Enhancing xv6 OS : Report

Specification 1

- Created a dummy system call first using given instructions to check whether the call is working or not using a user program.
- Included a new variable mask for a process in proc.h.
- whenever strace is called, updated the mask variable of the process with given argument.
- Modified fork in proc.c to pass on this mask value to its child.
- by checking the **setbit-mask-syscallIndex** thingy in syscall.c, printed the required details conditionally.

Specification 2

- ### FCFS
  - Included a new variable c_time (creation time) for a process in proc.h.,
  - Initialized this c_time with ticks on allocproc() function in proc.c
  - Using #ifdef-#endif thingy, conditionally modified scheduler() function for choosing a process having lowest c_time.
  - Similarly, conditionally stop the process to yield() in trap.c file.
- ### PBS
  - Included the following new variables
    - iow-time $\implies$ Time for which process is SLEEPING
    - tot-wtime $\implies$ Time for which process is RUNNABLE
    - r_time $\implies$ Time for which process is RUNNING
    - e_time $\implies$ Ticks at which process Exited
    - SP $\implies$ Static Priority of process
    - DP $\implies$ Dynamic Priority of process
    - niceness $\implies$ Niceness of a process
    - n_run $\implies$ number of times a process has been picked by CPU ;updated in scheduler() itself
  - Updated all these values except n_run, based on the process state for every tick increment in a update_function() implemented in proc.c, called from trap.c
  - Using #ifdef-#endif thingy, conditionally modified scheduler() function for choosing a process having DP
  - If DP is the same used n_run and c_time for breaking the ties.
  - Implemented set_priority just like any other system call.
- ### MLFQ
  - Included variables
    - s_time // When was the process started becomes runnable i.e, waiting for cpu;
    - int ticks[5]; // Ticks completed in i th que => ticks[i]; reset to zero when queue is changed

        - int total_ticks[5]; // Total ticks received by the process while it is RUNNING in particular queue
        - int q_num; // The que in which the process is present
        - int w_time; //wait time for cpu of a process.
    - Updated these variables across different function in proc.c
    - Using #ifdef-#endif thingy, conditionally modified scheduler() function for choosing a processes in the highest priority queue.
    - For every cpu tick, ticks for running processes are updated and scheduler is run again to find new highest priority process available.
    - When process is created, its priority is set to 0 (highest priority).
    - If a process releases cpu voluntarily and becomes RUNNABLE again it should enter same queue.
    - To implement this when a process changes state to SLEEPING Ticks of the process for current queue is set to zero. So that when it changes to RUNNABLE again, it is as if the process created just now and enters same queue as priority is not changed.
    - Aging: If the process is waiting for too long (w_time > WTIME), it is upgraded to higher priority queue (if exists)
    - Downgrade process: If a process uses its time slice completely (Ticks > time slice*tick value for the present queue), it is downgraded to lower queue.

Specification 3

- just printed the values in procdump function in proc.c

How a process can exploit.....?

ANS:

Average waiting and running times

Method	Waiting Time	Running Time
FCFS	8201	4699
PBS	4871	4611
MLFQ	6238	4767