COL 331

Assignment 1 Report

Parth Singh 2015CS10242

Part 1

Variables used:

- NO_SYS: defines ad no of syscalls
- int sys_count[NO_SYS] : keeps the count of each sys_call individually
- char *sys_index2name[NO_SYS]
- Toggle_value: keeps track of whether trace is to be printed or not. ‘0’ means that trace will not be printed. Initially defined as ‘1’.

sys_toggle: Function where the implementation takes place. This is where the value of toggle_value is changed.

The name and counts are printed in the syscall function. Before printing the program checks the value of ‘toggle_value’.

‘Toggle_value’ is gotten by using extern.

Part 2

The same way we added the syscall ‘sys_toggle’, we create another syscall ‘add’. The syscall uses ‘argint’. This function gives you the value of argument passed, in the form of an integer.

Now all we have to do is the two arguments and then return their sum.

Variables used:

- num1, num2: Store the value of their guardians.

Procedure to add a syscall:

- #define SYS_add in syscall.h
- Declare it with extern in syscall.c and update the array syscalls[]
- Declare int add() in user.h
- Define int sys_add in sysproc.c. This function does the actual work.
- Add SYSCALL(add) in usys.S

Part 3

‘sys_ps ‘ is defined inside ‘sysproc.c’. This function calls another function ‘int ps()’ whose definition is written in ‘proc.c’ . It is declared as ‘extern’. We use “ptables” to get the information about current process. We loop through all the processes and check if they are “running”, “runnable”, or “sleeping” (according to the status value); if yes, the name and pid will be printed. All of the code inside ‘ps()’ is enclosed by lock.