EJ Lilagan, Kyrstn Hall

Dr. Xiaoyu Zhang

CS433 – Operating Systems

16 October 2022

Assignment 3 Report

Submitted Files

- **scheduler_fcfs.h**
- **scheduler_fcfs.cpp**
- **scheduler_sjf.h**
- **scheduler_sjf.cpp**
- **scheduler_priority.h**
- **scheduler_priority.cpp**
- **scheduler_rr.h**
- **scheduler_rr.cpp**
- **pcb.h (to add turnaround and wait)**

How to Compile/Run the Program:

To compile: make fcfs

make sjf

make priority

make rr

Execute: ./fcfs schedule.txt (FCFS Scheduler)

./sjf schedule.txt (SJF Scheduler)

./priority schedule.txt (Priority Scheduler)

./rr schedule.txt 6 (Round Robin Scheduler)

./priority_rr schedule.txt 6 (Priority RR Scheduler)

Features Implemented (via. scheduler):

1. FCFS – First Come First Serve is to obtain and read the first process that is up in the list. It will not account burst time or priority as its constraints and will go through the order.
    a. Constructor initializes the count, total_turnaround, and total_wait to default or to zero as it will be used for functions within the file.
    b. Destructor will pop the following inside the queue and vector to ensure that the queue and vector is empty.
    c. The init function will require to push in the process_list of processes into the queue for tracking the task name, priority, and CPU burst (i.e. T1, 4, 20.)
    d. print_results() will display the calculations to be performed consisting of turn-around time and its average including the waiting time and its average.
    e. simulate() will function as FCFS, which will require a queue to call the first process (Q.front.burst_time) to calculate the total turnaround time and waiting time. In addition, it will be supported with a while loop to keep running until the queue is empty.
2. SJF – Short Job First will read through the list of processes with the shortest burst time. In other words, it will track the shortest job to be executed first.
    a. Similar to FCFS, the constructor will initialize the count, total_turnaround, and total_wait to zero.
    b. Will work similar to FCFS to ensure that the queue will be empty.
    c. Unlike FCFS, it is important to initialize the following processes by the shortest burst time, so in that case we used the sort function that is called from the algorithm class (which will be further explained on bullet ‘i’.) After the sort, it will do the same process as FCFS to push the processes inside the queue.
        i. The sort will be used for assigning the first and last elements of the vector to be sorted for finding the shortest burst time.
    d. Likewise to FCFS, it will display the calculations performed to collect turn-around time and its average and as well as waiting time and its average.
    e. simulate() will work the same as FCFS as it will calculate the total_turnaround time and total_wait time and simulate how the SJF scheduler should work.
3. Priority – will read through the list of processes with highest priority with highest number. In short, with a higher priority, that process will be executed first.
    a. Similar to the previous two schedulers, the constructor will initialize the count, total_turnaround and total_wait to zero.
    b. Destructor will work the same as the other two schedulers as it will ensure the queue to be empty.
    c. Similar to SJF, it will require the sort function to assign the first and last elements of the vector to sort the highest priority. Once the sort is complete, the processes will be pushed inside the queue.
    d. Similar to the other two schedulers, the following will display the turnaround time and its average as well as the waiting time and its average.
    e. simulate() will continue to work the same as it will calculate the total_turnaround time and total_wait time that will be simulated by highest priority.
4. Round Robin – will be unique compared to the other schedulers where time quantum will determine each process’ burst time. In other words, it runs in a cycle that slices burst time to a waiting queue to have equal turns to run certain processors.

a. As opposed to the other schedulers, it is important to keep into account that quantum will be used, so it will be assigned with time_quantum that is on the parameter. Including count, total_turnaround and total_wait, all will be set to 0.
b. Similar to all the schedulers, it will pop the following processes in the queue to ensure that it is empty.
c. Since time quantum will be held into account, it will be unnecessary to call it on the initialize function since we did not have to worry about sorting priority or burst time. Similarly to FCFS, it will be set up that way.
d. With the previous schedulers, it will sort the id of processes to print out the order of the performance with RR. The following will continue to do the same by displaying the turnaround, waiting, and both of their averages.
e. Uniquely, this function will have two conditionals while it loops through the list. The first condition will check if the curr_burst time will be higher than the quantum and when the curr_burst is lower than the quantum. Once both conditions are met, it will provide the process that is running and will display the curr_burst to show the cycle. Pop both the queue and vector once completed.
    i. When the first condition is called, it will get the front element of the queue and assign that with the curr_burst - quantum to store the remaining units to be put into the queue. After, it will assign curr_burst with the quantum to complete the Round Robin, so it will be pushed back on the PCB in a queue and vector (both similarly used to keep track of quantum and amount of burst time left until burst time is less than quantum.) Sum up curr_turnaround with curr_burst to get the final result.
    ii. When the second condition is called, it will continue to output all the last burst times that are left in the processes and record the total turn-around time and total waiting time.

Additional:

- Queue<PCB> Q allows storing processes into a waiting queue like burst time or priority.
- Vector<PCB> done allows storing processes with turnaround time & waiting times, and to display it in print_results().
- Variables name, id, priority, burst_time, arrival_time, turnaround, wait, will be all used to access the following queue/vector to access into the PCB of processes.

Choice of Data Structure: With a general idea of using schedulers, we decided that it was best to use queues and vectors for this programming assignment. Firstly queues were primarily used as the ready queue to account for pushing into the process list of vectors. As shown on the init functions of all the submitted files, they will all share a queue to read the process that will be simulated to get turnaround/waiting time. In contrast, the vectors are primarily used to store the list of processes and PCB's to track the following turnaround/waiting time to be outputted onto the compiler.

Lessons Learned/Re-Learned:

As always, we started this assignment early which was beneficial for us. It was definitely overwhelming at first glance because there was just so many different files. Trying to understand where certain cout's go and what happens inside init() was also a bit confusing. However, after

we got a good idea of how the program works, there wasn't much trouble completing this assignment. I feel as we had a solid understanding of how scheduling works from lectures and our homeworks. I did find it cool how we were able to create a program that allows us to run what we have been learning in class.

Sources

- How to override
- Another way to use override_2 and override_3
- How to sort a vector
- Vector library used
- Queue library used