Assignment 4

Problem Statement:

Write a C program which will be able to read a file using `read()` system call. Now you provide a mechanism to set a lock by using `fcntl()` system call and to release the lock by using the same `fcntl()` system call. For setting up and releasing out a lock, you have to fill up a structure `flock (struct flock)` with proper values.
Now you run the program from two different terminals simultaneously and demonstrate that if one process has got the lock, then that process only can read the file. But the other one is not permitted to read the file until the lock is released by the first process. That means, only one process who acquires the lock can read the file at any instant of time.

Related Information:

- Read `read(), write()` and `fcntl()`system call from W. R. Steven's "Unix Network programming" book .
- Also open the man page for `fcntl()` to learn quickly about its usage pattern.

Hints:

- The structure for the *flock* consists of the following members...

  `l_type`

  This is where you signify the type of lock you want to set. It's either `F_RDLCK`, `F_WRLCK`, or `F_UNLCK` if you want to set a read lock, write lock, or clear the lock, respectively.

  `l_whence`

  This field determines where the `l_start` field starts from (it's like an offset for the offset). It can be either `SEEK_SET`, `SEEK_CUR`, or `SEEK_END`, for beginning of file, current file position, or end of file.

  `l_start`

  This is the starting offset in bytes of the lock, relative to `l_whence`.

  `l_len`

  This is the length of the lock region in bytes (which starts from `l_start` which is relative to `l_whence`.

  `l_pid`

  The process ID of the process dealing with the lock. Use `getpid()` to get this.

- Know about the system call `fcntl()`. The second arguments of `fcntl()` system call may be of the following types....

`F_SETLKW`

This argument tells `fcntl()`to attempt to obtain the lock requested in the `struct flock` structure. If the lock cannot be obtained (since someone else has it locked already), `fcntl()`will wait (block) until the lock has cleared, then will set it itself. This is a very useful command. I use it all the time.

`F_SETLK`

This function is almost identical to `F_SETLKW`. The only difference is that this one will not wait if it cannot obtain a lock. It will return immediately with `-1`. This function can be used to clear a lock by setting the `l_type` field in the `struct flock` to `F_UNLCK`.

`F_GETLK`

If you want to only check to see if there is a lock, but don't want to set one, you can use this command. It looks through all the file locks until it finds one that conflicts with the lock you specified in the `struct flock`. It then copies the conflicting lock's information into the `struct` and returns it to you. If it can't find a conflicting lock, `fcntl()`returns the `struct` as you passed it, except it sets the `l_type` field to `F_UNLCK`.

