Warm up Programming Project

- Three weeks, due on Friday (02/21)

Implement a substring search program:

```
% echo "abcdefgh" > bar
% ./p01 bar bc
1
%
```

Level 1

The application (**`p01`** in the example) requires at least two arguments. The first argument (**`bar`** in the example) specifies the file in which you will be finding sub-strings.

The third and remaining arguments (**`bc`** in the example) are sub-strings that are to be searched for in the file specified by the first argument. In our problem, "sub-strings" are not necessarily separated by whitespace; that's why we're calling them "substrings" instead of "words".

Let's look at another example:

```
% echo "aaab" > bar
% ./p01 bar aa B
2
1
%
```

This shows that the substring 'aa' occurs twice in 'aaa'. Also, the program must handle multiple search strings on command line and searches are case-insensitive.

Your program must be robust:

- The output **must** look exactly as shown in the examples, there should not be any extra whitespace or other characters in the output lines
- You **must** check for possible failure conditions and report them with the appropriate error string defined by Linux.
- You **can** read the entire file at once, but the program **must** fail gracefully if the file does not fit into memory. You cannot statically allocate memory for the file.
- You cannot use uninitialized or unallocated memory.

The program should be written in C and must include a Makefile. Submission will be in a tarball (e.g., run *tar cvzf username-p01.tar.gz username-p01* on the directory with all your source files and Makefile).

Level 2

Provide an additional command line argument --systemcalls that switches between 2 different implementations of this assignment:

- If the command line flag is provided, the program should read each individual character from the input file separately using a direct system call read.
- Otherwise, the program should either read each character individual using the stdio library, or read the complete file with a single read.

Measure the runtime of both version of your program. For this experiment, generate a test input file of size 10MB and run both version of your implementation using the **time** command over this file. Report the runtime numbers and briefly explain the result in a file runtime.txt

SUBMISSION INSTRUCTIONS

The assignment must be completed **individually**. You can and are encouraged to help each other with programming environment (e.g., editor, compiler) problems and discuss general algorithms but you cannot look at each other's code. I expect you to be familiar with your responsibilities under the Policy of Academic Integrity.

Submission instructions:

1. Create a gzipped tar ball that includes all your source code

   - The filename for the tarball **must** be `username-p01.tar.gz` where username is your login name (NetID) on the EIT system
   - All your files in the tarball **must** be in a directory, named `username-p01`
   - The tarball **must** contain either a Makefile or a configure command to generate a Makefile
   - The source code must be in a file named `p01.c`
   - The executable **must** be named `p01`
   - You can use the following command to create the tarball:

     ```
     % tar czf username-p01.tar.gz username-p01
     ```

2. Test your submission with the given bash script. As mentioned in the first class, assignments will be partially graded through test inputs. To help you test your

programs for submission, this short bash script extracts your tarball, compiles the program and runs a few tests. You should see PASS for build and three test cases at the end of the output.

The bash script is *test.sh*. Run (you may need to change the first line depending on where your bash executable is located):

```
% ./test.sh username-p01.tar.gz
```

3. Upload the tarball to the Canvas assignment website.

WHY

The purpose of this assignment is to get familiar with the Linux programming environment C programming concepts such as command line arguments, pointers, IO, etc.