CSC 357 Lab 1
Getting Started with UNIX and C

ISSUED: Monday, 2 April 2007
DUE: On or before 11:59:59PM Monday 9 April, via `handin` on falcon/hornet
POINTS POSSIBLE: 100
WEIGHT: 2% of total class grade
READING: Lecture Notes Week 1, "UNIX Basics" Handout, K&R Chapter 1, Stevens Chapter 1, various cited `man` pages

Tasks

1. Login to hornet or falcon and cruise around the 357 directory.

2. During the course of the lab, familiarize yourself with the user-level UNIX commands described in the "UNIX Basics" handout.

3. Start Emacs and read at least the first section of the tutorial. You can run Emacs on hornet through a terminal, or locally on one of the lab machines.

   The normal way to start the emacs tutorial is by typing 'control-h' followed by the letter 't'. That is, hold down the control key when you type 'h', then type 't'.

   Unfortunately, some terminals map the 'control-h' key to the 'delete' function instead of the separate 'backspace' function that it should be. You'll notice this when you type 'control-h' and it deletes a character in the edit buffer, or rings the bell, rather than starting the help function. To get around this problem, do the following to start the tutorial:

   a. Type 'escape-x', that is, type the 'escape' key followed immediately by the 'x' key.
   b. In response, emacs will put the prompt "`M-X`" at the very bottom of its display window.
   c. Type "help" followed by the Enter key at the prompt.
   d. Emacs will then display another prompt at the bottom of the window, starting with "Type one of the options ..."
   e. Type the letter 't', for 'tutorial', after this prompt.

4. Investigate other editors and IDEs, and consider which you'd like to use for 357.

5. Using an editor of your choice, type in `hello.c`. Then compile and run it on hornet.

6. Copy `hello.c` to `hello5.c` and do the following:
   a. Modify the program so that it prints out "hello world" five times, using a `for` loop.
   b. The output must be on five separate lines, but must not have an extra blank line at the end. I.e., the output must look line this, where "`> `" is the Unix prompt

```
> hello5
hello, world
hello, world
hello, world
hello, world
hello, world
>
```

c. Compile and run `hello5.c` on hornet.

7. Read the K&R discussion, the Stevens discussion, and the man pages for the following C library functions:
   - `printf`
   - `scanf`
   - `strlen`
   - `strcat`
   - `atoi`

8. Convert the following Java program to C, storing it in a file named `lab1.c`. Compile and run the program. The converted C program must accept the same form of input and produce precisely the same output as the Java program, when the Java program produces defined output.

```
/****
 *
 * This program inputs a numeric value from stdin and performs some
 * computations with the input.  The compuations are performed on the input
 * value in both string and numeric forms.
 *
 * The program has undefined behavor if the supplied input is not an integer of
 * at least three digits.
 *
 */

import java.io.*;

public class lab1 {

    public static void main(String[] args) throws java.io.IOException {

        BufferedReader stdin =
            new BufferedReader(new InputStreamReader(System.in));
        String line;

        System.out.print("Input a number and press the Enter or Return key: ");
        line = stdin.readLine();

        System.out.println("\nThe number has " + line.length() + " digits");

        System.out.println("The third digit of the number is " + line.charAt(2));

        System.out.println("The number times 100 is " + line + "00");

        System.out.println("The number plus 100 is " +
                                (Integer.parseInt(line) + 100));

    }

}
```

Hints:

a. As discussed in Lecture Notes 1, a string in C is represented as a character array. See page 29 of K&R for an example of how to declare an input string variable of the type used in this program.

b. The `scanf` function may look a bit intimidating from its description in Section 7.4 of K&R. However in this program, its use can be as simple as "`scanf("%s", line)`", with a suitable definition of the variable `line`.

c. In the second-to-last line of `lab1.java`, the following expression concatenates the string variable `line` and "00":

```
line + "00"
```

You might consider using the following call to `strcat`

```
strcat(line, "00")
```

to accomplish the same thing in C. If you do, you'll probably get an incorrect result somewhere in the output. So, instead of the string concatenation gimmick, try using integer arithmetic to compute the number times 100.

9. Consider the following statement in the Javadoc comment for `lab1.java`:

The program has undefined behavior if the supplied input is not an integer of at least three digits.

Describe in your own words the meaning of "undefined behavior" in Java compared to C. Limit your description to one half page of text or less. For hand-in purposes, put your answer in a ***plain text*** file named "`behavior.txt`".

"Plain text" means just that. In particular, it's NOT a Microsoft Word file, an HTML file, or a PDF file.

Deliverables

You must submit the following three deliverables for this lab:

1. `hello5.c` -- the modified version of `hello.c`, done for Task 6
2. `lab1.c` -- the C version of `lab1.java`, done for Task 7
3. `behavior.txt` -- the discussion of "undefined behavior" in C versus Java, done for Task 9

The deliverables must be in three separate files, named ***exactly*** as shown above, including spelling and all-lower-case characters.

Scoring Details

`hello5.c` is worth 20% of this lab, `lab1.c` 55%, and `behavior.txt` 25%.

Collaboration Allowed

Lab partners may submit the same programs for the first two deliverables. Everyone must do their own individual work for the third deliverable.

How to Submit the Deliverables

Submit your deliverables using the `handin` program on falcon/hornet. The `handin` program is a simple command-line application that takes several arguments. The first argument is the name of the recipient user, which is always "`gfisher`" for this quarter of CSC 357.

The second argument is the name of the assignment being turned in. For this lab, the name is "`lab1`".

The remaining arguments are the names of the files to turn in. For this lab, the files are the three cited above, namely `hello5.c`, `lab1.c`, and `behavior.txt`.

So, the command to run for this lab is the following:

```
handin gfisher lab1 hello5.c lab1.c behavior.txt
```

You can confirm that your submission was successfully received by running the `handin` command without the file arguments, i.e.,

```
handin gfisher lab1
```

You can resubmit your files as many times as you like, up to the submission deadline. Each new submission completely replaces the previously submitted file(s).

IMPORTANT NOTE: If lab partners elect to submit the same programs for the first two deliverables, they must still run `handin` individually to submit their work.