Report for Enhanced xv-6 Operating System

Running Instructions:

- make sure qemu is installed on your machine
- make clean (clear all the .o files_
- make qemu-nox SCHEDULER=”scheduler” (Note: No default value one values has to be added from {‘RR’, ‘FCFS’, ‘PBS’, ‘MLFQ’ }

Modifications to struct proc:

- wtime: Waiting time of the process till it runs on the cpu
- rtime: Runtime of the cpu in total
- stime: sleeping time of the process(when the process is waiting for i/o)
- etime: ending time of the process
- ctime: creation time of the process.
- num_run: number of times a process has been picked by a cpu.
- curr_queue: current queue of the process.
- enter: Time at which the process becomes runnable
- curr_ticks: ticks taken while running in one session
- queue[ticks]: Running time of the processes in each of the queues(MLFQ scheduling only)
- is_changed :Flag to check whether queue will be changed or not.

Functions Added:

1. int waitx(int *wtime, int *rtime): Works exactly similar to the wait() function but sets the parameters rtime and wtime to appropriate values of the process.
    - *wtime = p->wtime;
    - *rtime = p->rtime;
2. void pinfo(void) : Prints all the necessary information about the process
    - pid: process id
    - state: process state
    - rtime: runtime of the process on the cpu
    - wtime: waiting time of the process before it has been run
    - q[i] (i in (0,5)): Running time of the processes in each of the queues
    - num_run: number of times a process has been picked by a cpu.

Scheduling Policies Implemented:

1. FCFS Scheduling
    - In FCFS scheduling, the scheduler picks the process with the lowest ctime(creation time) by iterating on the proc table’s runnable processes and runs it.

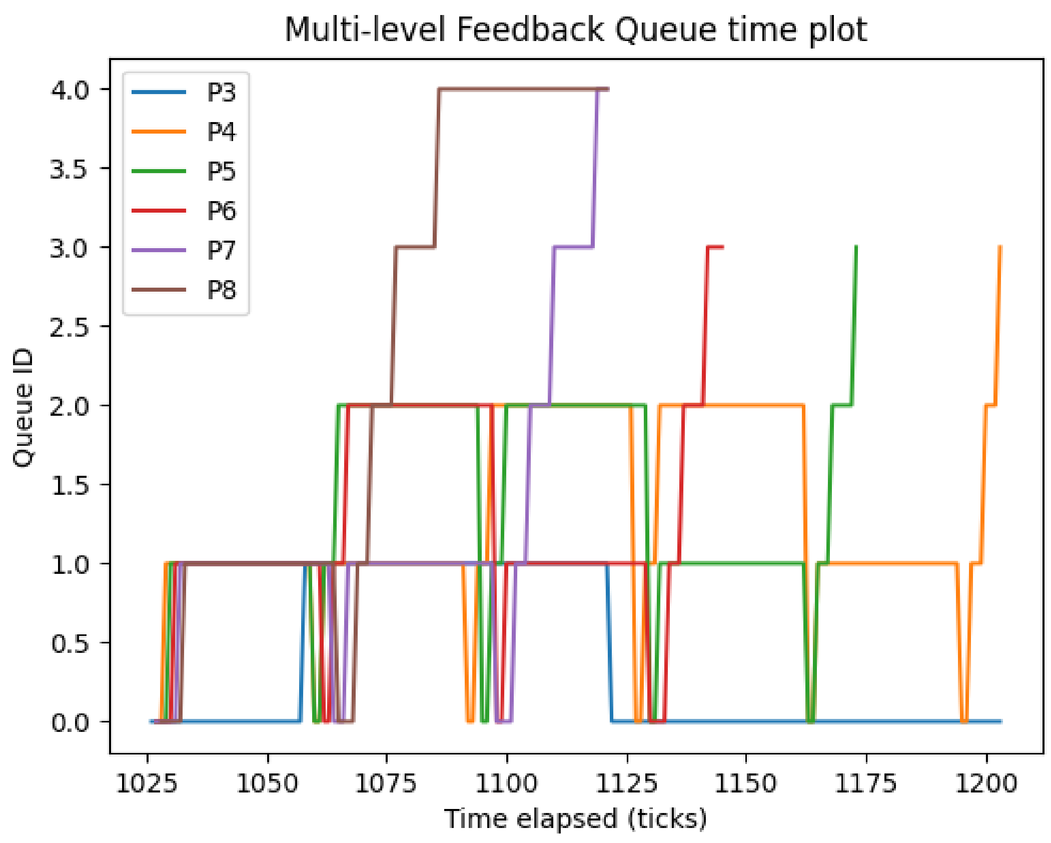
- As the scheduling needs to be non-preemptive, the yield() call should not be called trap.c if the scheduling policy is FCFS non-preemtive

2. PBS Scheduling

    a. If there are multiple processes of the same lowest priority, round robin scheduling policy is used.
    b. Setpriority (int pid, int new priority): This system call changes the priority (lies in 0,100) of the process and returns the old priority of the process. If it is set to a lower priority than before, the yield() call is called so as to start the scheduling again.
    c. Default priority of every process is 60.

3. MLFQ Scheduling
    a. 5 queues are maintained (queue[5][NPROC]) to store the processes in every queue.
    b. queue_ [top stores the last occupied index in the queue.
    c. queue_ticks_max stores the maximum time slices a process in a particular queue.
    d. Firstly for aging, by iterating over all the queues except the 0th one, we check whether some process has aged or not. If it has aged we push it up the queue.
    e. If a process completes its timeslices in a queue, is_changed flag is set to 1 and queue gets shifted down.
    f. In the last queue, round robin scheduler is implemented

Graph for MLFQ SCHEDULING:



Comparison between SCHEDULERS:

Speed of scheduling :

- Round Robin Scheduling works the fastest (only a little compared to PBS and MLFQ)
- MLFQ and PBS work almost similar
- FCFS is the slowest of them all

Conclusion:

FCFS performed the worst as CPU intensive programs came first and were served first. I/O processes had to wait for a longer time if when they had short CPU burst times. CPU intensive processes who took a large amount of CPU time get down the priority queues in MLFQ scheduling and therefore it worked better than FCFS scheduling. RR performed the best although it was almost similar to PBS and MLFQ because all the processes had the same priority and after running for a fixed amount of time, got yielded. In PBS scheduling, the speed will depend on the relative priorities of the i/o extensive processes and the cpu extensive.

Helper Functions Added:

- enque_process : Adds a process through pid(parameter) into the queues given as parameter
- deque_process: removes a process by pid (parameter) in to the queues
- increase_ticks
- increase_wait_time : updates waititme of all processes
- increase_sleep_time :updates sleep time of all processes.
- print_deets() : prints all the details for the logs that can be redirected inorder to form a plot.