**Graphical user interface, application

Description automatically generated**

**COMP2035 Coursework Report**

**Process Scheduling Algorithms**

22nd November 2021

Group 19 Chew Language

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Student ID** | **Usernames** | **Course Name** |
| Tan Zhun Xian | 20313854 | hcyzt1 |  |
| Chong Hao Wei | 20194465 | hfyhc3 |  |
| Leong Chang Yung | 20307078 | hcycl3 | CS with AI |

**Table of Contents**

Introduction ................................................................................................................................................ 3

User Interface ……..................................................................................................................................... 4

Coding Technique ...................................................................................................................................... 5

Application of different scheduling algorithm under different workload .................................................. 6

Influence of different time quantum in Round Robin (RR) Scheduling algorithm ............................…... 6

Explanation on how scheduler handles task in program ........................................................................... 6

Conclusion ................................................................................................................................................. 7

Appendix ................................................................................................................................................... 10

Marking Scheme ....................................................................................................................................... 10

**Introduction**

Shortest Job First (SJF) is a CPU scheduling algorithm that will prioritize the process with the least amount of job time and is non preemptive by its nature. In most cases, this algorithm output the least amount of waiting time by average. But due to the nature of this algorithm, the long processes might not be able to be processed if there is too much shorter process in queue, resulting in the starvation of that process.

Meanwhile, Round Robin (RR) will give all the process the same priority. This is done through defining a time quantum and executing all the processes for a that specific time quantum before jumping to the next process in its queue while putting the previous process to the end of the circular queue. Through this scheduling algorithm, longer process won’t be neglected until leading to starvation.

Our program compares both algorithms and its counterparts to determine which one is the most suitable for the workload given. The program does this by prompting the user to enter the dataset, including time quantum and time delay, then predict which of the 6 algorithm (Shortest Job First (SJF) Pre-emptive Version, Shortest Job First (SJF) Non-Pre-emptive Version, Round Robin (RR) with Overhead, Round Robin (RR) without Overhead, Round Robin (RR) with Overhead without Arrival Time, Round Robin (RR) without Overhead without Arrival Time) is the most effective for the given condition. After that, it will compare all the algorithms based on the given info for the lowest average wait time and turnaround time, displaying the results and the process of execution for each algorithm. There will be a maximum of 2 algorithms that will be deemed as “most effective”, which is the algorithm with the lowest average wait time and the algorithm with the lowest average turnaround time.

**User Interface**

At launch of the program, the user will be requested to choose between “Input jobs manually” and “Input jobs through .txt files”, the user may choose one of those options by typing “1” or “2” into the program (diagram 1.1), similar input method is later repeated through the program.

If user choses “Input jobs manually” (diagram 1.2), the program will prompt the user to input the details including: “Job name”, “Arrival Time” and “Burst Time”, the program will automatically ask if user wanted to input another job until user wish to proceed, which in this case user need to input “q” after they finish the previous job input.

However, if user chooses to “Input jobs through .txt file” (diagram 1.3), the program will ask the user in input the file name to be read. If successfully read the file, the program will proceed to the next step. Or else, the program will display error and terminates.

After getting the input for the jobs, no matter through manual input or file read, the program will then prompt the user to input the “time quantum” and “time delay”, which will be used when testing through the different algorithms.

With all the required data being entered by user, the program will ask the user to predict which of the scheduling algorithm will be the most suitable for the workload given (diagram 1.4), user will be asked to choose between one of the following options: Shortest Job First (SJF) Preemptive Version, Shortest Job First (SJF) Non-Preemptive Version, Round Robin (RR) with Overhead, Round Robin (RR) without Overhead, Round Robin (RR) with Overhead without Arrival Time, and Round Robin (RR) without Overhead without Arrival Time.

After user input their prediction for the most efficient algorithm, the program will go through each of the individual algorithm and compare their “average wait time” and “average turnaround time”, the algorithm that is lowest in one of these 2 categories will be deemed as the “most efficient algorithm” (diagram 1.5). Which then the program will compare that to the user input to check if they predicted the correct outcome. Even though the output of the program will slightly differ depends on whether user selected the correct algorithm prediction, but the most effective algorithm in terms of “lowest average wait time” and “lowest average turnaround time” will be displayed along with all the process and results from each individual algorithm (diagram 1.6).

**Coding Technique**

The program uses structure to declare all the variables for a Job, which is then referred from each part of the codes through pointers. The struct names “Job” (diagram 2.1) consists of all the info for one job, including: job name, arrival time, burst time, burst time left, completion time, turnaround time, waiting time, entry time and all the flags trigger for some of the associated process.

All the Job sorting in this program was done through bubble sort (Diagram 2.2), which compares the Nth element and (N+1)th element, and swap both elements if a certain condition is met. To accomplish that, an empty pointer is used as a temporary placeholder for one of those elements during the swapping process so that the data is not lost.

For algorithms with arrival time, the program declares a variable called “systemClock” (diagram 2.3) to simulate the time where the job was added into the queue. The variable will increase by 1 at the end of each loop and if the process found a job where arrival time equals to “systemClock” variable, the job will then be added into the queue, this process is rinse and repeated until all the process ended, which then the break flag will be set to 1, stopping the process from looping. While for those without arrival time, arrival time was set to 0 instead and the sequence of the queue follow the sequence of the job inputs instead.

For Shortest Job First (SJF) non-preemptive version (diagram 2.4), the queue will be sorted based on “burst time left” at the start of each queue to make sure the job with the least burst time left is always first in the queue. In its preemptive counterpart (diagram 2.5) however, the core is still the same: keeping the job with the least burst time left first, but this time the queue will be sorted per tick, stopping any ongoing process if a new process with a shorter burst time left was added into the queue.

In Round Robin algorithm, all the jobs are put into a circular queue (diagram 2.6), which the job on top will be moved to the bottom of the queue after being executed for the number of time quantum, so all the jobs in queue will be given the same priority of execution.

Round Robin with overhead will add a delay time (diagram 2.7) by the end of the process each time the program finish executing a job by its time quantum. This is accomplished by having a delay flag set to TRUE until the program finishes the delay time so that it’s added into the “systemClock” variable before setting back to FALSE, which allows the program to continue proceeding with the job next in queue.

**Application of different scheduling algorithm under different workload**

By comparing these two CPU scheduling algorithms, SJF can provide a shorter waiting time average in that given workload since it focuses on one task at a time instead of trying to execute multiple tasks at once. However, if there’s a job with a long burst time, given the way where how SJF schedule the task the job might be delayed indefinitely when multiple jobs with shorter burst time was continuously added into the queue, this may lead to starvation if that job needs to be executed within a certain time.

Hence, Shortest Job First are more suitable in the scenario where there’s no requirement for any processes to be executed in each time, while in case of any of the task can only be completed if a certain process can be executed successful in each time, Round Robin will be a better solution when compared with Shortest Job First.

**Influence of different time quantum in Round Robin (RR) Scheduling algorithm**

In Round Robin (RR) Scheduling Algorithm, time quantum plays a huge role as it determines how long will each process be processed before moving on to the next in queue. To understand the influence of different time quantum on this scheduling algorithm, we executed the same sets of jobs with different time quantum set on Round Robin (RR) with overhead. The delay time was set to 1 second across all the tests run while the time quantum was set to: 1,15 and 40. By observing the results (Diagram 3.1, 3.2, 3.3), we may conclude that even though shorter time quantum might give a lower average waiting time to the Jobs, it also increases the average Turnaround time overall given that a delay time is present when switching jobs. However, if the time quantum was too long, the algorithm will just execute the jobs on a “first come, first serve” basis as the time quantum was enough for the algorithm to run through the entire burst time of the current job before switching to another, leaving the reason of having a scheduling algorithm behind.

**Explanation on how scheduler handles task in program**

In this program, the scheduler cycles through loops where the variable “mode” (diagram 4.1) increase by one each time by the end of each loop starting from 0 until it is greater than or equal to 6. Each mode features a different sorting algorithm which will be run through all the jobs input from the user. By using the “mode” variable as an indicator, the program will be able to identify which process will be carried out by each individual algorithm (diagram 4.2). For example, when mode equals to 1 (Shortest Job First non-preemptive), the program will go through all the process and execute those that are either shared among all the algorithms or those that requires “if (mode ==1)”.

All the algorithms start off with a system clock (diagram 4.3), which records the total runtime for the whole algorithm. The jobs will first be stored in a queue array by copying the respective job in order (either based on arrival time or the sequence of job input) into the queue using a “temp” pointer, which then was filled in into an empty node in the array. Which then the scheduler will proceed through the queue based on the mode that the scheduler is currently looping on.

When the burst time left of a job reaches 0 (diagram 4.4), the job will be removed from the queue and the scheduler will carry on with the next job in the queue array while resetting some of the variables, until there’s no job left. At the same time, the completion time, turnaround time and waiting time will be recorded individually, which will later sort based on completion time and saved in the “save” array.

At the end of each algorithm, all the respective info for each individual jobs will be printed to the user and the average wait time and turnaround time will be calculated (diagram 4.5).

When the scheduler loops through all the available modes in the program, the average turnaround time and wait time of each individual algorithm will be compared to determine which is the most effective algorithm for the workload (diagram 4.6).

A picture containing shape

Description automatically generated

Diagram 4.1 Program cycles through all modes

A picture containing graphical user interface

Description automatically generated

Diagram 4.2 Process that only executes when mode equals to 4 or 5

Text

Description automatically generated

Diagram 4.3 The System Clock loop

Graphical user interface, text, application, email

Description automatically generated

Diagram 4.4 Process when a Job’s burst time left reaches 0

Text, application

Description automatically generated

Diagram 4.5 Printing and calculation when all jobs done

Text

Description automatically generated

Diagram 4.6 Comparing for lowest average wait time and turnaround time

**Conclusion**

In conclusion, scheduling algorithms allows CPU to handle process more efficiently. Without the help of scheduling algorithm, time and resources will be wasted in waiting I/O while CPU is free to handle another process during that time.

There are a lot of scheduling algorithm, including both Shortest Job First (SJF) and Round Robin (RR) where both have their own advantages and disadvantages. There’s no algorithm among these two that is superior to another as both algorithms are the most efficient in their optimal workload environment.

By utilizing the combinations of different types of algorithms based on the most effective scenario, CPU can execute all the processes efficiently without wasting too many time and resources.

Text

Description automatically generated

Diagram 1.1 User interface (Startup)

Text

Description automatically generated

Diagram 1.2 User Interface (Manual Input)

Text

Description automatically generated

Diagram 1.3 User Interface (File read error)

A computer screen capture

Description automatically generated with medium confidence

Diagram 1.4 User Interface (User prediction)

Text

Description automatically generated

Diagram 1.5 User Interface (Result output)

A picture containing graphical user interface

Description automatically generated

Diagram 1.6 User Interface (algorithm process and result)

Text

Description automatically generated

Diagram 2.1 Struct (Job)

Text

Description automatically generated

Diagram 2.2 Bubble sort to sort the queue based on burst time left

Graphical user interface, text, application

Description automatically generated

Diagram 2.3 System clock that adds job into the queue based on their arrival time

Text

Description automatically generated

Diagram 2.4 Sorting algorithm for Shortest Job First (SJF) non-preemptive version

Text, application

Description automatically generated

Diagram 2.5 Sorting algorithm for Shortest Job First (SJF) preemptive version

Graphical user interface, text, application

Description automatically generated

Diagram 2.6 Sorting algorithm for Round Robin (RR)

A picture containing text

Description automatically generated

Diagram 2.7 Process for delay when delay flag was set to TRUE

A picture containing calendar

Description automatically generated

Diagram 3.1 Round Robin (RR) with overhead output with time quantum of 1

A picture containing calendar

Description automatically generated

Diagram 3.2 Round Robin (RR) with overhead output with time quantum of 15

Calendar

Description automatically generated with low confidence

Diagram 3.3 Round Robin (RR) with overhead output with time quantum of 30

**MARKING SCHEME**

|  |  |  |  |
| --- | --- | --- | --- |
| Student Name &  Stud-Id: | (1) Tan Zhun Xian 20313854  (2) Chong Hao Wei 20194465  (3) Leong Chang Yung 20307078 | | |
| Group No: | 19 | | |
| Module Title & Code: | **Operating Systems and Concurrency (G52OSC)** | | |
| Attributes | | Marks Allotted | Marks Awarded |
| Part A: Software Part: (30%) | |  |  |
|  | |  |  |
|  | |  |  |
|  | |  |  |
|  | |  |  |
|  | |  |  |
|  | |  |  |
|  | |  |  |
| 2. Report Part: (20%) | |  |  |
|  | |  |  |
|  | |  |  |
|  | |  |  |
|  | |  |  |
|  | |  |  |
|  | |  |  |
|  | |  |  |
|  | |  |  |
|  | |  |  |
|  | |  |  |
|  | |  |  |
| **Total** | | 50 |  |

Note: Include this marking scheme page in the last page of your report.