- problem4.ksh**
- problem5.csh**