Github link: <https://github.com/huhu72/Project-1>

Grade using commit ID: 11f1a8cbc702a8de4a989404d8c0f1039c4aa5cc

Text

Description automatically generated

The OS first instantiates the CPU, Process, Dispatcher, and Scheduler.

After doing that, it will connect the dispatcher, CPU, and scheduler with each other. After connecting each other, it will then call on the method the process creation method, createProcesssesPrompt() to create all the processes that the user requests. It then puts all of those processes in the job queue within the cpu. Once that is done, it will grab the job queue and send it to the dispatcher class where the, unsorted, ready queue is then populated. Once the cpu starts running the processes, the OS will print out the status every 4ms after waiting 1ms after its called.

Text

Description automatically generated

This method will take a users input as a bash command and then parse into an array of process arguments when the user types “create”. From there it will call createProcesses with the input of template name and how many processes they wish to create etc…

Text

Description automatically generated

createProcesses will create a specified number of processes, based on user input and template, and create a process with a unique name and PID. After the process is created, it will then put newly created process into the job queue. Then I will create a PCB that is linked up with the process and adds it into a hashmap that is instantiated in the cpu.

Text

Description automatically generated

After all of the operation from the last section are completed, the OS will call this run command.

The cpu will first grab from the ready queue if the waiting queue is empty.

After grabbing the process, set the state to RUN, check if the first command is a I/O command so that it can put that process in the waiting queue, and start a timer that will increment the priority of all processes every 10 seconds after waiting 10 seconds when the timer is first called.

Text

Description automatically generated

It will then enter a while loop that can only be ended using break;. Before running any cycles, it will start the time quantum clock that is handled by the scheduler. It will then enter another while loop that will increment the cycle of the current command the process is on until the scheduler has reached the specified round robin or the process has ran all of the cycles in that command and then will increment the program counter and set the cycle counter back to 0 and then break out of the while loop.

Text

Description automatically generated

The CPU will then decided if the process has anymore commands to run or if there are any more cycles that need to be ran. If there are no more commands to be ran, the cpu will then set the PCB associated to the process, change its state and update its map of PCB it currently holds. If it determines that a process needs to run more cycles, it will then add it to its respected queue in the dispatcher. Text

Description automatically generated

If there are more cycles to run, it will look at its next command and places the process in the respected queue and resets the processes cycle counter.

Text

Description automatically generated

The dispatchers addToWaiting/ReadyQueue will take the process that needs to be added to its respected queue, and calls apon the sort method which will pass that queue into the scheduler to be sorted.

Text

Description automatically generated

When getting the process from either queue, it will grab the head of the queue and send that process back to the CPU and will also update the hashmap of the PCBs just incase there are anything changed within the PCB

Text, timeline

Description automatically generated

The scheduler will sort the queues by priority ,with the highest number being the most important.

A picture containing graphical user interface

Description automatically generated

The timeQuantum clock will run for 4ms at max every time this method is called. Onces its called it will stop its self so that a new timer can run the next time it gets called

In this project, the round robin with priority scheduling algoritm was used. The quantum time was accuired by running 4 processes within the os and getting its average run time.

Commands:

* Help
* Template
* Create <template name> <number of desired processes> …

Requirements:

* Atleast java 8