Assignment 4

Hide Assignment Information

Instructions

Assignment overview

As discussed in class, there are multiple CPU scheduling algorithms that could be used for a given set of processes. There are several methods one could employ to evaluate which CPU scheduling algorithm is best. Deterministic modeling is one type of analytic evaluation. This method takes a particular predetermined workload and defines the performance of each algorithm for that workload. We employed this method **manually** in class to determine the wait and turnaround times for a set of processes with a given arrival time and burst time.

For this assignment, write a C program which will **automatically** simulate multiple CPU scheduling algorithms for any given set of processes. The program will maintain wait and turnaround times for each process over time and then display the average wait and turnaround time for all processes.

Purpose

The goals of this assignment are the following:

- Understand the kind of data the CPU scheduler would need to keep track of to execute certain algorithms and how it might track that data over time
- Learn the differences between various CPU scheduling algorithms
  - Note how those differences may cause shorter or longer wait and turnaround times for the same process
  - Note how those differences influence average wait and average turnaround times

Computing platform

You are welcome to develop your program on your own workstation if you wish, but you are responsible for ensuring that your program compiles and runs without error on the Gaul computing platform. Marks will be deducted if your program fails to compile, or your program runs into errors on Gaul.

- Gaul tips

Instructions

You will write a program called `assignment-4.c`. This program will:

- Accept 2 or 3 parameters (You can assume parameters are supplied in this order)
  - The first parameter defines the algorithm to simulate.

        - -f for First Come First Served (non-preemptive)
        - -s for Shortest Job First. (preemptive based on burst time).
        - -r <integer> for Round Robin (preemptive based on time quantum)
            - (If -r is supplied, the next parameter is a positive integer defining the time quantum)
    - The last parameter defines the filename to read input from. For the purposes of the examples, this will always be `assignment-4-input.csv`, but of course, you can create your own versions for testing
    - Your program should fail gracefully if there is an issue with the input parameters (eg. invalid option, file cannot be opened, invalid time quantum, etc.)
- Depending on which algorithm is selected, your program should display the simulation of that algorithm including each processes' wait and turnaround time for each "tick" (one tick per line).
- Once all processes have finished, display the wait time and turnaround time for each process. Then display the average wait time and average turnaround time for all processes using that algorithm (to an accuracy of at least one decimal point)

Output

Executing `./assignment-4 -f assignment-4-input.csv` should produce the following output (truncated with "..." for brevity):

```
First Come First Served
T0 : P0 - Burst left 3, Wait time 0, Turnaround time 0
T1 : P0 - Burst left 2, Wait time 0, Turnaround time 1
T2 : P0 - Burst left 1, Wait time 0, Turnaround time 2
T3 : P1 - Burst left 8, Wait time 2, Turnaround time 2
T4 : P1 - Burst left 7, Wait time 2, Turnaround time 3
T5 : P1 - Burst left 6, Wait time 2, Turnaround time 4
T6 : P1 - Burst left 5, Wait time 2, Turnaround time 5
T7 : P1 - Burst left 4, Wait time 2, Turnaround time 6
T8 : P1 - Burst left 3, Wait time 2, Turnaround time 7
...
T209: P19 - Burst left  2, Wait time 186, Turnaround time 190
T210: P19 - Burst left  1, Wait time 186, Turnaround time 191

P0
        Waiting time:           0
        Turnaround time:        3

P1
        Waiting time:           2
        Turnaround time:       10
...
P18
        Waiting time:         178
        Turnaround time:      187

P19
        Waiting time:         186
        Turnaround time:      192

Total average waiting time:     84.6
Total average turnaround time:  95.1
```

Note that in this example, even though P1 arrived at T1, P0 had not finished yet. There is no preemption so P0 runs until completion at the end of T2. Then at T3, P1 will begin its burst.

Executing `./assignment-4 -s assignment-4-input.csv` should produce the following output:

```
Shortest Job First
T0 : P0  - Burst left  3, Wait time  0, Turnaround time  0
T1 : P0  - Burst left  2, Wait time  0, Turnaround time  1
T2 : P0  - Burst left  1, Wait time  0, Turnaround time  2
T3 : P2  - Burst left  1, Wait time  1, Turnaround time  1
T4 : P3  - Burst left  4, Wait time  1, Turnaround time  1
T5 : P3  - Burst left  3, Wait time  1, Turnaround time  2
T6 : P3  - Burst left  2, Wait time  1, Turnaround time  3
T7 : P3  - Burst left  1, Wait time  1, Turnaround time  4
T8 : P8  - Burst left  3, Wait time  0, Turnaround time  0
...
T209: P5  - Burst left  2, Wait time 187, Turnaround time 204
T210: P5  - Burst left  1, Wait time 187, Turnaround time 205

P0
        Waiting time:           0
        Turnaround time:        3

P1
        Waiting time:          11
        Turnaround time:       19
...
P18
        Waiting time:           8
        Turnaround time:       17

P19
        Waiting time:           1
        Turnaround time:        7

Total average waiting time:     58.8
Total average turnaround time:  69.3
```

Note that in this example, at the start of T2, P0 still has a burst left of 1. P2 has arrived and also has a burst left of 1. Since a tie goes to the currently running process (we don't want to context switch if we don't have to), P0 should not be preempted and should just continue. At T3, P2 is the shortest so it goes next. At T7, P3 completes its last burst. At T8 a context switch must occur anyway and P8 (which just arrived) happens to be the shortest job in the list so it goes next. You should observe that when selecting a new process, the burst time remaining will be higher and higher until P5, which happens to have the longest burst time of 19, goes last.

Executing `./assignment-4 -r 3 assignment-4-input.csv` should produce the following output:

```
Round Robin with Quantum 3
T0 : P0  - Burst left  3, Wait time  0, Turnaround time  0
T1 : P0  - Burst left  2, Wait time  0, Turnaround time  1
T2 : P0  - Burst left  1, Wait time  0, Turnaround time  2
T3 : P1  - Burst left  8, Wait time  2, Turnaround time  2
T4 : P1  - Burst left  7, Wait time  2, Turnaround time  3
T5 : P1  - Burst left  6, Wait time  2, Turnaround time  4
T6 : P2  - Burst left  1, Wait time  4, Turnaround time  4
T7 : P3  - Burst left  4, Wait time  4, Turnaround time  4
T8 : P3  - Burst left  3, Wait time  4, Turnaround time  5
...
T209: P15 - Burst left  1, Wait time 179, Turnaround time 194
T210: P5  - Burst left  1, Wait time 187, Turnaround time 205
```

```
P0
        Waiting time:           0
        Turnaround time:        3

P1
        Waiting time:          95
        Turnaround time:      103
...
P18
        Waiting time:         116
        Turnaround time:      125

P19
        Waiting time:          77
        Turnaround time:       83

Total average waiting time:     118.9
Total average turnaround time:  129.4
```

Note that P2 was able to finish its CPU burst at T6 which was less than the time quantum of 3. Therefore, a context switch to P3 occurs and the time quantum is reset to 3. P3 is then awarded a time quantum 3.

Executing `./assignment-4 -r 12 assignment-4-input.csv` should produce the following output:

```
Round Robin with Quantum 12
T0 : P0  - Burst left  3, Wait time  0, Turnaround time  0
T1 : P0  - Burst left  2, Wait time  0, Turnaround time  1
T2 : P0  - Burst left  1, Wait time  0, Turnaround time  2
T3 : P1  - Burst left  8, Wait time  2, Turnaround time  2
T4 : P1  - Burst left  7, Wait time  2, Turnaround time  3
T5 : P1  - Burst left  6, Wait time  2, Turnaround time  4
T6 : P1  - Burst left  5, Wait time  2, Turnaround time  5
T7 : P1  - Burst left  4, Wait time  2, Turnaround time  6
T8 : P1  - Burst left  3, Wait time  2, Turnaround time  7
...
T209: P15 - Burst left  1, Wait time 179, Turnaround time 194
T210: P17 - Burst left  1, Wait time 181, Turnaround time 193

P0
        Waiting time:           0
        Turnaround time:        3

P1
        Waiting time:           2
        Turnaround time:       10
...
P18
        Waiting time:         144
        Turnaround time:      153

P19
        Waiting time:         152
        Turnaround time:      158

Total average waiting time:     112.8
Total average turnaround time:  123.4
```

Helpful hints

- Your program should read in all the data and populate the data into some kind of data structure. An array of structures or a multi-dimensional array would be the best way to represent the data. After you have read the entire input file and have the data, you run a loop to step through all the processes until the

CPU burst for all processes is 0. This should be easier than trying to read from the file as you go - although you may do this if you wish.

- Each process will need to retain the **arrival time**, the **burst time**, the **wait time**, and the **turnaround time**. At every tick, you must make some updates on **all** processes, not just the active process.
  - The **arrival time** is easily determined by the process id. For example, the arrival time for P0 is 0. The arrival time for P4 is 4. This value never gets updated.
  - The **burst time** will start at the value read in. It will be decremented every time it is the **active** process.
  - The **wait time** will start at 0. It will be incremented every time the current time is greater than or equal to the arrival time, the burst time is not 0, and it is **not the active process**. In other words, update the wait time every time the process is in the ready queue.
  - The **turnaround time** will start at 0. It will be incremented every time the current time is greater than or equal to the arrival time, and the burst time is not 0. This is irrespective of whether this is the active process or if it is in the ready queue.
- It's important to remember that each tick represents the start of that tick. For example:

  -
    ```
    T0 : P0  - Burst left  3, Wait time  0, Turnaround time  0
    T1 : P0  - Burst left  2, Wait time  0, Turnaround time  1
    T2 : P0  - Burst left  1, Wait time  0, Turnaround time  2
    T3 : P1  - Burst left  8, Wait time  2, Turnaround time  2
    ```
  - This means P0 started at T0 and finished at the **start** of T3 (represented as 0-3 in our Gantt charts). So its turnaround time is 3, although this is not actually represented on the screen. The last turnaround time we saw was 2 but that was at the **start** of T2. P1 arrived at T1 and did not start until T3. Therefore, its wait time is 3-1 = 2. Its wait time and turnaround time is at least 2.
  - Every time tick that P1 is active, the turnaround time is **incremented** and the burst time is **decremented**. Every time tick that P1 is not active and not complete, the wait time and the turn around time is **incremented**.
- Here is some sample code for reading in a file line-by-line
  - https://www.geeksforgeeks.org/scanf-and-fscanf-in-c/
- Checking your output
  - We manually calculated a few examples in class. Does your implementation match the calculations in the notes?
  - LLM tools such as ChatGPT can be helpful at verifying your calculations. Simply provide your list of processes, the burst time, the arrival time, the algorithm to use, and ask for the burst times and turnaround times. However, as is the case with all LLM interactions, you should still verify some of the results yourself. You can do this with a gantt chart as we have seen in class.

Submitting

When you are finished your assignment, follow these steps:

1. **Cite your work:** Did you use code that you did not write yourself? Just like an essay, if you did not write it yourself, you must properly attribute the author. Leveraging existing code for your assignment is acceptable but it should

represent a minor portion of your submission. Specifically, any code that fulfills the learning outcomes detailed in the Purpose section above should be almost entirely your own. Contact the course instructor if you are unsure.

   1. If you used sample code from the course notes or the textbook, in the comments, state which sample program you used or which page(s) from the lecture notes you used.
   2. If you used code from an online source (eg. StackOverflow, GeeksForGeeks, etc.), in the comments, provide the link where the code was found.
   3. If you used an LLM such as ChatGPT, in the comments, provide the link to the prompts you used to generate the code.
   4. Using code from another student is a violation of the policy on academic integrity and is not permitted.

2. Verify your code runs on Gaul. Create screenshots demonstrating your program and place them in the `Assignment-4` directory. Remove any spurious files (eg. the executable version of your program). The directory should have a single file called `assignment-4.c` and any screenshots. Supporting files like `Makefiles` or `READMEs` are acceptable.
3. If necessary, run the following command to get out of the `Assignment-4` directory: `cd ..`
4. Package your assignment into a tarball: `tar -cvf Assignment-4.tar Assignment-4`
5. Verify the contents of your tarball (`tar -tvf Assignment-4.tar`) (`du -sh Assignment-4.tar`). The tarball should have your `assignment-4.c` file and all screenshots. **If your tarball is 10kb in size** you have an empty tarball and you made an error on this step. Make sure you are properly creating your tarball with the right files in it.
6. Use an SFTP program to download the tarball to your local workstation and then upload it to OWL.

Due on Mar 18, 2025 11:55 PM

Available on Mar 5, 2025 12:00 AM. **Access restricted before availability starts.**

Available until Mar 20, 2025 11:55 PM. **Access restricted after availability ends.**

Attachments

assignment-4-input.csv (142 Bytes)

Download All Files

Show Rubrics

Submit Assignment

Allowed File Extensions

tar

Files to submit

(0) file(s) to submit

After uploading, you must click Submit to complete the submission.

Add a File Record Audio Record Video

Comments