ITU Computer Engineering Department
BLG 223E Data Structures, Summer 2021
Homework #2
Due July 16, 2021 23:59

Definition

You are asked to implement classes that simulate a single processor system that can execute jobs in the determined order by the input file. Please pay attention to the following details inyour implementations:

- You should implement two classes, one for representing the processor, other for representing a single job.
- Your processor class should contain a queue data structure to keep the incoming jobs in the waiting queue. A data structure (not necessarily a queue) to keep current running jobs and another to keep finished jobs.
- Your processor class should contain an MIPS value which represent millions of instructions that the processor can execute in a second.
- Your job class should contain an MI value which represent the total work, as millions of instructions, required by the job.
- Your job class should contain an id field as an integer.
- When running, the processor should spare a fair amount of MIPS to its running jobs. For instance if the processor's MIPS value is 100 and there are 5 jobs running, each of them should complete 20 MI of their total work each second. If there are 3 jobs running each of them should complete 33 MI (you don't need to use fractions) of their total work each second.
- By checking a predetermined value (MIPS THRESHOLD), your processor should stop accepting new jobs when its current utilization is high. For instance if your processors MIPS is 100 and the utilization value is set set to 10 MIPS and there are 9 jobs running it shouldn't accept the 10th job since it will make the processor spare 10 MI to each job.
- Your processor should keep running in discrete time (second by second).
- You should simulate time by incrementing an integer variable, second by second. No need to perform the operations on real time. Also, you won't need to perform fractional calculations on job running times.
- Your processor should read the incoming jobs as events from a file, defined in the following section.

Input-Output

Your program should accept a filename as a command line parameter. In the accepted file, each line will contain a distinct command and parameters that should be processed by your program. In the input, the following commands may be given:

- Each line starts with an integer value which indicate the time of the event, happening in the line.
- The first line always starts with 0, followed by two numbers where the first number represents the MIPS value of the processor and the second value represents the MIPS THRESHOLD defined in the previous section.

- <time> JOB <id> <mi>: Sets the id of the job to start on time <time> requiring <mi> processing power.
- <time> PRINTQ: Print list of job ids in the waiting queue on <time>. THe list should be preceded with "WQ " (see calico file and inputs)
- <time> PRINT <id>: On time value <time> print the status of the job with <id>. The status of the job is represented by the queue it is currently in ('W' for wait queue, 'R' for running, 'F' for finished), start and finish times (use -1 if it has not yet started or finished). (see calico file and inputs)

Only the PRINT and PRINTQ commands should produce output. The output they need to produce is explained in the previous definitions.

The input file would be error-free both syntactically and semantically. The time value, for each row, would never decrease but it may be the sam efor consecutive rows, representing multiple events happening at the same time value. You may use input order for such events. Briefly, you shouldn't need to go back and forth while processing the input. Just read line by line and eecute event for each line.

Deliver

Please zip and deliver the directory structure defined below:

- HW1 : Topmost folder, that will contain all the folders in your submission. No other files should be present under this folder in your submission.
- HW1/src: Contains all the *.cpp files
- HW1/src/main.cpp: Contains you main function.
- HW1/src/Processor.cpp: Contains your processor class.
- HW1/src/Job.cpp: Contains your job class.
- HW1/include: Contains all the header files you use
- HW1/bin: An empty directory that will contain objective files when your project is compiled

Please check the calico test file in the homework definition to see how your files will be compiled and tested.

Restrictions and Guidelines

- Compilation environment: Only the code that can be compiled by the environment of the container definition provided in ninova will be accepted.
- Testing of your program will be performed using Calico (https://calico.readthedocs.io/en/latest/). Test cases that will be used to test your homework is provided as an attachment in ninova.
- **STL usage is allowed and promoted.**
- **This homework is for individual submissions**. Any kind of code sharing or code adaptation from an external source is strictly forbidden. Submitted code will undergo a plagiarism check process and plagiarsim penalties may be given if the submitted code's similarity is approved by the instuctor.
- Make sure you write your name and number in all of the files of your project, in the following format:
  /* @Author
  Student Name:<studentname>
  Student ID :<studentid>

Date:<date>*/

- Only electronic submissions through Ninova will be accepted no later than deadline
- Use comments wherever necessary in your code to explain what you did.