CSC 104 Lab 2, 27 January 2012

This lab comprises 4% of your grade in this course. It must be completed by the end of the lab time you have signed up for. **You must sign up for a lab time on the course web page**; you are not enrolled automatically. Please sign up well in advance.

This lab can be done individually or in pairs. No more than two people can work together on parts B or D of this lab.

Please type only into your own account, and don't let anyone else use your account. If working in pairs, one student should use their account for part B, and the other student should use their account for part D.

Part A: Instruction

This lab will begin with some instructional material using the picture at

```
http://www.dgp.toronto.edu/~ajr/104/tut/03/fsys.gif
```

Please load this picture into your web browser to follow along.

Part B: File path names

This part can be done individually or in pairs.

This part should be done entirely in the terminal window (and in nedit). Commands to use include cd and ls, which will be introduced in part A.

1. Start *nedit*, into which you will record your results. If you start nedit by typing ''nedit'' into a terminal window, instead type ''nedit&'' so that you get your prompt back for further commands.

2. Type ''ls'' to see a list of your files in your ''current directory''. This is your ''home directory'' by default. Now type ''ls /local'' to see a list of the files in /local.

3. Type ''cd /local'', and then ''ls'' (two separate commands). Why does ''ls'' now show something different? What was it showing in question 2 (when you did simply ''ls'') and what is it showing now? Answer these questions very tersely in your nedit document (identified as the answers to question 3)—do not list the output of the two occurrences of ''ls''; state what they represent.

4. Type ''pwd'', which shows you what your current working directory is. Type ''cd'' with no arguments (i.e. just the two letters, then press return) to set your current working directory back to your home directory. Find out what the absolute path name for your home directory is, and type it into your nedit document (identified as the answer to question 4).

5. In a terminal window, examine the directory /u/turing/test.
5a) Name one file in this directory.
5b) State an absolute path name for that file.
5c) If your current working directory were /u, what would be a relative path name for that file?
Answer these questions in your nedit document as well.

6. Save your nedit document into the file name ''partb'', and submit it for marks with the command:

```
submit -c csc104h -a lab2 partb
```

It should contain answers to questions 3, 4, 5a, 5b, and 5c.

If you are working in a pair, also submit a file named ''bpartner'' (partner for Part B) which says who the other student is, so that you both get credit for this work. The ''bpartner'' file should contain the CDF account name of the other student only (and no other text).

(continued)

Part C: Terminal-oriented e-mail program

This part of the lab can be done in pairs or in groups of three.

To get the marks for this part, you must send and read mail from your CDF account, and you must do so using ALPINE. (Yes, there are many good e-mail programs which one can use, in general; however, you should be able to use a particular one of them if directed!)

ALPINE often displays important messages at the bottom of the window, so remember to look there while using it.

(Suggested changeover procedure: Suppose your names are Alice and Bob, and you are logged in to Alice's account in which you've just finished Part B. Your neighbours' names are Xerxes and Yolanda, and they are logged in to Xerxes's account in which they've just finished Part B. First Alice and Xerxes can send each other e-mail and comply with part C of this lab; then they log out and Bob and Yolanda can log in and send each other e-mail and comply with part C of this lab; then you can stay logged in to Bob and Yolanda's accounts to proceed with part D.)

1. Use the ALPINE e-mail program on CDF to send a message to another member of your pair or group. To start ALPINE, you can type ''alpine'' (not capitalized) at a command prompt (and then you can press 'e' to get past the ''greeting'' screen if applicable).

Press 'c' (for ''compose'') to start a new e-mail message. CDF e-mail addresses are of the form username@cdf.toronto.edu. The e-mail message can have any subject and any message contents, but everyone must send a message to someone else (not to themselves).

You will get the marks for this item when your addressee saves and submits the message as in the next point. If this turns out to be a problem for some reason, please send explanatory e-mail to ajr *from ALPINE on CDF* to get the marks.

2. Use ALPINE to read your message from the other member of your pair or group. Press 's' to save it into a file. Use the file name ''ex'' (short for ''example'').

This 's' command has actually put the message into a file by that name inside your ''mail'' subdirectory. You can therefore view it from a command prompt by typing ''cat mail/ex''. And you should submit the file for grading purposes with the command

```
submit -c csc104h -a lab2 mail/ex
```

Your message must have been saved from ALPINE, and for full marks, your submitted file must contain only one e-mail message. If you save multiple messages in this file by mistake, you can simply delete it and start again.

3. Use the 'q' command (while not composing a message) to quit the ALPINE program.

Part D: Files which *do* stuff

This part can be done individually or in pairs (but not in threes). If you worked in someone else's account for Part B, with that other person typing, then you must use *your* account for Part D, with *you* typing (and clicking).

1. Create a program in the ''shell script'' programming language to display a short message, as follows. Use nedit to create a one-line file in your home directory. Its contents will be the ''echo'' command followed by your desired message. Most punctuation, including quotation marks and apostrophes, requires extra care, so omit that for now. Example:

```
echo Hello I am a friendly computer
```

Now run your little program by using a terminal window and typing ''sh file'', where ''file'' is the file name you saved your shell script as. (''sh'' stands for ''shell''.)

2. Shell scripts can express complete programs, but they're clumsy for some programming purposes. However, they have the great advantage that they can execute any unix command at all. When you type ''sh file'', each of the commands in the file is executed in turn, just as if you had typed them in the terminal.

(continued)

Experiment in a terminal window with the ‘‘expr’’ command, which does simple calculations. For example, type

```
expr 2 + 3
```

The numbers and operator must all be separate words, i.e. separated with spaces.

Multiplication is trickier because ‘*’ is a special character to the shell, so avoid this for now. Integer division is available using ‘/’ to get the quotient and ‘%’ to get the remainder. (Mnemonic: ‘%’ looks kinda like ‘/’.)

3. Shell scripts can refer to the ‘‘arguments’’ to the script as ‘‘$1’’, ‘‘$2’’, etc. Make a file (let’s say it’s called ‘‘file’’) whose contents is this:

```
echo You typed $1
```

If you type ‘‘sh file blahblah’’, it will output

```
You typed blahblah
```

On the other hand, if you type ‘‘sh file two things’’, it will output

```
You typed two
```

because ‘‘things’’ is ‘‘$2’’.

Experiment with this.

4. Now, write a shell script named ‘‘add’’ which is going to behave as follows, where the user types the first command to the shell (whose prompt is indicated here with a ‘%’):

```
% sh add 28 34
Ah, I see you want to add 28 and 34
The sum is
62
```

5. Submit ‘‘add’’ for marks. If you are working in a pair, also submit a file named ‘‘dpartner’’ (partner for Part D) which says who the other student is, so that you both get credit for this work. The ‘‘dpartner’’ file should contain the CDF account name of the other student only (and no other text).

6. (optional) If you’re wondering about multiplication with the expr command: The solution is to ‘‘escape’’ the ‘*’ symbol by preceding it with a backslash. Example:

```
expr 2 \* 3
```

This can also be used with the echo command to output special characters. Example:

```
echo I\'m special
```