Lab 0 – Getting Reacquainted with C and Unix

Due: 9:55 AM, Tuesday, February 8, 2005

1. Send me mail at *terescoj@cs.williams.edu* with a brief (a couple sentences) indication of your level of experience with the Unix operating system and its variants, plus a list of other operating systems you have used. Also include list programming languages you have used and your proficiency in each, and anything else you'd like me to know about your background coming in. (0 points)

2. (0 points) Log into and familiarize yourself with your CSLab Unix account.

   - Forward your CSLab electronic mail to an address you read regularly. I may sometimes use your *@cs.williams.edu* address and will expect that you will read what it sent there on a regular basis. The class mailing list may also use this address.
   - Try FreeBSD systems in the lab (see the online version for a link to a list of machines) and the Solaris cluster (*bullpen*). Connect remotely from anywhere using your favorite ssh client, and log into the consoles in TCL 312.
   - Identify the function of and experiment with these Unix Commands:

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

   - Emacs is a powerful text editor that you probably want to become familiar with. Identify the function of and experiment with these Emacs Commands:

     ```
     C-x C-s   C-x C-c   C-x C-f   C-x C-w   C-g   C-a   C-e
     C-d       C-_       C-v       M-v       C-s   C-r   M-%
     ```

     Learn these commands – you will use them often. Hints can be found in the Unix and Emacs web pages on the course website.

   - Create a directory in your account for work from this class. Change the permissions on the directory so only you have read or write access to it.

3. Copy the C program on the online version of this page that computes the late penalties for this course to your CSLab Unix account. Compile and run it, redirecting your output to a file `late.txt`. (1 point)

4. Copy the file on the online version of this page to your CSLab Unix account, either from this link, or from `/home/faculty/terescoj/shared/make-example.tar`. It is a "tar file" of a small C program that demonstrates the use of multiple source files and `Makefile`s. Extract the files (`tar xvf make-example.tar`) and compile the program with `make`. In a plain text file called `make.txt`, briefly describe how `make` uses the rules in the `Makefile` to produce the executable `main`. (1 point)

5. You've all used Java applications that take command-line parameters. Write a C program that takes an arbitrary number of command-line parameters, each of which should represent an integer value. Print out the sum of the values provided. Call your C program `argadder.c` and include a `Makefile` that guides compilation of your program into an executable `argadder`. (3 points)

   Hint: See `main.c` in the make example, and note that the parameter `argc` to the `main` function is a count of how many command-line strings are included in the `argv` array of strings. Also, `argv[0]` is not the first parameter, it is the program name itself, and the program name is included in the value of `argc`.

To submit this lab, create a "tar file" called `lab0.tar` that contains your files to be submitted (`late.txt`, `make.txt`, `argadder.c`, `Makefile`). To do this:

```
tar cvf lab0.tar late.txt make.txt argadder.c Makefile
```

Then use the `turnin` utility on any FreeBSD system in the lab to submit your work:

```
turnin -c 432 lab0.tar
```

Please use the filenames specified here and all semester. Don't include your name in the tar file name. The `turnin` utility will store things in a subdirectory based on your Unix userid. Consistent filenames make my job when extracting everything for grading, and you don't want to annoy your grader with misnamed or missing files just before he grades your assignment.