Lab 1: Unix and C

Objectives

- to become familiar with commands available at the Unix command line.
- to learn about redirection and pipes in Unix.
- to experience writing some simple C programs.

A secondary goal for CSCI 210 is to become more familiar with using the Unix operating system and the C programming language. In this lab we will explore these two elements.

A The Unix command line

Being proficient with Unix requires a good understanding of the Unix command line, so we'll investigate different commands and their use in the Unix environment.

You can find a tutorial named *Unix Tutorial for Beginners*, by Michael Stonebank, at the following Web address.

`http://www.cburch.com/cs/210/unixtut/`

You are to study this tutorial during the lab. The tutorial contains exercises; I recommend them, but they should not be part of your laboratory report.

The questions of this lab assignment are based on the tutorial material. After you have completed the tutorial, write a single command to accomplish the following tasks, and explain in English how the command works. For each task, I've included a brief transcript illustrating how your command should work.

1. Delete all the files in the current directory whose name ends in "`.bak`".

   ```
   % ls
   science.txt  science.bak  main.bak
   % your command here
   % ls
   science.txt
   ```

   The command should work no matter what is in the directory — if the directory also contained the file "`a.bak`", for example, the same command should remove it too.

2. Display lines 10 to 14 of `science.txt`.

   ```
   % your command here
   at it in something close to wonder.

   But this was Fabricators' Week at the Herstmonceux Science Centre, with
   exhibitors from science centres all over Europe arriving to demonstrate
   prototypes of experiments they hoped to produce as hands-on displays - a tube
   ```

   To accomplish this, you'll want to use the `head` and `tail` commands.

3. Count how many lines contain the word *science* in `science.txt`. (The answer is 9.)

   ```
   % your command here
          9
   ```

   To accomplish this, you'll want to read the `man` page for `wc` to learn about the `-l` option.

4. List the statistics of the largest file (and only the largest file) in the current directory.

```
% cd /usr/people/classes/writeups/lib
% your command here
-rw-r--r--   1 cburch   cs          7767 Aug 28 20:13 science.txt
```

You'll want to learn about the `-k` option to the `sort` command.

The body of your report for this section needs only to contain the commands you composed to accomplish each of the four tasks, and your brief English description (one to four sentences) explaining what each command means.

B A short C program

Goldbach's conjecture is that any even number more than 2 can be written as the sum of two primes. For example, the number 18 is the sum of 7 and 11, both prime. Your C program should read a number from the user and displays two prime numbers whose sum is that number.

```
% ./a.out
Number? 18
7 11
```

If there is no such prime number, the program should print a message to this effect.

Your program should consist of three files. The first file, "`prime.c`," will contain the following two functions.

`int isPrime(int n)`
: Returns a non-zero number if `n` is prime, and 0 if `n` is composite.
(Note that the easiest way to determine if a number is prime is to iterate some variable i from 2 up to where $i^2 > n$. If i divides into n exactly (remember the remainder operator `%`), then the number is not prime and you can return 0 immediately. If you get to where $i^2 > n$, then you can return 1.)

`int findGoldbach(int n)`
: Returns a prime number a where $n - a$ is also prime, or 0 if there is no such prime number. This function should not print anything for the user to see: That's the job of `main()`.

The header file "`prime.h`" will contain prototypes for the functions of "`prime.c`." And the final file, "`interface.c`," will contain the `main()` function.

The `gcc` command will compile a program using C.

```
% gcc prime.c interface.c
```

It will compile the program into a file called "`a.out`," which you can then execute on your computer.

To tell the computer to read a number from the user, you will want to use the `scanf()` function declared in `stdio.h`. You can use the following line in your `main()` function.

```
scanf("%d", &n);
```

This will read a number from the user, placing the number into the `int` variable `n`. (We'll discuss the `scanf()` function more in class when we get to Section 2.2.2.)

Lab report

As in your previous classses, each lab report in this class should contain an introduction, body, and conclusion. For this lab, the body will contain your Unix commands and decriptions from Part A and the text of your program from Part B.

Please note that throughout this course, both on tests and laboratory reports, I will evaluate your writing, particularly how well you describe concepts. Proving to me that you understand is not enough; what you write should be understandable to others who don't have such a strong grasp. Think about your explanations, and proofread carefully.

You should also use electronic hand-in to submit your completed program ("`handincs 210 1`").