Operating Systems Programming Assignment 1 Report

Using Python

Table Of Contents

* Page 1 Table of Contents
* Page 2 Introduction
* Pages 3 & 4 General Flow Chart of logic
* Page 5 Results and discussion
* Page 6 Program Output Sample for FCFS
* Page 7 Program Output Sample for SJF
* Page 8 Program Output Sample for MLFQ
* Page 9 Program Final Results FCFS
* Page 10 Program Final Results SJF
* Page 11 Program Final Results MLFQ

Introduction

I was tasked to simulate 3 scheduling algorithms: First-Come-First-Served, Shortest-Job-First, and Multi-Level-Feedback-Queue which uses Round Robin Scheduling as well. I decided to write the simulation in python 3 because at the time of writing this I am taking a python course at FAU. Python has lots of useful applications as well as readability, so I hope the code is easy on the eyes. I started by creating a class called Process simply a constructor that creates a multitude of variables used to move the simulation forward. I then created another class called Process Life Cycle which is the class representation of the different states in the process life cycle. It contains the functions new, ready, waiting, running, and terminated to match the flow of the process life cycle. Enums were added to assign each process to a state, so p1 state is equal to new when it starts then is moved to ready, running, waiting, running, and terminated throughout the simulation. Inside the process life cycle class, a function called time tick was created to handle the processes each time tick, the main function will have a for loop that will act as a clock. Then once that started working, I created functions that would be their respective scheduling algorithms called by the ready function depending on which algorithms was to be simulated. In the main function you can enter the string abbreviation in all caps as a parameter of the process life cycle class to choose which algorithm you want to test. Finally, some functions that don’t pertain to a class were made to calculate the average waiting time, average response time, average turnaround time, and CPU utilization. You can see the flow of logic on the following page.

General Flow Chart(logic) of the simulation program

**ProcessLifeCycle**

-time

-processes

-terminated processes

-current process

-scheduling algorithm

-ready queue

**Process**

-State

-name

-data

-is CPU Burst

-started

**Running**

**Waiting**

**Time\_tick**

**Ready**

**Terminated**

**SJF**

**FCFS**

**MLFQ**

**RR**

Main

**Time\_Tick()**

Time += 1

For each process:

If process state = Ready:

Go to ready()

Else if process state = Waiting:

Go to waiting()

Else if process state = Running:

Go to running()

Reorganize process

Time units = 1000

P1 = Process([…], “P1”)

…

P2 = Process([…], “P2”)

OS = ProcessLifeStyle[p1, …, p8]

From 0 to time units + 1:

OS.time\_tick()

**Ready()**

If scheduling algorithm = “FCFS”:

FCFS(process)

Else if scheduling algorithm = “SJF”:

SJF(process)

Else if scheduling algorithm = “MLFQ”:

MLFQ(process)

Output Ttr, Tw, Tr

Diagram

Description automatically generated

Results and discussion

|  |  |  |  |
| --- | --- | --- | --- |
|  | **SJF** | **FCFS** | **MLFQ** |
| CPU Utilization | 82.81% | 84.58% | 89.64% |
| Avg Waiting time (Tw) | 134.38 | 186.25 | 152.0 |
| Avg Turnaround time (Ttr) | 470.63 | 522.5 | 488.25 |
| Avg Response time (Tr) | 27.13 | 24.38 | 16.13 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **SJF CPU utilization:** 82.81% | | | | **FCFS CPU utilization:**  84.58% | | | | **MLFQ CPU utilization:** 89.64% | | | |
|  | ***Tw*** | ***Ttr*** | ***Tr*** |  | ***Tw*** | ***Ttr*** | ***Tr*** |  | ***Tw*** | ***Ttr*** | ***Tr*** |  |
| P1 | 43 | 269 | 11 | 170 | 396 | 0 | 48 | 274 | 0 |
| P2 | 74 | 501 | 3 | 168 | 595 | 5 | 109 | 536 | 5 |
| P3 | 277 | 669 | 16 | 170 | 562 | 9 | 226 | 618 | 9 |
| P4 | 51 | 535 | 0 | 171 | 655 | 17 | 17 | 501 | 14 |
| P5 | 238 | 547 | 109 | 211 | 520 | 20 | 278 | 587 | 17 |
| P6 | 122 | 337 | 24 | 227 | 442 | 36 | 186 | 401 | 22 |
| P7 | 150 | 478 | 47 | 188 | 516 | 47 | 183 | 511 | 27 |
| P8 | 120 | 429 | 7 | 185 | 494 | 61 | 169 | 478 | 35 |
| Avg | ***134.38*** | ***470.63*** | ***27.13*** | ***186.25*** | ***522.5*** | ***24.375*** | ***152.0*** | ***488.25*** | ***16.13*** |

As I expected SJF has the lowest waiting time, turnaround time, and CPU utilization of the three while also having the longest turnaround time because it must keep pushing the largest burst away. FCFS is the worst scheduling algorithm of the three because it has the highest waiting time and turnaround time, while also having the 2nd highest response time and an average CPU utilization. MLFQ sits right in the middle between SJF and FCFS but utilizes the CPU much more than the other two.

Program Output Sample

Text

Description automatically generated with medium confidence**A picture containing graphical user interface

Description automatically generatedText

Description automatically generatedFCFS**

Program Output Sample

**SJF**

A picture containing text

Description automatically generatedText

Description automatically generatedA picture containing text

Description automatically generated

Program Output Sample

**MLFQ**

A picture containing text

Description automatically generatedA picture containing text

Description automatically generatedText

Description automatically generated

Program Output of the results

Text

Description automatically generated**FCFS**

Program Output of the results

Text

Description automatically generated**SJF**

Program Output of the results

Text

Description automatically generated**MLFQ**