System Calls for Practice

* access: checks if calling process has file access
* alarm: sets a process's alarm clock
* chdir: changes the working directory
* chmod: changes the mode of a file
* chown: changes the ownership of a file
* chroot: changes the root directory
* close: closes a file descriptor
* dup, dup2: duplicates an open file descriptor
* execl, execv, execle, execve, execlp, execvp: executes a file
* exit: exits a process
* fcntl: controls open files
* fork: creates a new process
* getpid, getpgrp, getppid: gets group and process IDs
* getuid, geteuid, getgid, getegid: gets user and group IDs
* ioctl: controls character devices
* kill: sends a signal to one or more processes
* link: links a new file name to an existing file
* lseek: moves read/write file pointer
* mknod: makes a directory, special or ordinary file
* msgctl, msgget, msgsnd, msgrcv: message passing support
* nice: changes priority of a process
* open: opens a file for reading or writing
* pause: suspends a process until a signal occurs
* pipe: creates an interprocess pipe
* read: reads from a file
* semctl, semget, semop: semaphore support
* setpgrp: sets process group ID
* setuid, setgid: sets user and group IDs
* shmctl, shmget, shmop: shared memory support
* signal: control of signal processing
* sleep: suspends execution for an interval
* stat, fstat: gets file status
* stime: sets the time
* time: number of seconds since 1/1/1970
* ustat: gets file system statistics
* utime: sets file access and modification times
* wait: waits for a child process to stop or terminate
* write: writes to a file

Exercise for Practice

* 1. Attempt to increase the priority of your program with a nice() system call. Print out the resultant error number and associated error message.
  2. Write a program that opens and existing file for writing with the O\_APPEND flag, and then seeks to the beginning of the file before writing some data. Where does the data appear in the file? Why?
  3. Two processes, client and server, communicate via two message queues "Up" and "Down".

                        Server   
                        ^     |   
                   Up |    ^ Down   
                        Client

The client reads a message from the standard input and sends it to the server via the Up queue, then waits for the server's answer on the Down queue. The server is specialized in converting characters from lower case to upper case and vice versa. Therefore, if the client sends the message "lower case" via the Up message queue, the server will read the message, convert it, and send "LOWER CASE" via the Down queue. When the client receives the message from the server, it prints it out.  You may assume the maximum size of any message is 256 bytes.

* 1. Implement producer consumer using shared memory IPC, where Consumers are now *greedy*, they will grab items off the shelf even if there are no items on the shelf (they grab a competitive brand.) Now, you shelf count can go drop below zero, but it may not rise above five. Producers are *overly diligent* and will try to overstock. Try to maintain your own count for max number.
  2. Illustrate the process of SIGNAL handling using User defined handler that prints a fixed message whenever CTRL+C is pressed.

Students are to complete any three questions during the lab hours that will be checked by the faculty. Question No 3 **or** 4 is compulsory.