Bharat Venkat

December 2, 2018

Software and hardware requirements: Project created on MacBook Pro, using Java version 11.0.1. Needs Java JDK to run.

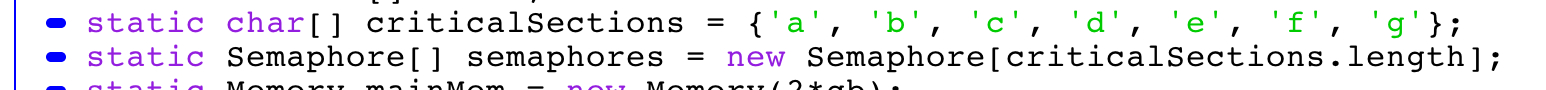
1. Click button of choice then click jframe once anywhere
2. Enter either names or number of files, then press enter, then close Jdialog
3. Close original Jdialog
4. Select scheduler, then close jdialog
5. Enter number of cycles desired then press enter then close Jdialog.
6. Output will be displayed in JFrame!

Process/PCB: Processes are represented in the Process class, each one having its own PCB along with other important functions. Generated processes are stored as job files, which are txt files.

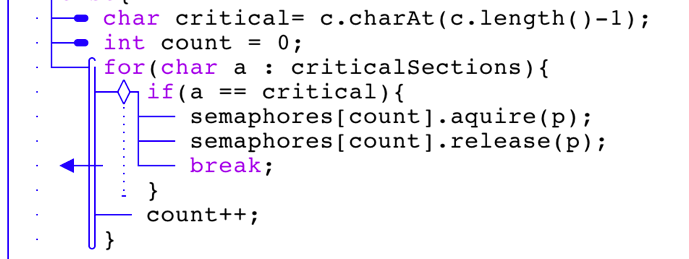
A screenshot of a social media post

Description automatically generated

A screenshot of a cell phone

Description automatically generated

Each process that is generated by the generator has a Critical section in it. It is one of 7 possible critical sections determined randomly. Each section has an associated semaphore that is accessed when an attempt to access the critical section is resolved. In addition, memory accesses are protected with semaphores as well, due to multithreading being implemented



A screenshot of a cell phone

Description automatically generated



Two interprocess communication methods are used, named pipes and ordinary pipes. Ordinary pipes are used to determine whether or not a process’ child has completed before taking it out of the waiting queue. Named pipes are used to send a variable between the first two processes.

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

Multielevel parent child relationship is implemented, A parent process can have a child, which can have a child of its own, which can in turn have a child of its own, etc.

A screenshot of a social media post

Description automatically generated

Two schedulers are implemented, shortest job first and a multilevel feedback queue. In the shortest job first, processes are prioritized according to their CPU burst time from high to low. In the multilevel feedback queue, processes are given 8 cycles to run on a round robin scheduler, 16 to run on a different round robin scheduler, than inserted into a FCFS queue where they are ran in order.

A screenshot of a cell phone

Description automatically generated

A screenshot of a social media post

Description automatically generated

Processes are assigned priority according to their CPU burst times.

A screenshot of a cell phone

Description automatically generated

Deadlock avoidance is implemented as such. No preemption is nullified because higher priority processes can preempt lower priority ones and take their resources. Only one resource was used for this simulator, but circular wait is avoided because resources are awarded in ascending order only (however all have same value). Hold and wait is nullified because resources are allocated before processes start execution.

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

One resource class with a limited amount of instances was used for this simulation

A screenshot of a cell phone

Description automatically generated

Memory was implemented as such. Processes are put into virtual memory at creation. If no room in VM, they are put into a waiting queue. For each frame, the simulator checks if the frame is in cache. If not, the simulator looks in main memory. if a process is not in main memory, it is swapped in. The frame or frames that were replaced the longest time ago are the frames that are replaced. If it is also not in cache it is swapped in using the same victim page replacement algorithm.

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

Hardware IO interrupts had a percentage to be thrown while a process is running. If thrown, the HardwareIOException handling code is called. IO interrupts from the job file lead to the process being put in the wait state until the IO is completed.

A screenshot of a cell phone

Description automatically generated

Multithreading was implemented using hardware threads. The Java thread library was used. There are 8 threads that run, simulating a 2 core, 4 thread each architecture.

A screenshot of a cell phone

Description automatically generated

GUI was implemented using Java AWT and Swing.

A screenshot of a social media post

Description automatically generated

Processes can be randomly generated or inputted into the GUI.

A screenshot of a social media post

Description automatically generated