# Osn Assignment Report

## Specification-1: Adding System calls

### Strace:

- The files to be added are as follows:
  - strace.c - contains code for strace.
- For adding a system call we need to alter the following files.
  - kernel/sysproc.c
  - kernel/syscall.h
  - kernel/syscall.c
  - user/user.h
  - user/usys.pl
  - MAKEFILE
- We have to add the system call additions in all these files.

## Sigalarm and Sigreturn:

- alarm.c file is to be added to contain the routeine for alarm systemcall.
- We need to add the systemcall alarm as mentioned in the spec1.
- And we need to modify the MAKEFILE to execute the alarm.c.

## Specification 2:

We have implemented the following scheduling algorithms:

- FCFS - First come first serve
- LBS - Lottery based Scheduling

- PBS - Priority based Scheduling
- MLFQ - Multi-level feedback queue

## FCFS: First come first serve

- CPU is allocated to the processes in the increasing order of the creation times.
- We added a new variable ctime in the process control block of the process.
- We traversed all the processes and select the process that has the least ctime and allocate CPU to it.

## LBS: Lottery based scheduling

- In struct proc we declared a new variable tickets in the PCB of the process.Each process is given a ticket initially.
- The number of tickets for the process can be changed using the system call settickets.
- We generate a tckt using the random generator and we check for the process which has the tckt and allocate the CPU to that process.

## PBS: Priority based scheduling

- We add new variables in struct proc called
  - rtime - the run time of the process
  - stime - the sleep time of the process
  - spro - the priority of the process
  - no_time - the number of times a process has been allocated CPU
  - niceness - the niceness of the process
- Added a system call set_priority that is used to update the priority of the process.
- The meaning of niceness values are:
  - 5 is neutral
  - 10 helps priority by 5

    - 0 hurts priority by 5
- The scheduler function traverses and allocates the CPU to the process that has the highest dynamic priority.

## MLFQ: Multi-level Feedback queue

- We add new variables in struct proc called
    - cqueue
    - en_time
    - qtic[5]

  in the PCB of the process in struct proc of proc.h.
- initialised variables and updated en_time = ticks in allocproc() in proc.c.
- updated scheduler() - implementing aging of process.
- there will be 5 queues denoting the set of process executed in the CPU.
- The nth queue processes start to execute if and only if the all processes in the above n-1 queues are done executing. This is the key idea of MLFQ scheduling algorithm.

# Scheduler Analysis:

- We have used the waitx file provided in the tutorial and added scheduler.c to analyse the various scheduling algorithms implemented.

| Scheduler | FCFS | LBS | PBS | MLFQ |
|---|---|---|---|---|
| rtime | 28 | 14 | 14 | 14 |
| wtime | 127 | 153 | 129 | 157 |

# Specification 3: Copy-On Write fork

- Whenever we fork a process a child and parent are created and a copy of the page table of the parent is created and allocated to the the child, this increases data redundancy even though it is not needed always.

- A copy needs to be created only if the child process not the page table of the parent needs to write in the pagetable.
- So in the Copy on Write we check if the child process writes in the pagetable and if not we can allocate the parent and child process a shared memory of the pagetable to read the data.
- else, we create a copy of the page table of the parent and assign it to the child.

## Implementation:

- Updated the uvmcopy(), copyout() functions in vm.c.
- Implemented the function pfh() in trap.c .
- Modified usertrap() in trap.c with else if condition on r_scause()
- Declared and implemented the functions,pageinit(), decpage(), getpage() and ipr() in kalloc.c

## Cow-test:

- Added cowtest.c in user for testing the syscall.
- Made necessary changes in makefile.