**Xv6 MLFQ implementation**

Implicit queue initialization

* Within the proc struct, a p->priority field is added to denote priority to a process
* As processes are added to the process table, the queue is instantiated with FCFS starting at the beginning of the ptable.

Visualizing implicit queues within ptable processes.

Ptable



|  |  |
| --- | --- |
| p->pid = 1 | p->priority = 2 |
| p->pid = 2 | p->priority = 3 |
| p->pid = 3 | p->priority = 2 |
| p->pid = 4 | p->priority = 1 |
| p->pid = 5 | p->priority = 1 |
| p->pid = 6 | p->priority = 3 |
| p->pid = 7 | p->priority = 2 |
| p->pid = 8 | p->priority = 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | 5 |  |  |  |



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 3 | 7 | 8 |  |



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | 6 |  |  |  |



Assume

p->state == RUNNABLE

First loop around ptable Second loop around ptable



|  |  |
| --- | --- |
| p->pid = 1 | p->priority = 2 |
| p->pid = 2 | p->priority = 3 |
| p->pid = 3 | p->priority = 2 |
| p->pid = 4 | p->priority = 1 |
| p->pid = 5 | p->priority = 1 |
| p->pid = 6 | p->priority = 3 |
| p->pid = 7 | p->priority = 2 |
| p->pid = 8 | p->priority = 2 |

|  |  |
| --- | --- |
| p->pid = 1 | p->priority = 2 |
| p->pid = 2 | p->priority = 3 |
| p->pid = 3 | p->priority = 2 |
| p->pid = 4 | p->priority = 1 |
| p->pid = 5 | p->priority = 1 |
| p->pid = 6 | p->priority = 3 |
| p->pid = 7 | p->priority = 2 |
| p->pid = 8 | p->priority = 2 |



**Priority algorithm within scheduler()**

Search through the ptable to find the first runnable process

CurrentP = firstRunnableP (=> **p**)

Search through the ptable again iterating with process pointer **p**

If the currentP process priority value is less than the **p** process priority value

Assign currentP to the new proc that **p** is pointing to in the ptable

Increment tick value

Condition if the process has consumed time allotted in queue

Demote if conditioned necessary

Promote if conditioned necessary

Run process currentP until system interrupt (10ms)

//currentP compares its priority to the process pointed to by **p**. If currentP->priority is GREATER than **p**->priority, then currentP should now point to the new process found in the ptable with a lower priority than itself.

//If the priority value of currentP matches the process pointed to by **p**, there is NO switch.

//This ensures that the lowest priority process that is reached first in the list will be executed (which follows FCFS).

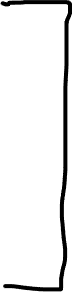
**State Diagram**

Newly created processes always push to Q1

Process spends 60ms in Q2

Still not Done?

Push to Q3



Q2



New process

Process spends 30ms in Q1

Still Not Done?

Push to Q2

Process spends 270ms in Q3

Still not done?

Push to Q1



Q3



Q1



**Code**

Proc.c

A picture containing text, screenshot, indoor

Description automatically generated

Graphical user interface, text, application, chat or text message

Description automatically generated

Proc.h

Graphical user interface, text, application

Description automatically generatedGraphical user interface, text, application

Description automatically generated

**Xv6 Console Output**

Command entered: spin 1000000 &; spin 2000000 &; spin 3000000;

**Console Output**

* quantum time consumption statements along with which jobs (with PID) are created and exited.
* Queue switch statements when job is demoted to a queue
* Priority boost when process (spin 300000) eventually reached 270 ms total CPU time and is priority boosted to Q1

A screenshot of a computer

Description automatically generated with medium confidenceSpin command output 1.1

Graphical user interface, text, application

Description automatically generatedSpin command output 1.2

Graphical user interface, text

Description automatically generatedSpin command output 1.3