Github Link: FeevaDVA/CMSC312OSsimulator (github.com)

CMSC312OSsimulator

This is an Operating Simulator made for my class CMSC 312 at VCU

The design of this Simulator uses a GUI as the main way of navigating and will show the processes in real time. You can also add and stop the simulator in real time. I decided to use a GUI instead of a UI for phase I because I felt that it was much more natural to navigate a GUI for the User and will make it much more pleasing to watch the simulator. As for what scheduling algorithm that I choose to run I decided to go with round robin as I felt that it was the most pleasant to watch work. The quantum that I settled with was 20 not because it was the most optimal, I felt that it was the most pleasing to watch run as you can see it complete task if I used a bigger number there are chances that I would skip over some tasks as the way that I run the simulator is to not run each CPU cycle but to decrement the amount of cycles from the process and match the task. I use a list to keep all the active processes and to check them but when they are out of cycles, they get marked to TERMINATED which then moves them from the process list to the terminated list. This is to keep the list of processes in the list manageable so that we won't have to cycle through a list of ever-growing processes the only list that grows is the terminated list. critical sections are dealt with by a semaphore like operation where P is called in the template and then V is called when it's done. No other processes can interrupt it and enter the critical section while a process is in it, they will just go to busy wait. I spent quite a while on making sure that the program runs smoothly and that it looks satisfying to watch the processes complete. I setup ways for memory to be implemented and I think that adding fork will be easy also since I made it check the tasks, but I think that for Phase I that this came out well. The process List class is my PCB like class, so it handles everything pertaining to process information. For the times I use real time so if the program does lag a little it will find a time difference which is why the wait time of a single process is a little more than zero but this is something not much of a problem I think. On the github is a better formatted version of the document in my opinion so please feel free to refer to that also as this one it should have the same information just looks a little better

Installation

There is a runnable jar in the root of the project. Make sure that the jar is in the same folder as the Process Templates folder since that's where it grabs the templates from. As for java version the jar was made with the Java 17 development kit so make sure that Java 17 is downloaded. JDK 17 can be found here https://www.oracle.com/java/technologies/downloads/

For compiling I used IntelliJ to make and compile the project as well as their GUI builder so whatever dependencies you need for that you will need for this

Usage

There is a start/stop button at the button along with a add button with two text fields

The first of which is the number of processes to add to the os

The second is the template you want to generate these processes from right now there is only 6 templates 1-6

1 - short running calculate heavy process

2 - calculate heavy process with critical section

3 - balanced process

4 - calculate heavy process

5 - long calculate heavy process with critical section

6 - i/o heavy process

Try to keep the number of processes running at a time to around 10 - 15 anymore and it runs somewhat slow but it can be done

The average turnaround time is shown at the bottom with the average wait time which is displayed in seconds

Each process will show its number/pid, state, and remaining cycles until it is terminated where it will then show its turnaround time in seconds

The simulator can be started and stopped via the start/stop button.

The three section of list is split into ready list, waiting, and all the processes.

Documentation

- Class: Scheduler
  - Variables:
    - private static final int quantum
      - the quantum for the round robin scheduler
    - private final ProcessList list
      - houses the PCB
    - private final mainGUI main
      - stores the GUI so that we can update lists
    - private int cycleCount
      - counts the cycles(not yet needed)
    - private int runningProc
      - returns the current running process(used to debug)
    - private int semaphore
      - the semaphore value I use it a bit different its 1 if there is a process in the critical section its 0 if not
    - private int procCrit
      - the process currently in the critical section
    - private volatile boolean exit
      - tells the simulator to stop or not
  - Constructor: public Scheduler(ProcessList l, MainGUI m)
  - Desc: This creates the scheduler object which extends a thread so it acts as a runnable thread
  - Method: public void setExit(boolean e)
  - Desc: This will set the boolean exit to the given boolean e
  - Method: public void run()
  - Desc: This is the main loop of the class this will while exit is not true run a scheduler algorithm on the list of processes from the class
  - processlist the scheduler algo that it is running as of now is round robin it will try to run at 20 CPU cycles per 2 milliseconds but the
  - number of processes in the list will make it longer

- Class: ProcessList
  - Variables:
    - private int countProcess
      - Keeps count of the processes to assign them a pid
    - private List<Process> processlist
      - The list of all the processes that aren't terminated
    - private List<Process> terminatedList
      - The list of all terminated processes

- Constructor: public ProcessList()
  - Desc: creates a list for processes, the count of processes, and a list for terminated processes
- Method: public void generateProcesses(int count, int temp)
  - Desc: generates a given number of processes of a given template number and adds them to the process list
- Method: public List<Process> getList()
  - Desc: returns the list of processes
- Method: public List<Process> getTerminatedList()
  - Desc: returns the list of terminated processes
- Method: public Process getProcess(int n)
  - Desc: returns the process at given location n from the process list
- Method: public void removeProcess(int n)
  - Desc: removes the process at location n in the process list

- Class: Task
  - Variables:
    - private String taskName
      - stores the name of the task
    - private int time
      - stores the time/cycles of the task
    - private int memory
      - stores the memory needed for the task
  - Constructor: public Task(String name, int t, int m)
  - Desc: the constructor for the task object that takes in a name, time, and memory of the task
  - Method: public int getTime()
    - Desc: Returns the time/cycles of the task
  - Method: public void setTime(int t)
    - Desc: sets the time/cycles of the task to the int t
  - Method: public int getMemory()]
    - Desc: returns the memory of the task
  - Method: public String getTaskName()
    - Desc: returns the name of the task

- Class: Process
  - Variables:
    - private int number
      - Stores the number/pid of the process
    - private double burstTime
      - stores the estimated burst time of the application
    - private long arrivalTime, completionTime

        - stores the arrival time and completion time of the process in milliseconds
    - private String state
        - stores the state of the process
    - private List<Task> taskList
        - stores a list of all the task for the process
    - private int currentTask
        - stores the location of what task the process is currently on
- Constructor: public Process(int tempNum, int num)
    - Desc: Constructor for the process class that takes in the template number for the process and the number of the process also generates the tasks for the process
- Method: private void generateTasks(int tempNum)
    - Desc: generates a task from the template number then uses the file for the template to make all the tasks from it
- Method: public void setArrivalTime(long n)
    - Desc: sets the arrival time of the process in milliseconds with the given long n
- Method:public void setCompletionTime(long n)
    - Desc: set the completion time of the process in milliseconds with the given long n
- Method:public long getTurnAroundTime()
    - Desc: returns the turnaround time in milliseconds of the process which is completion time - arrival time
- Method:public long getWaitTime()
    - Desc: returns the wait time of the process in milliseconds which is turnaround time - burst time
- Method:public int getTotalMem()
    - Desc: returns the remaining memory of the process
- Method:public int getTotalCycles()
    - Desc: returns the remaining cycles of the process
- Method:public Task getCurrentTask()
    - Desc: returns the current task of the process
- Method:public boolean nextTask()
    - Desc: return true if there is a next task and moves to it returns false if there is no next task
- Method:public void updateState(String newState)
    - Desc: updates the state of the process to the string newState
- Method:public String getState()
    - Desc: returns the state of the process
- Method:public int getNumber()
    - Desc: returns the number of the process

- Class: MainGUI
  - Constructor: public mainGUI()
    - Desc: makes a new ProcessList called pcb and starts the thread scheduler initializes all the text and button fields also the scrollable lists
  - Method: public JButton getStartButton()
    - Desc: returns the JButton startButton
  - Method: public static void main(String[] args)
    - Desc: starts the MainGUI by making a new object
  - Method: private int getTempNumber()
    - Desc: returns the template number entered the text field of a0TextField1
  - Method: private int getNumberProcess()
    - Desc: returns the number of processes to be added from the text field of a0TextField
  - Method: public void updateList()]
    - Desc: updates the three scrollable lists with all the processes from the process list of the PCB and organizes them into their columns it will also update the average turnaround time and wait time