Homework 3

Due: November 8, 2019 **Points:** 100

UNIX/Linux Questions

Please put the answers to these two questions in a text file called *linux.txt* or a PDF file called *linux.pdf*.

1. (*10 points*) What Linux program could I use to generate a random number?

2. (*10 points*) What is the name of the process with PID 4?

C Programming

3. (*30 points*) Write a program to search for words in a file. Your program is to work like this.
   First, read in a file of no more than 1000 words (called a *dictionary*), and store them in a list of character pointers, each pointing to a word. You will have to use *malloc*(3), which allocates space dynamically. Each line, including the last, will be ended with a newline, and no line will be over 1024 characters.
   Then, read a second file. This file is also a list of words. Search your list for each word, and for each word print the number of words checked before you either get a match or find it is not in the list. For letters, the case must match (so "N" does *not* match "n"). Also, after you have finished searching for the words, print the *average* number of words checked.
   The two files are given as command-line arguments to your program. The first argument is the dictionary, and the second is the list of words to be found.
   To turn in: Put this program into a file called *lookup.c*; turn it in as directed below.

4. (*30 points*) Now we will do the next step in the Game of Life program we began in Homework 2. For reference, the program *life.c* in the directory */home/bishop/hw3-programs* draws a board. You can use that, or the one you did for homework 2.
   The next step is to load a predefined pattern onto the board for the game. The predefined pattern is in a file. That file contains rows of "X"s and "O"s (the characters "X" and "O"). The "X" corresponds to a populated cell and the "O" to an unpopulated one. Each line in that file should have the same number of columns. Your program is to read the file, load the configuration into memory, and print it out, properly framed by the board, and stop.
   Your program will take the file with the predetermined pattern as its only argument. It is to print out the board and quit. Note your program will have to count the rows and columns as these will not be given. Report any errors such as a line too long or too short.
   As an example, the inmput file

   ```
   XOXXOOXOX
   OOOXXXOOO
   OXOXOXOXO
   XOXOXOXOX
   ```

   defines a board of 4 rows and 9 columns. It is a valid board. But the board

   ```
   OXOXO
   XXXX
   OOOOOO
   ```

   is not valid as the rows are of different lengths, and the board

   ```
   XOXXOOXOX
   OOOXYXOOO
   OXOXOXOXO
   XOXYXOXOX
   ```

is not valid because there is a character other than X, O, and newline in the file (specifically, two instances of "Y"). Your program is to report errors in these cases, and then terminate.
The board dimensions will never be more than 100 rows and 100 columns.
To turn in: Put this program into a file called *life2.c*; turn it in as directed below. If you want to submit two files, that is, have this in a separate file, call the main one *life2.c* and the other file(s) *initn.c*, where *n* is a single digit (for example, *init0.c*). Turn it in as described below.

C Program Debugging

5. (*20 points*) The program *revfile.c*, available at */home/bishop/hw3-programs* on the CSIF, is to read in up to 1023 lines of input, the maximum input line length being 1023 characters. It will then print the lines in reverse order. But it doesn't; it crashes. Please identify the problem, and say how to fix it. Give the specific line(s) where the problem arises, and the code that you would fix it with.

   To turn in: Fix the bug and say what you did in a comment at the beginning of the file. Turn in the *fixed* file as described below.

Extra Credit

6. (*10 points*) Take the program you wrote for problem 3. Once you have read in the word list from the dictionary, sort the words (you an do this yourself or use the library function *qsort*(3). Then proceed as in problem 3. So, you will read a second file, search the list for each word in the file, and for each word print the number of words checked before you either get a match or find it is not in the list. Also, after you have finished searching for the words, print the *average* number of words checked.

   To turn in: Put this program into a file called *lookups.c*; turn it in as described below.

Turning In Your Homework

For this assignment, you are to create a *tar*(1) archive and a *Makefile*.

The *Makefile* is to have 3 or 4 targets: *lookup*, *life2*, *revfile*, and (if you do the extra credit) *lookups*. So, when I type `make lookup`, the *Makefile* will execute the command to compile your *lookup.c* program, and save the executable as *lookup*.

Put the answers to the Linux questions (in a text or PDF file), the *Makefile* and the programs you wrote into a directory called *hw3-ans*. Next, create a *tar* archive:

```
tar cvf hw3-ans.tar hw3-ans
```

or whatever files you wish to submit. Then submit the *tar* archive to Canvas.