Konrad Dagiel 120329501

Operating Systems II

Lab 1 report

Multilevel Feedback Queue Python simulation.

The theory behind the MLFQ algorithm and implementation are given in more detail in the comments of the python file attached. This file is a rundown of the input and output of the algorithm.

Input:

Text

Description automatically generated

Input explained:

We initialize 5 processes (see Process class in .py file)

p0 is an I/0 process with medium priority (3) and an exec time of 2000 units.

* This is to show what would happen if an IO process became unlocked and entered a lower priority than the usual highest priority

p1 is a regular process with high priority (0) and exec time of 60 units.

* This is to show how the algorithm handles high priority processes

p2 is a regular process with an extremely low priority (7) and very high execution time of 10000 units.

* This is to show how the algorithm handles the round-robin scheduling in the lowest priority queue

p3 is a regular process with high priority and high execution time

* This is to show a process starting at the top of the queue structure and being pushed down to the bottom, eventually being scheduled round robin with p2. This and p2 will be the only 2 remaining processes after a while, causing the voltage to decrease. (timeslice to increase)

p4 is an IO process which has the regular high priority of IO processes.

* This is to show how the scheduler handles IO processes

NOTE: for ease of demonstration, IO processes are assumed to complete whenever a queue has been executing for 320 units.

There is no logical reason for this value, other than simulating a “wait time” in a program that executes almost instantly.

This value gives the other processes a chance to go halfway down the structure before an IO interrupts them.

OUTPUT RUNDOWN:

Text

Description automatically generated

The 5 processes are added to their respective queues / the blocked list

----------------------------------------------------------------------------------------------

Text

Description automatically generated

Processes 1 and 3 execute in queue 0 for 10 seconds each, and are pushed to queue 1.

They then execute for 20 seconds in queue 1, where the quantum is 20sec

In queue 2, the quantum is 40 sec, so Process 1 finishes with 10 seconds to spare.

-----------------------------------------------------------------------------------------------



Process 3, which is now the only process in the Queue structure, gets the extra 10 seconds added to its execution time in queue 2 (40+10=50), therefore executes for 50 seconds in queue 2 before being pushed to queue 3, then queue 4.

------------------------------------------------------------------------------------------------

For the sake of simulation, this is where Process 3 it is interrupted by an IO process, process 4, which is placed in queue 0.

A picture containing text, scoreboard

Description automatically generated

Since Process 4 is the highest priority process, it interrupts the execution of Process 3. It goes down the queue structure by itself, until completing in queue 4.

At this point, the structure looks like this:

0:

1:

2:

3:

4:

5: process 3

6:

7: process 2

----------------------------------------------------------------------------------------

A black background with white text

Description automatically generated with low confidence

Now, IO process 0 enters the Queue structure in queue 3.

It executes for 80 secs in queue 3, as all other higher queues are empty.

Then is pushed to queue 4, executes, and is pushed to queue 5.

Process 3 then executes, then process 0, and both go to queue 6.

Both execute in Queue 6.

--------------------------------------------------------------------------------------------

Text

Description automatically generated

Both processes then go to queue 7, where process 2 has been waiting to execute.

Process 0 finishes in Queue 7.

Our power saving policy states that if there are two or less processes in queue 7, we increase timeslice by 20% and decrease voltage by 20%

Since our structure looks like this:

0:

…

7: process2 , process3

We engage power saving.

------------------------------------------------------------------------------------------------

Text

Description automatically generated

The new timeslice is 1536 (1280\*1.2=1536). Process 2 executes for 2016 secs because it takes the leftover cpu time of process 0 (480 sec).

(2016 = 1536+480)

Process 2 and Process 3 then execute round-robin in queue 7, until process 3 finishes.

Process 2 then finishes, meaning there are no more processes to execute.

Our power saving policy states that when there are no more processes to execute, the idle process takes over the CPU.

FULL OUTPUT:

Text

Description automatically generated