Salma Rashed, Taya Wongsalong
CS433 Operating Systems
Dr. Xiaoyu Zhang
November 3rd, 2022

Programming Assignment 3 Report

Submitted Files:

- **PCB.h** :A process control block (PCB) Process control block(PCB) is a data structure representing a process in the system.It contains the following fields: process ID (PID) , process name ,burst time, priority, burst time, arrival time,waiting and turnaround time
- **Scheduler_fcfs.h:**header file that declare a scheduler class that implements the First come fist serve (FCFS)scheduling algorithm.
- **Scheduler_fcfs.cpp:** A source file that contains the functions that has scheduler class for the FCSF (first come first serve) scheduling algorithm which schedules tasks in the order in which they request the CPU.
- **Scheduler_sjf.h:**header file that declare a scheduler class that implements the SJF(Shortest job first) scheduling algorithm.
- **Scheduler_sjf.cpp:** A source file that contains the functions that has scheduler class for the SJF (shortest job first) scheduling algorithm which schedules tasks in order of the length of the tasks' next CPU burst.
- **Scheduler_priority.h:**header file that declare a scheduler class that implements the Priority Scheduling scheduling algorithm.

- **Scheduler_priority.cpp:**A source file that contains the functions that has scheduler class for the Priority scheduling algorithm which schedules tasks based on priority. A bigger number means higher priority.
- **Scheduler_RR.h:**header file that declare a scheduler class that implements the RR(Round Robin) scheduling algorithm.
- **Scheduler_RR.cpp:** A source file that contains the functions that has scheduler class for the RR(Round Robin) scheduling algorithm where each task runs for a time quantum (or for the remainder of its CPU burst).

How to Compile and Run the Program:

- *To compile the program, use command:*

  FCFS:make fcfs

  SJF:make sjf

  Priority: make priority

  Round Robin: make rr

- *To run the program, use command:*

  FCFS:./fcfs schedule.txt

  SJF: ./sjf schedule.txt

  Priority: ./priority schedule.txt

  Round Robin: ./rr schedule.txt time quantum-> in this case-> ./rr schedule.txt 10

References:

We planned out our program by writing down the given in example.out and got the exact results and after we looked up a reference for the files so we can make sure we are on the right track.We got the formula for FCFS from FCFS Source and we also found there how we can calculate the turnaround and waiting time for FCFS but because we needed the "completion" so we declared it in the fcfs.h file and did the formula: *completion=completion+myQueue[i].burst_time.* After FCFS, we worked on SJF which was Referenced by SJF Source. For priority we used this reference, Priority source . Final reference we used was for round robin and we used this source to double check our steps RR source

Implementation:

As per the given starter code from the CS433 GitHub from the professor, each program was able to read a text file of tasks, run the corresponding scheduling algorithm and print out the average turnaround and waiting time.It is assumed that all tasks arrive at the same time 0, and in the order according to the list in the input text file. An example input file "schedule.txt" is provided for testing your programs. The results of our implementation should match the "example_out.txt" for the example input file and the tests on replit as well. In all cpp files, we had a ***constructor function, destructor function, innit function, print results function, and stimulate function*** and all were being called in pcb.h file .We didn't need to add more functions other than the one goven from the professor

Constructor for all cpp files: intializing anything that will be used through out the file to 0 (ex: turnaround time and average waiting time)

*Destructor function for all cpp files:*was the same , we took care of it by doing: while vector is not empty we remove the last element in the vector by using *pop_back*

Innit function: called once before the simulation starts.It is used to initialize the scheduler.Parameters: process_list which is is the vector list of processes in the simulation. We started this function by setting index to 0, setting q size to the vector size, then had a while loop to keep looping while the index is not equal to q_size and in this loop it keeps on adding a new element at the end of the vector, and then finally we increment index and increment count/curr_size

print results function: prints the result of the turn around time, waiting time inside a for loop so it can keep on printing as it increases. It also prints the average turn around time and average waiting time after it's done with the loop

stimulate function: This function simulates the scheduling of processes in the ready queue.It stops when all processes are finished. This function changed depending on which scheduling we were working on, but it mainly had the formulas for calculating turn-around time, total turn-around time, waiting time, total waiting time, average turn-around time, average waiting time which was referenced from the sources above . It also had any needed sorting, depending on if we are doing a scheduling algorithm which schedules tasks in the order in which they request the CPU or tasks in the order in which they request the CPU or if a bigger number means higher priority or each task runs for a time quantum (or for the remainder of its CPU burst).