CS 314 ASSIGNMENT 5

HRISHIKESH RAVINDRA KARANDE
210010020

PART1: Study How standard MINIX 3 scheduler schedules Jobs

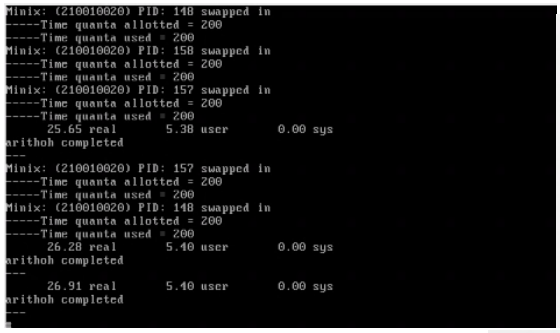
Prepare (at least 4) workload mixes having different characteristics, ranging from all compute-intensive benchmarks to all I/O-intensive benchmarks. Each workload should spawn around 5 processes(background process). You can compose these workloads of benchmarks from the UnixBench suite, or write your own.

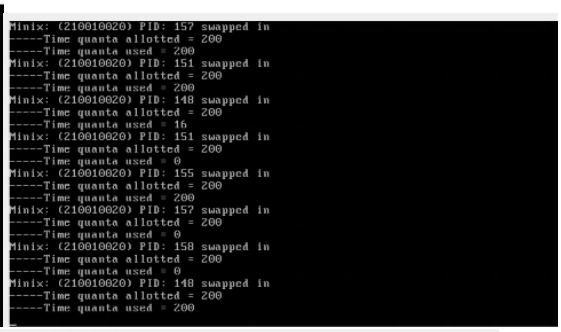
Answer:

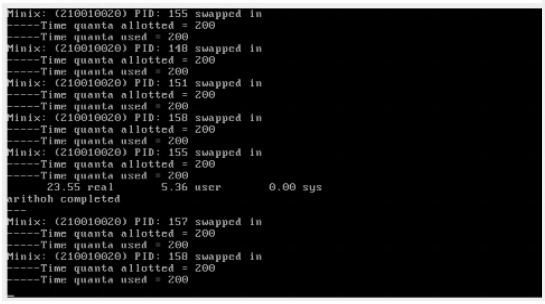
To record the time quanta for each process, we make changes in system.c file present in minix/kernel/ folder. We print out the total time quanta allotted (total quanta) and used (total quanta - CPU time left) in sched proc() function as follows:

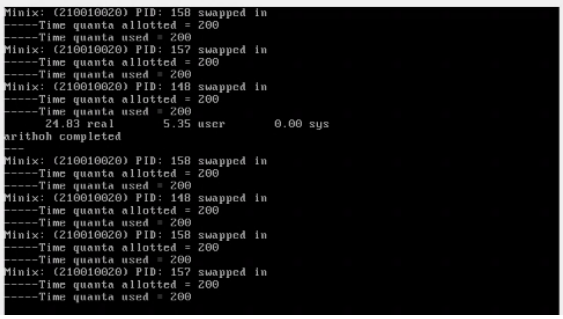
```
printf("-----Time quanta allotted = %d\n", p->p_quantum_size_ms);

printf("-----Time quanta used = %d\n", ( p->p_quantum_size_ms) - (cpu_time_2_ms(p->p_cpu_time_left)));
```

- Workload_mix1: Completely CPU Intensive.



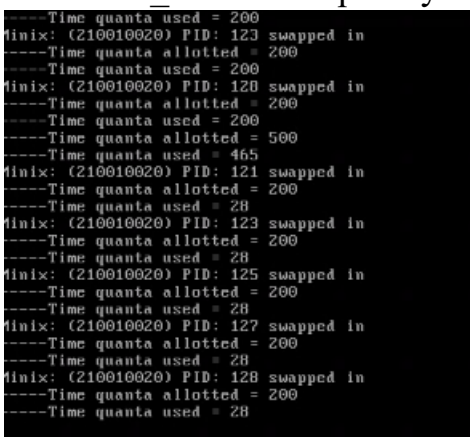


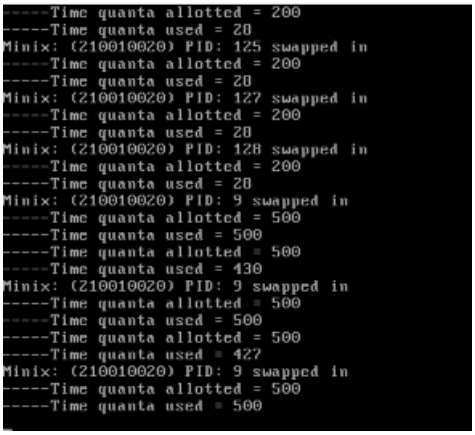




Workload1: Executes all the 5 processes as arithoh.sh, the last few processes execute in round robin manner completing it's entire time quanta of 200s which is for cpu bound processes. Each of the CPU bound process gets 2 quanta consequently to execute.

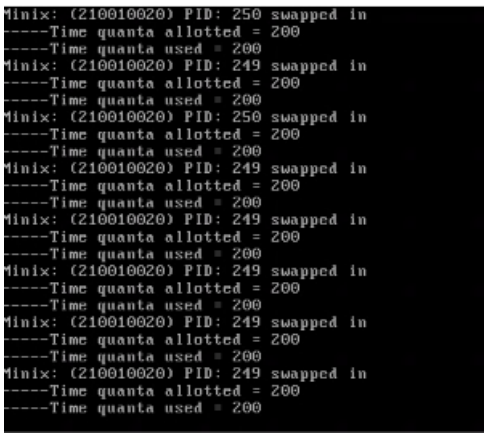
- Workload_mix4: Completely IO Bound

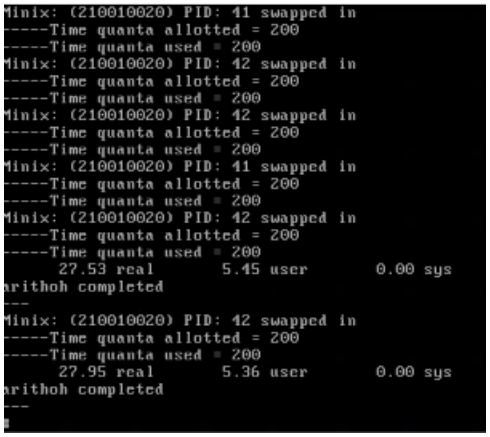




Pipe behaves CPU bound is given a time slice for 200ms and the part that executes IO given a 500 quanta in Round Robin fashion. But none completes it's complete time quanta.

- Workload2: Mix of processes



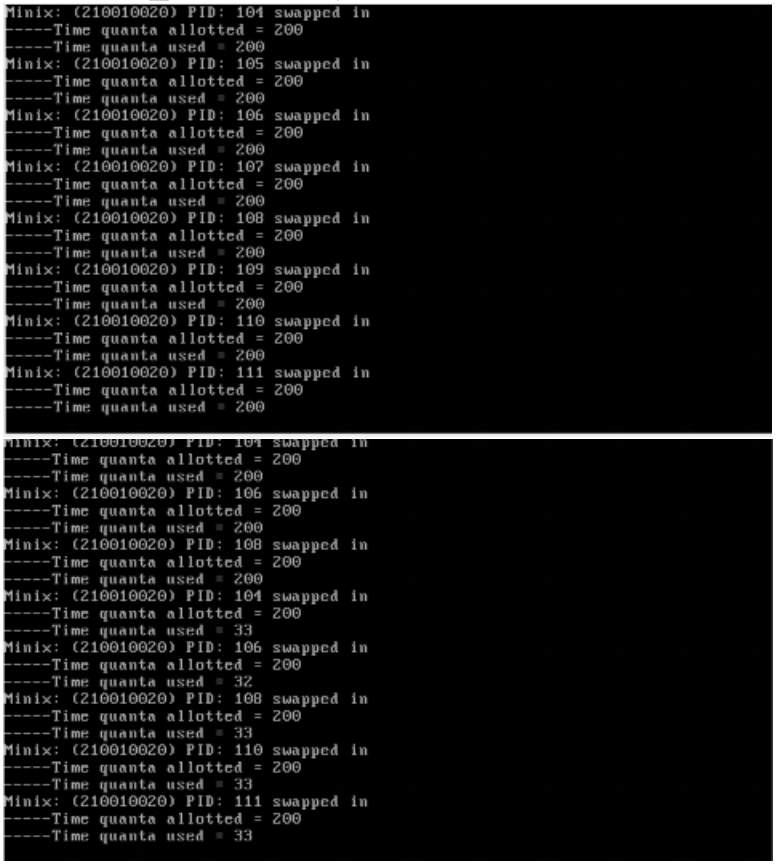


This workload is a mix of IO Bound and arithoh.sh

Each process will be allocated CPU time quanta. The exact amount of time allocated to each process depends on the round robin scheduler and the system's current load. Compute-intensive tasks like **arithoh.sh** may consume more CPU time compared to I/O-bound tasks like **syscall.sh** and **pipe.sh**. Task with process ID 249 gets scheduled multiple times. The **syscall.sh**

which is mostly IO Bound does not complete it's entire timeslice. As per this workload arithoh.sh completes execution at the end.

- Workload_mix3: All syscalls.sh



The 5 processes syscall.sh are system call-intensive i.e process relies heavily on the operating system for various tasks and makes frequent requests to the kernel via system calls to perform these tasks.. In minix, they run in round robin fashion until completion. The time quanta allotted to semi-CPU-intensive jobs is 200 ms. In the second screenshot we can observe that the processes execute for less time quanta than 200ms. They finish at almost at same time.

PART2: Modify the user-level scheduler in Minix3 to the following "Pseudo-FIFO" policy: among the user-level processes that are ready to execute, the one that entered the earliest must be scheduled.

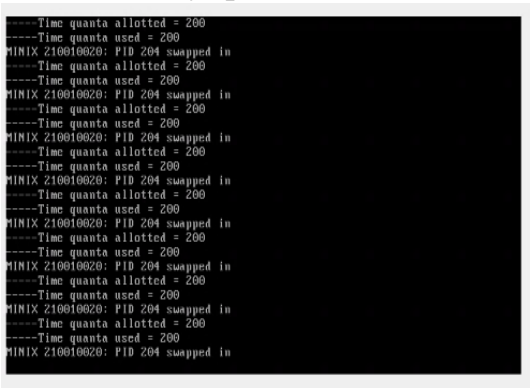
In minix/servers/sched/ folder, we modify the scheduler.c file.

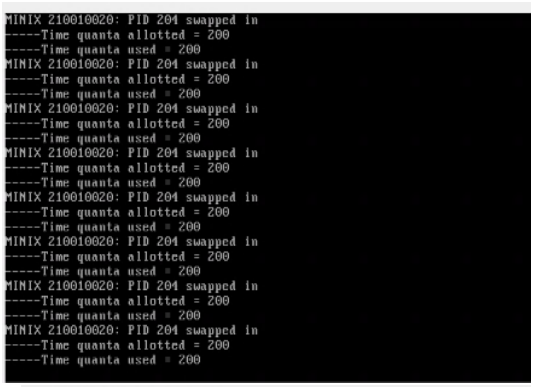
In do noquantum() function, we changed priority from lower priority to higher priority:

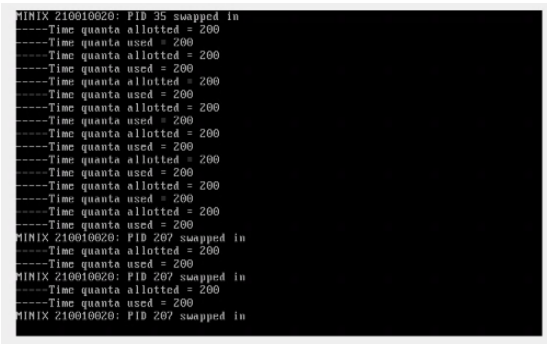
```
//rmp->priority += 1; /* lower priority */
 rmp->priority -= 1; /* changed to higher priority */
And in the balance queues() function, we commented out the line: //commented to prevent overflow of the priority queue
//rmp->priority -= 1; /* increase priority */
```

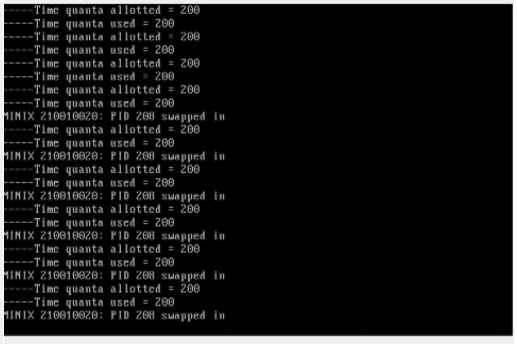
By reducing the priority by 1 of the scheduled process gives it higher priority when the scheduler tries to schedule the process again by selecting it from ready queue, the process with the highest priority.

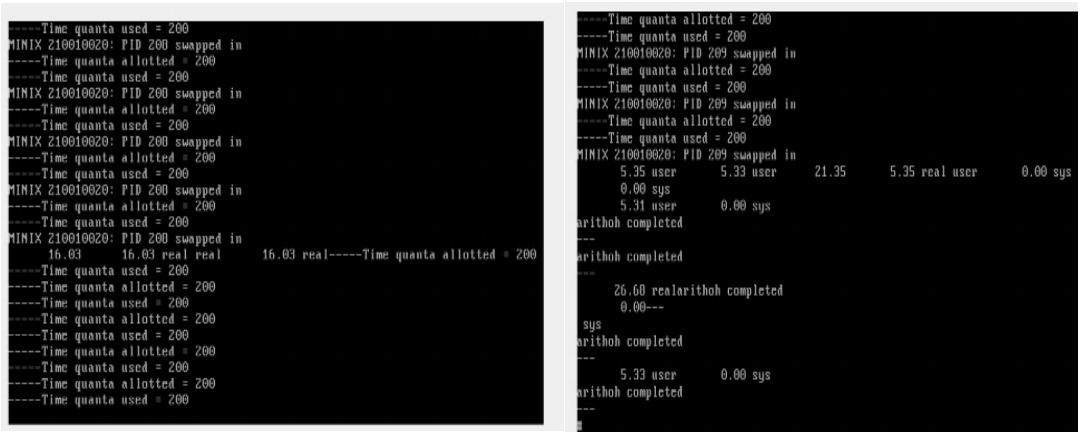
- Workload_mix1.sh:
  There are 5 arithoh.sh processes which run one after the other. Unless one is completed the other cannot start it's execution. The order of leaving of processes from the queue is same as order of arrival which justifies FIFO. Also each process completes it's execution time because it is cpu bound. First Process wth pid = 204 starts executing followed by process with pid 35 and pid 200 and lastly 209 they exit in the same order as they enter into the ready queue.



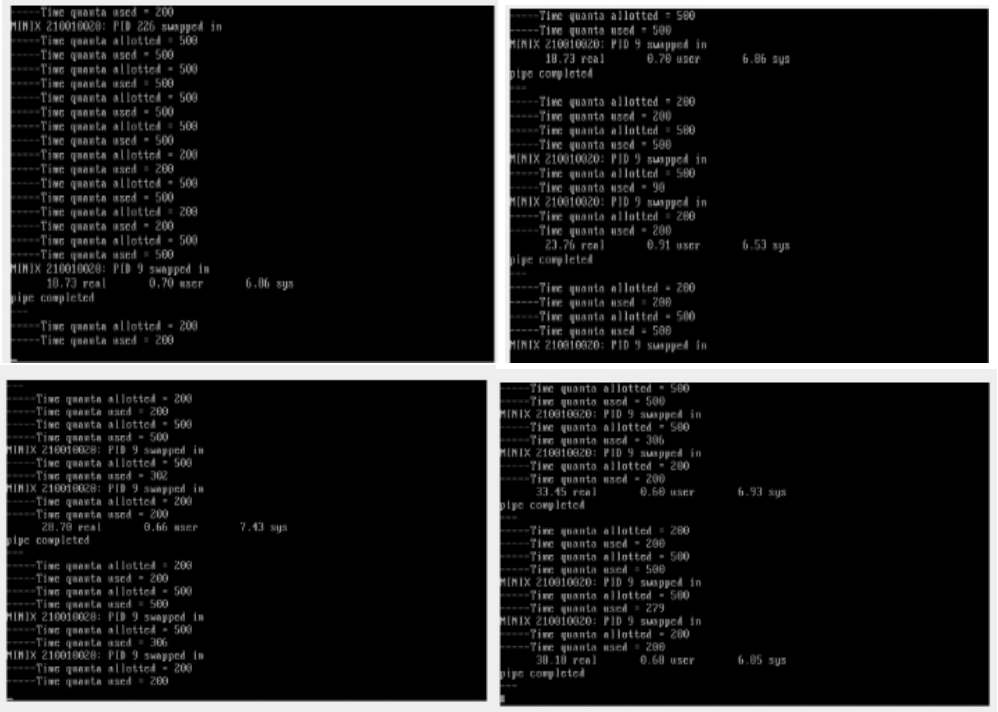






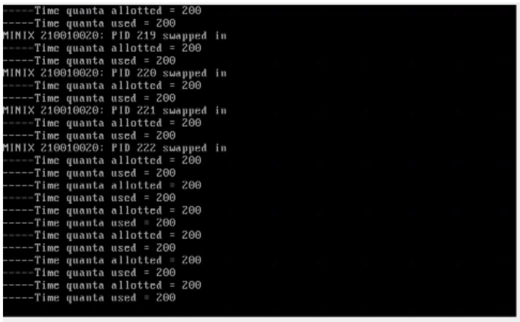


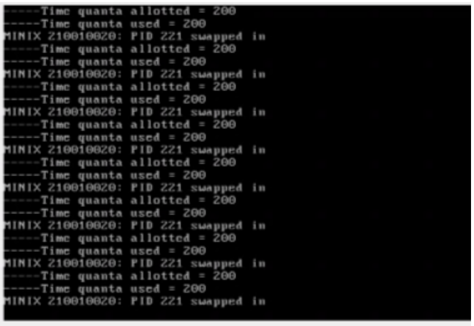
- Workload_mix4.sh:
  The 5 processes pipe.sh are all only I/O-intensive tasks. They get completed in the order they enter the queue. The time quanta allotted to I/O-intensive jobs is 500 ms. Each process uses its allocated time. Since these are IO bound processes so even in pseudo FIFO while the CPU is in block state for IO operation for a process another process gets scheduled. As seen in the screenshot below the process 226 starts executing it executes until some IO occurs once IO occurs new process pid=9 is scheduled and it starts execution this trend repeats. Until an IO operation occurs the process follow a FIFO .

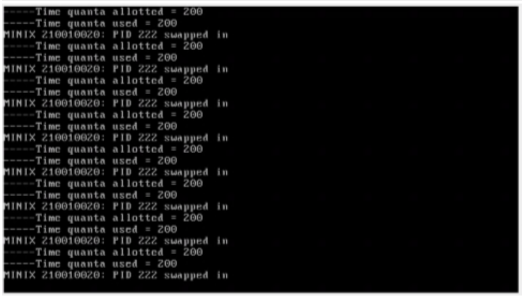


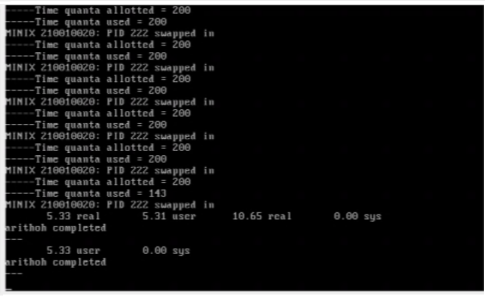
- Workload_mix2.sh

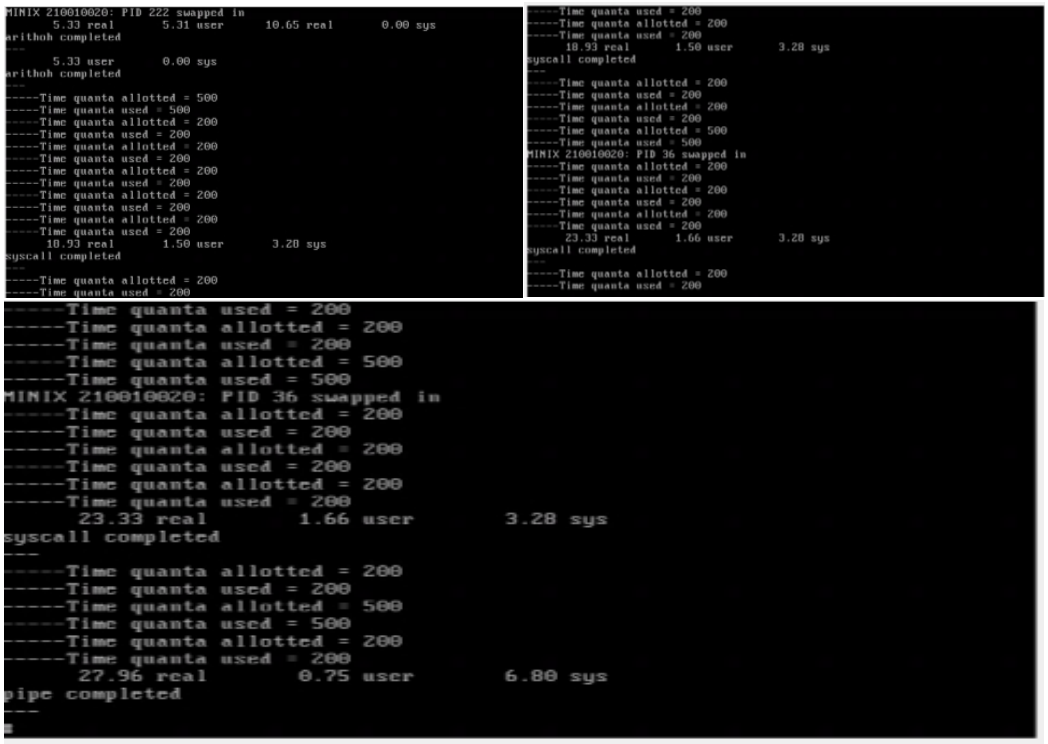
This is a mix of CPU and IO Bound processes the CPU bound executes for 200 ms and IO bound executes for 500ms but the first one to complete execution is the first one to arrive in the priority queue i.e arithoh.sh -> arithoh.sh->syscall.sh->syscall.sh->pipe.sh.
The execution order begins with syscall.sh(pid = 219) followed by syscall.sh (pid=220) then pipe.sh (pid=221) all the above execute for a cycle then enter blocked state the arithoh.h process with pid = 222 and pid=36 starts its execution and gets completed. When the IO bound process enter into ready state they are executed in the order of their arrival. So first both syscall gets completed then pipe.sh gets finally completes it’s execution.



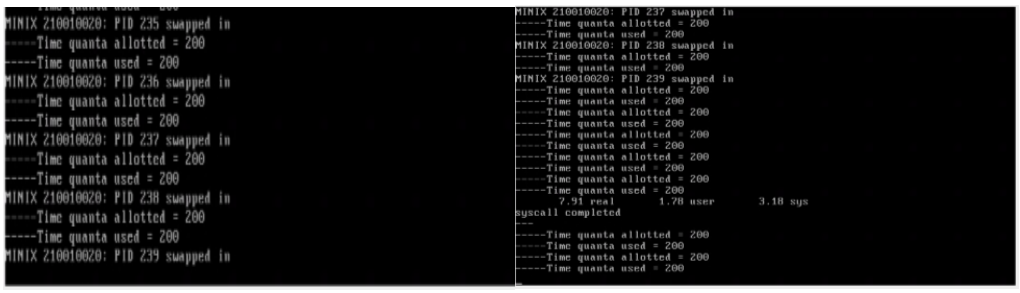


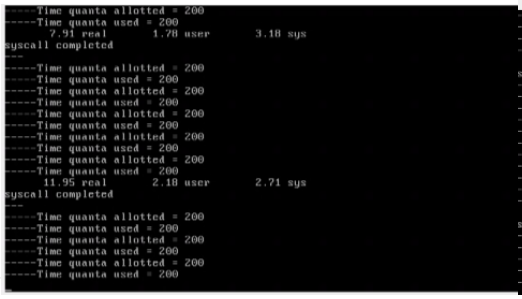






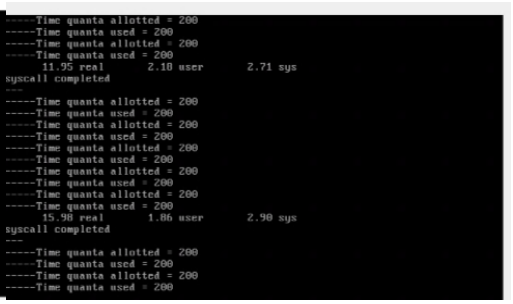
- Workload_mix3.sh All syscall.sh
  syscall.sh (semi-CPU-intensive) gets completed using time quanta of 200ms. In the beginning each process executes for CPU cycle before they enter the block state. Starting from process with pid = 234 to process with pid = 239 each run for one cycle in order of their arrival in the run queue. Once they enter the blocked state in the next cycle each run for two more cycles, then finally each of the process runs until it’s entire cpu time is used up. The process exits in FIFO manner.




-----Time quanta allotted = 200
-----Time quanta used = 200
7.91 real 1.78 user 3.18 sys
syscall completed

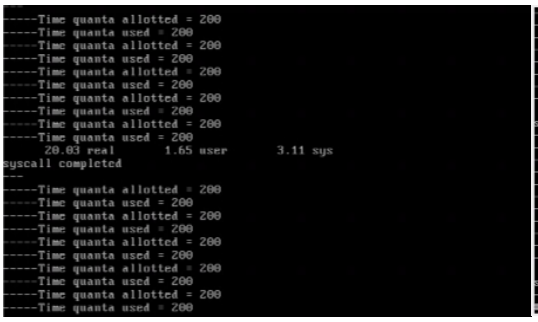
-----Time quanta allotted = 200
-----Time quanta used = 200
-----Time quanta allotted = 200
-----Time quanta used = 200
-----Time quanta allotted = 200
-----Time quanta used = 200
-----Time quanta allotted = 200
-----Time quanta used = 200
-----Time quanta allotted = 200
-----Time quanta used = 200
11.95 real 2.18 user 2.71 sys
syscall completed

-----Time quanta allotted = 200
-----Time quanta used = 200
-----Time quanta allotted = 200
-----Time quanta used = 200
-----Time quanta allotted = 200
-----Time quanta used = 200

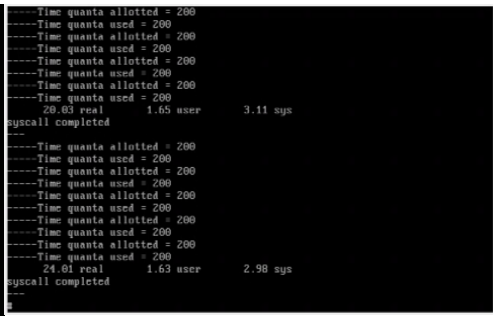

-----Time quanta allotted = 200
-----Time quanta used = 200
-----Time quanta allotted = 200
-----Time quanta used = 200
11.95 real 2.18 user 2.71 sys
syscall completed

-----Time quanta allotted = 200
-----Time quanta used = 200
-----Time quanta allotted = 200
-----Time quanta used = 200
-----Time quanta allotted = 200
-----Time quanta used = 200
-----Time quanta allotted = 200
-----Time quanta used = 200
15.98 real 1.86 user 2.90 sys
syscall completed

-----Time quanta allotted = 200
-----Time quanta used = 200
-----Time quanta allotted = 200
-----Time quanta used = 200


-----Time quanta allotted = 200
-----Time quanta used = 200
-----Time quanta allotted = 200
-----Time quanta used = 200
-----Time quanta allotted = 200
-----Time quanta used = 200
-----Time quanta allotted = 200
-----Time quanta used = 200
-----Time quanta allotted = 200
-----Time quanta used = 200
20.03 real 1.65 user 3.11 sys
syscall completed

-----Time quanta allotted = 200
-----Time quanta used = 200
-----Time quanta allotted = 200
-----Time quanta used = 200
-----Time quanta allotted = 200
-----Time quanta used = 200
-----Time quanta allotted = 200
-----Time quanta used = 200


-----Time quanta allotted = 200
-----Time quanta used = 200
-----Time quanta allotted = 200
-----Time quanta used = 200
-----Time quanta allotted = 200
-----Time quanta used = 200
-----Time quanta allotted = 200
-----Time quanta used = 200
20.03 real 1.65 user 3.11 sys
syscall completed

-----Time quanta allotted = 200
-----Time quanta used = 200
-----Time quanta allotted = 200
-----Time quanta used = 200
-----Time quanta allotted = 200
-----Time quanta used = 200
-----Time quanta allotted = 200
-----Time quanta used = 200
24.01 real 1.63 user 2.98 sys
syscall completed
