Operating System Lab
Assignment#3

[ps systemcall]

1. You need to copy of command "**ls**" as "**myls**" and modify the make file accordingly to run command "**myls**" in terminal.

2. In this lab, we will use a simple example to walk you through the steps of adding a new system call to xv6. We name the system call **cps**(), which prints out the current running and sleeping processes. Create a testing file "**ps**" using cps system call to display the current running processes in tabular format, with their states and process IDs.

   First, examine the structure of per process state in the file "proc.h", and identify the fields you need to access to display the process id, process state etc.

   - You are going to need to edit the following files : defs.h, user.h, sysproc.c, usys.S, syscall.c, proc.c.
   - Assume the process state enum has the following three labels : RUNNING, RUNNABLE, SLEEPING. Before accessing the process table, sti() is used to enable interrupts on processor.
   - To gain and release lock on the process table, acquire(&ptable.lock) and release(&ptable.lock) can be used.

   The table ranges from ptable.proc till before &ptable.proc[NPROC], which are pointers of type struct proc. This is nothing but a list of proc structures. Loop through the table and print the process state accordingly by checking the fields in proc data structure.

   Do the experiment as described above. Copy-and-paste your outputs and commands to your report.

   1. Modify **cps**() in *proc.c* so that it returns the total number of processes that are SLEEPING or RUNNING.

   2. Modify *ps.c* so that it prints out a message telling the total number of SLEEPING and RUNNING processes. Copy your code and outputs to your report.