COL 331
Operating Systems

Assignment 1

Saket Dingliwal 2015CS10254

Part 1

In the first part, printing of trace of any system call was to done. Since for every system call, the syscall() defined in syscall.c is used, a small printf statement was added there**. A variable counter array** was made initialized with zeros to maintain the count of each of the calls. **Another character* array** was made for getting the name of the call from its id number. To identify which system call is made, the eax register of the current process is stored in variable num and corresponding counter and name is printed.

Since for making a new system call, various files were changed after understanding how the calls are implemented. First of all, a user program is made which calls the **user function toggle().** The function is declared in user.h as well. The definition of the function is given in usys.S where the **id = 22** is put in eax register of the process. Then, a software interrupt with value 64 is put, which is identified by trap.c as a system call interrupt. The trapframe is made and saved in the process control block. Finally, the syscall function is called by function trap(). A variable **v_toggle** is defined to tell the kernel whether the trace of the sys_call has to be printed or not. The variable is accessed in sysproc.c using extern where the **function sys_toggle()** is defined to invert the variable.

Part 2

This is done in a similar way as above except for the arguments being passed and returned. The user function **add(int,int)** is defined. The **id = 23** is put into **eax** in usys.S and arguments are saved. After the trap changes privilege to kernel mode a call to **sys_add** is made. This function is defined in sysproc.c. The function **argint** (arg_number, reference to where the output is expected) is called to get the arguments from the registers of the process which were saved while exiting the user mode. The values are simply added and the result is put back in eax register and taken out by the user process and printed.

Part 3

Another syscall needs to be implemented. The list of the processes is stored in the **ptable struct** defined in the file proc.c. This table consists of a list of process which have their structure defined in proc.h. Each process has its own **name,id and state** defined. So a **function ps()** is made in proc.c to print the pid and names of the processes by iterating in the table. Lock for the ptable is acquired and released for this. This function is made external and used in sysproc.c. The **sys_ps()** is defined in sysproc.c which just calls the ps() function. Rest of the standard process for making a system call is made as described above.