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

Grade using commit ID: ff1a660ce2275b193522d8a05555299e6ac34fb0

Text

Description automatically generated

This is the code for counting semaphores. If the value for the semaphore is < than 0 when wait it called, it will add that process in its list and retrieves it when signal is called.

Timeline

Description automatically generated

The signal and wait methods are called in the while loop, that increases the program count, when it reaches either the critical section (CS) or the end of it (CE).Text

Description automatically generated

Every time signal gets called, it is sorted by the scheduler, based on the priority, before being dispatched

Text

Description automatically generated

A separate status thread is created and started after the user inputs the number processes they wish to create and those processes are created.

Text

Description automatically generated

The Status thread will continuously to listen to user input. If the user types status, the thread will print everything that is happening on the CPU as well as the status for all the processes. In the CPU class, there are segments of print commands that will only print if the status variable in the cpu is true.

Text

Description automatically generated4 process thread will be created when the cpu first runs. First, it grabs a process from the respected queue and adds it into a Runnable ArrayList. Then it will assign the runnable to a thread and runs it

Text

Description automatically generated

Fork() is called randomly by using a random number generator.A screenshot of a computer

Description automatically generated with medium confidence

Fork() will create a new process using the child.txt template and will attach the parents PID and attach its own pid to the parent for referencing purposes. After the child is created, it will add it to the ready queue.

Text

Description automatically generatedWhen a process is terminated, it will also terminate its child to fulfill cascading termination

Text

Description automatically generatedText

Description automatically generated

This is the code for grabbing a process from the respected queues

Text

Description automatically generated

This is the code for re-referencing the process threads. This segment of code will run when the process is out of commands to run. It will also run when the process it being put into a queue.

Text

Description automatically generatedText

Description automatically generated

The thread will put itself to sleep for 1 second, pausing what its currently doing, randomly and then will wake up by calling interrupt()

Text

Description automatically generatedText

Description automatically generatedText

Description automatically generatedText

Description automatically generated

The CPU decides what scheduler to use, (Round robin vs priority queue) by first creating 1000 processes based on the compare.txt template. Then it will first run the round robin scheduler to see how many cycles it can run within 3 seconds. Then It will do the same thing with the priority queue scheduler. Which ever scheduler runs the most cycles will be the scheduler that the cpu will use

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

When a process gets create, it will randomly assign memory size from 1 – 1024 to the process. After the creation, it will take the process’ memory and add it to the total memory used. If the total memory is > the max memory allowed (1024), it will directly add it to the ready queue. As a process is being terminated, it will subract its memory, and its childs if it exists, from total memory

Commands:

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

Requirements:

* Atleast java 8