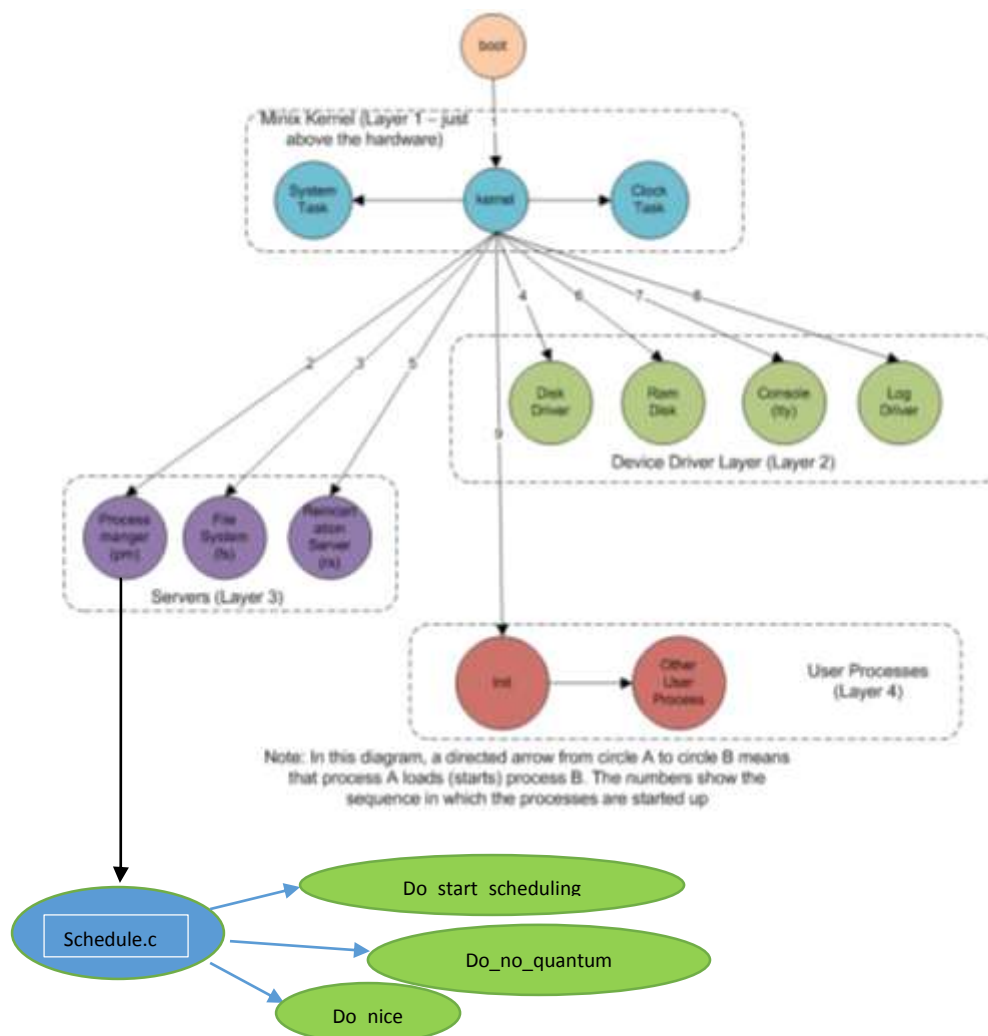


OS Group Assignment 2: Implementing Lottery and Multilevel feedback Queues schedulers for Minix

Aim: Implementing user level schedulers in Minix 3.2.1.

- Lottery Scheduler
- Multiple queue round robin scheduler

Flow Diagram



1. LOTTERY SCHEDULER:

Design:

Three more queues were added to the pre-existing 16 queues. Initially all the user processes will be placed in the 18th queue and they will be initialized with 5 tickets each. The priority of a given process can be modified with the function *nice*. A random number between 0 to *ntickets* (sum of tickets held by all the processes) is picked. Then, a process whose priority (number of tickets) are greater than or equal to the random number chosen is picked. This process is put in the 17th queue and the kernel schedules this process for one time quantum and after that it is placed back into the 18th queue. Similarly, another random number is picked and another process is chosen accordingly.

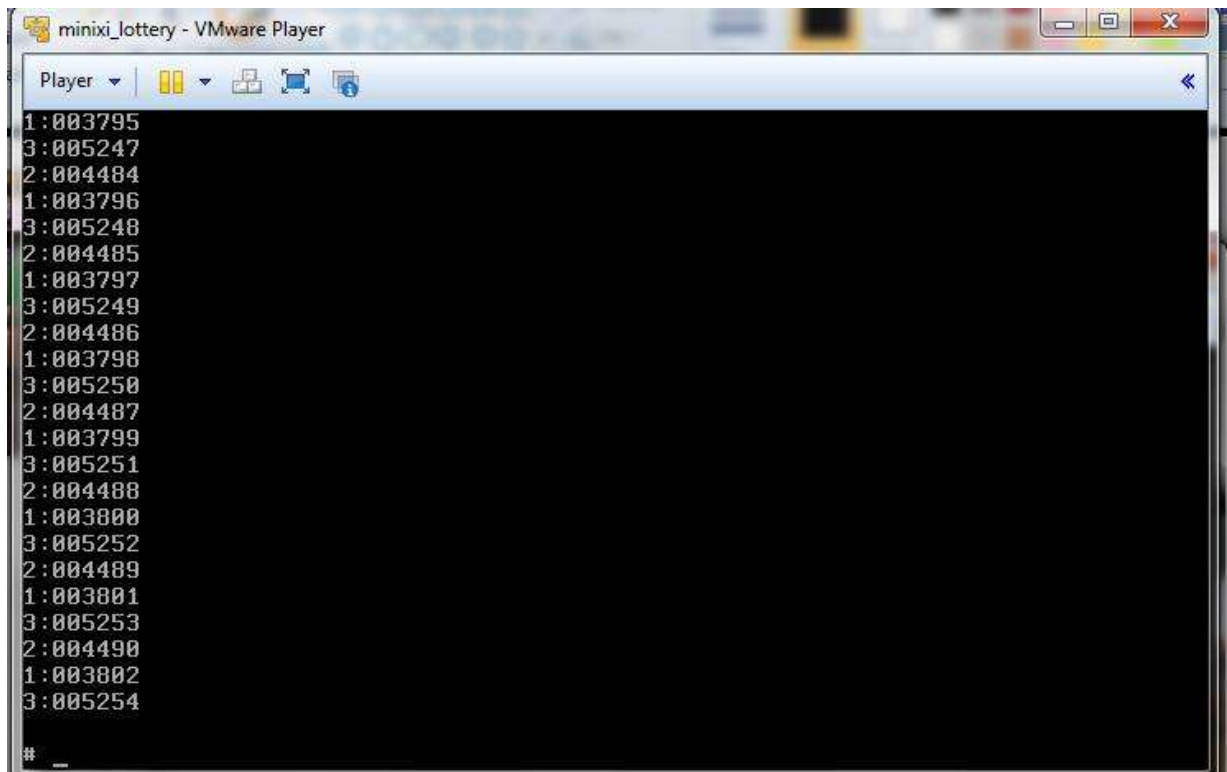
Implementation:

- The number of queues were increased from 16 to 19 in */usr/include/minix/config.h*
- A new data member, *p_tickets* was added to the *schedproc* structure in */usr/src/servers/sched/schedproc.h* to store the number of tickets held by the process.
- *p_tickets* was initialized to 5 for every new process and all the user processes were placed in the 17th queue instead of inheriting the priority from parent process. This was changed in *do_start_scheduling()* in */usr/src/servers/sched/schedproc.h*.
- A function *do_lottery()* that implements the lottery scheduler functionality was created in */usr/src/servers/sched/schedule.c*.
- The *nice* value was taken from the *nice* function and added to the *p_tickets* of the current process. This was changed in *do_nice()* in */usr/src/servers/sched/schedule.c*.
- The *do_lottery()* function is called after every call to *do_noquantum()*, *do_stop_scheduling()* and *do_nice()* functions.

Test cases:

Run the *longrun.c* provided, which in turn runs the *longrun1.c*, *longrun2.c* and *longrun3.c* and we get the following output which signifies the order in which the processes are running.

Screenshot:



```
minixi_lottery - VMware Player
Player
1:003795
3:005247
2:004484
1:003796
3:005248
2:004485
1:003797
3:005249
2:004486
1:003798
3:005250
2:004487
1:003799
3:005251
2:004488
1:003800
3:005252
2:004489
1:003801
3:005253
2:004490
1:003802
3:005254
#
```

2. MULTIPLE QUEUE ROUND ROBIN SCHEDULER:

Design:

Three more queues are added to the existing 16 queues. Initially all the user processes will be placed in 17th queue. Processes are then moved to the 18th queue after they have *completed* five time quanta, and then to the 19th queue after they have *completed* ten time quanta. Processes are thereafter moved back to 17th queue after completion of 20 time quanta.

Implementation:

- The number of queues were increased from 16 to 19 in the `/usr/include/minix/config.h`
- A new data member, `p_quanta`, was added to the `schedproc` structure in `/usr/src/servers/sched/schedproc.h` to store the time quanta held by the process.
- `p_quanta` is initialized to 0 for every new process and all the user processes are placed in the 17th queue instead of inheriting the priority from parent process. This is changed in `do_start_scheduling()` in `/usr/src/servers/sched/schedule.c`.
- After every call to `do_noquantum()`, `p_quanta` of the process is incremented and is checked as to which queue it belongs to as per the criteria mentioned above. It is then placed into the appropriate queue. These changes are done in `/usr/src/servers/sched/schedule.c`.

Test cases:

Run the testfile.c provided, and we get the following output which signifies the time quantum of a running process in the order of queue priority.

Screenshot:

```
Process 36701 consumed 5 and Priority is 16
Process 36701 consumed 10 and Priority is 17
Process 36701 consumed 20 and Priority is 18
Process 36701 consumed 5 and Priority is 16
Process 36701 consumed 10 and Priority is 17
Process 36701 consumed 20 and Priority is 18
Process 36701 consumed 5 and Priority is 16
Process 36701 consumed 10 and Priority is 17
Process 36701 consumed 20 and Priority is 18
Process 36701 consumed 5 and Priority is 16
Process 36701 consumed 10 and Priority is 17
Process 36701 consumed 20 and Priority is 18
Process 36701 consumed 5 and Priority is 16
Process 36701 consumed 10 and Priority is 17
Process 36701 consumed 20 and Priority is 18
Process 36701 consumed 5 and Priority is 16
Process 36701 consumed 10 and Priority is 17
Process 36701 consumed 20 and Priority is 18
Process 36701 consumed 5 and Priority is 16
Process 36701 consumed 10 and Priority is 17
Process 36701 consumed 20 and Priority is 18
Process 36701 consumed 5 and Priority is 16
^C
# Process 36627 consumed 5 and Priority is 16
—
```

Performance Analysis

