Report.md

REPORT

Total time (etime-ctime) for scheduling algorithms for bench program (`bench.c`) on my machine

- RR : approx 7000
- FCFS : approx (7400 to 7700)
- MLFQ : approx 7100
- PBS : approx 7100 **(depends highly on priority given to processes so total time can increase or decrease according to priority given)**

So we conclude:

- RR performs the best
- FCFS performs the worst (performs even worse when run on 1 cpu core)
- MLFQ and PBS are better than FCFS but just behind RR

NOTE : PBS as mentioned above can perform a lot better and can also perform worst (depends on priority given to process)

Reason

- RR performed the best as both IO and CPU bound processes are given equal priority . Also since it is preemptive so any process don't have to wait for a longer time . Therefore when we run `ps` command we get the result almost immediately .
- FCFS performed the worst as it favours CPU bound . The IO bound processes have to wait even after having short CPU time.
- PBS is highly dependent on the priority given .
- In MLFQ no process waits for a lot of time because of aging and also more ticks are given to CPU bound processes as they lie mostly in lower priority queue (which runs according to RR) , therefore it performs well . Also since IO bound processes are given higher priority , so `ps` command outputs immediately .

Exploitation of MLFQ Priority Policy by a Process

If a process voluntarily relinquishes control of the CPU, it leaves the queue but when the process becomes ready again after the I/O, it is inserted at the tail of the same queue, from which it is relinquished earlier. This can be exploited by a process, if a process give up IO just before its time slice is about to over then it can retain the same priority queue again (this can happen many times). Thus it's priority will never decrement and hence the priority policy is violated.