SYSC 4001 Assignment 2  
Student Name: Oritsemeyiwa Jordan Temile   
Student Number: 101196274  
Date: November 11, 2022

**Part 1**

1. Firstly, the process may have executed for its allotted time quantum, in which it is moved back from the running to the ready state. The kernel makes a context switch to run the next process for the allotted time. When this occurs, the kernel saves all contents of the previous process and loads the contents of the new process to be run. Secondly, the process may have finished execution, so it will be moved from the running to the terminated state. The process will make a system call (exit()) to the OS to free all the memory it has used. Finally, for a pre-emptive priority-based scheduler, a process of higher priority may have entered the ready queue, or an I/O call has been made by the process. In this event the process in the running state will be moved to the ready state, and the kernel will forward the request to the device if an I/O call was made.
2. User-level threads are threads that the OS is unaware of. They are faster to create than kernel-level threads and are easier to manage, this feature proves beneficial over kernel-level threads in a time-shared OS. This is because context switching occurs frequently if the kernel is time shared and context switching between kernel-level threads has high overhead, but not overhead between user-level threads.  
   Kernel-level threads are managed by the OS and scheduled by the kernel by allocating time slices, this feature makes it better than user-level threads when there are multiple processes with I/O bound threads. This is because the scheduling algorithm will ensure a thread will always be running by avoid scheduling processes which consist of entirely idle threads. Also, these I/O bound threads will benefit from the additional time slices.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FCFS | | | | |
| Process | Arrival Time | Burst Time | Exit Time | Turnaround Time (Exit – Arrival Time) |
| A | 0 | 22 | 22 | 22 |
| B | 9 | 11 | 33 | 24 |
| C | 12 | 12 | 45 | 33 |
| D | 13 | 11 | 56 | 43 |
| E | 17 | 14 | 70 | 53 |
| Average | | | | 175/5=35 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E |
| 0 | 22 | 33 | 45 | 56 | 70 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| RR | | | | |
| Process | Arrival Time | Burst Time | Exit Time | Turnaround Time (Exit – Arrival Time) |
| A | 0 | 22 | 65 | 65 |
| B | 9 | 11 | 50 | 41 |
| C | 12 | 12 | 62 | 50 |
| D | 13 | 11 | 64 | 51 |
| E | 17 | 14 | 70 | 53 |
| Average | | | | 260/5=52 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | A | A | A | B | A | C | B | D | A | E | C | B | D | A | E | C | B | D |
| 0 | 3 | 6 | 9 | 12 | 15 | 18 | 21 | 24 | 27 | 30 | 33 | 36 | 39 | 42 | 45 | 48 | 50 | 53 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| A | E | C | D | A | E | F |
| 56 | 59 | 62 | 64 | 65 | 68 | 70 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Multiple queues with feedback | | | | |
| Process | Arrival Time | Burst Time | Exit Time | Turnaround Time (Exit – Arrival Time) |
| A | 0 | 22 | 50 | 41 |
| B | 9 | 11 | 57 | 44 |
| C | 12 | 12 | 66 | 54 |
| D | 13 | 11 | 69 | 52 |
| E | 17 | 14 | 70 | 70 |
| Average | | | | 283/5=52.2 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | A | A | A | B | B | C | D | C | D | E | D | E | A | B | C | D | E | A |
| 0 | 1 | 3 | 9 | 10 | 12 | 13 | 14 | 16 | 17 | 18 | 20 | 22 | 26 | 30 | 34 | 38 | 42 | 46 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| B | C | D | E | A | C | E | A |
| 50 | 54 | 57 | 61 | 65 | 66 | 69 | 70 |



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| FCFS | | | | | | | | | | | | | | | | | | | | | | |
|  | A |  | A |  | A |  | A | |  | A | B | A | B | A | C | B | D | A | C | B | D |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|  | | | | | | | | | | | | | | | | | | | | | | |
| E | A | C | B | D | E | A | C | | B | D | E | A | C | B | D | E | A | C | B | D | E |
| 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 |
|  | | | | | | | | | | | | | | | | | | | | | | |
| A | C | B | D | E | A | C | B | | D | E | A | C | B | D | E | A | C | D | E | A | C |
| 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 |
|  | | | | | | | | | | | | | | | | | | | | | | |
| D | E | A | C | E | A | E | A | | E | A | E | A | E |  | E |
| 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | | 71 | 72 | 73 | 74 | 75 | 76 | 77 |
| Average | | | | | | | | 283/5=56.6 | | | | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| RR | | | | | | | | | | | | | | | | | | | | | | |
| A |  | A |  | A |  | A |  | | A | B | A | B | A | C | B | D | A | C | B | D | P5 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 |
|  | | | | | | | | | | | | | | | | | | | | | | |
| A | C | B | D | E | A | C | B | | D | E | A | C | B | D | E | A | C | B | D | E | A |
| 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 |
|  | | | | | | | | | | | | | | | | | | | | | | |
| C | B | D | E | A | C | B | D | | E | A | C | B | D | E | A | C | D | E | A | C | D |
| 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |
|  | | | | | | | | | | | | | | | | | | | | | | |
| E | A | C | E | A | E | A | E | | A | E | A | E |  | E |
| 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | | 72 | 73 | 74 | 75 | 76 | 77 |
| Average | | | | | | | | 956/5=56.6 | | | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Multiple queues with feedback | | | | | | | | | | | | | | | | | | | | | | | | |
| P1 |  | P1 |  | P1 |  | P1 |  | | P1 | P2 | P1 | P2 | | P3 | | P4 | P1 | P2 | P3 | P5 | P4 | P1 | P2 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | | 9 | 10 | 11 | 12 | | 13 | | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 |
|  | | | | | | | | | | | | | | | | | | | | | | | | |
| P3 | P5 | P4 | P1 | P2 | P3 | P5 | P4 | | P1 | P2 | P3 | P5 | | P4 | | P1 | P2 | P3 | P5 | P4 | P1 | P2 | P3 |
| 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | | 30 | 31 | 32 | 33 | | 34 | | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 |
|  | | | | | | | | | | | | | | | | | | | | | | | | |
| P5 | P4 | P1 | P2 | P3 | P5 | P4 | P1 | | P2 | P3 | P5 | P4 | | P1 | | P2 | P3 | P5 | P4 | P1 | P3 | P5 | P4 |
| 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | | 51 | 52 | 53 | 54 | | 55 | | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |
|  | | | | | | | | | | | | | | | | | | | | | | | | |
| P1 | P3 | P5 | P1 | P5 | P1 | P5 | P1 | | P5 | P1 |  | P1 |
| 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | | 72 | 73 | 74 | 75 | |
| Average | | | | | | | | 280/5=56 | | | | | | |

1. **FCFS** algorithm allocates processes to the CPU based on their arrival time. This algorithm discriminates in favour of long-term processes because the waiting time is less if long processes come after a short time.  
   **RR** algorithm allocates all processes the same time quantum, so long-term processes will be in the ready queue longer and will wait for a long time to finish executing unlike short-term processes which terminates earlier. Hence, this algorithm discriminates against long-term processes as they will require more time to terminate.  
   **Multilevel feedback queues** discriminates against long-term processes because short-term processes are given a higher priority, while long processes are moved down the waiting queue.

|  |  |
| --- | --- |
| First Fit | |
| Job No. | Partition |
| 1 | 205K |
| 2 | 180K |
| 3 | No space |
| 4 | 102K |

|  |  |
| --- | --- |
| Best Fit | |
| Job No. | Partition |
| 1 | 125K |
| 2 | 150K |
| 3 | 205K |
| 4 | 91K |

|  |  |
| --- | --- |
| Worst Fit | |
| Job No. | Partition |
| 1 | 205K |
| 2 | 180K |
| 3 | No space |
| 4 | 150K |

**Part 2**

The discussion involves the comparison of the average turnaround and wait time of the First Come First Serve (FCFS), Pre-emptive and Round Robin (RR) algorithms. The comparison evaluated later in this discussion will determine which algorithm is best suited for the type of processes to be executed.

The First Come First Serve (FCFS) algorithm executes the process that enters the ready queue first, until completion or makes an I/O call. This means the algorithm prioritizes the arrival time of processes, the process that arrives first gets a higher priority than others. The Pre-emptive algorithm is like the FCFS algorithm; however, each process has a priority which determines the process to first execute. Finally, for the Round Robin algorithm, each process is given a portion of the CPU time called a time quantum to limit the execution time. After this time expires, the process is interrupted and makes a context switch to the ready state, but if the burst time is less than the time quantum the process runs until completion.

The table below shows a set of processes along with their arrival, execution, and I/O time and priority to be run using the scheduling algorithms discussed, along with the result after each execution. The following data is extracted from test\_case\_2.csv.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PID** | **Arrival Time** | **Total CPU Time** | **I/O Frequency** | **I/O Duration** | **Priority** |
| 1001 | 0 | 22 | 1 | 1 | 2 |
| 1002 | 9 | 11 | 1 | 1 | 5 |
| 1003 | 12 | 12 | 1 | 1 | 1 |
| 1004 | 13 | 11 | 1 | 1 | 2 |
| 1005 | 17 | 14 | 1 | 1 | 3 |

FCFS: Average waiting time: 42, Average turnaround time: 56

Priority: Average waiting time: 32, Average turnaround time: 46

RR (time quantum=10): Average waiting time: 42, Average turnaround time: 56

The RR algorithm ensures that all process shares the CPU equally, but as seen above the average wait time is longer with other scheduling algorithms. Also, the kernel performs context switch each time a process quantum expires, which results in a higher turnaround time depending on the value of the time of the quantum. But using larger time quantum to reduce context switch may result in the process running continuously until it terminates or makes an I/O call. This eliminates the benefit of having each process share the CPU, as now the process will run until completion before the next process utilizes the CPU, which is like the FCFS algorithm. Therefore, RR is ideal for a Time-Sharing OS with a small-time quantum.

In the case of the Priority Pre-emptive algorithm, the average wait time is 32 and turnaround time is 46. This algorithm can be seen as fast in comparison to the others, but because high priority process run to completion lower priority process may experience starvation if the burst time are long. Unlike the use case above, if the burst time for the process is long, the CPU clock will be longer resulting in a longer average turnaround time. So, the Pre-emptive algorithm is the optimal scheduling algorithm to minimize average waiting time for a set of processes, also will have a shorter turnaround time of less CPU intensive process have a higher priority.

The FCFS algorithm favours CPU bound processes, as a process with a long burst time will execute to completion unless it makes an IO call. But because of this feature, FCFS discriminates against processes with a short burst time as the waiting time is long. This effect results in lower CPU and device utilization. AS a result of these features, FCFS have high waiting and turnaround processes and will not be ideal for interactive and time sensitive programs.

To conclude, the FCFS is better for processes with a small burst time, and the Pre-emptive algorithm is best for processes that require an immediate response. Finally, the RR algorithm is better for processes that must execute in a short time, as the average waiting time can be varied with the time quantum.

For the second part of part 2 I implemented a paganing like memory to simulate, I will present my findings and discuss on it. I compared my test cases against two different memory partitions and an example test case is below.

Memory 1: 500, 250, 150, 100

Memory 2: 300, 300, 350, 50

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PID** | **Arrival Time** | **Total CPU Time** | **I/O Frequency** | **I/O Duration** | **Memory Required** |
| 1001 | 0 | 30 | 10 | 6 | 250 |
| 1002 | 1 | 25 | 20 | 7 | 10 |
| 1003 | 2 | 10 | 8 | 5 | 320 |
| 1004 | 3 | 32 | 10 | 7 | 200 |
| 1005 | 20 | 20 | 4 | 4 | 12 |
| 1006 | 10 | 12 | 6 | 7 | 50 |
| 1007 | 20 | 15 | 6 | 4 | 300 |
| 1008 | 10 | 13 | 5 | 2 | 10 |

From the test case I had these results.

Memory 1: Average waiting time: 19, Average turnaround time: 39

Memory 2: Average waiting time: 14 Average turnaround time: 34

From the results Memory 2 had a faster total execution time to complete all process because the partition size was evenly split between the 4 partitions in this memory, unlike memory 1 where only 1 memory got a large size. This feature ensures large processes can run simultaneously along other large processes. But the disadvantage is there is internal fragmentations, especially when the process only required small amount of memory, which makes the average turnaround time longer. However, because the page sizes are equal, swapping processes are easier.