Github repository link: https://github.com/mclaurind/CMSC312-OS-Simulator.git

Commit to grade: “phase 3”

Phase 3 12/03

OS Simulator Methods & Classes

OS class

public static void main (String [] args)
User is asked to choose a program and to input how many processes to spawn for the program. Then user will have the choice to choose scheduler, which will be used by 1 of 2 cpus and have a unique inter-process communication method.

Multithreading fixed to simulate a multi-processor system. Each cpu has 4 threads.

public static pcb generateProcess (String programName)
Parses program files and returns a process

Scheduler class

public synchronized void roundRobinScheduler ()
Implements a round robin scheduling algorithm. A process will go through the new, wait, and ready queues, and ultimately run until its cycles are completed in relation to the time quantum. Also checks processes against memory limits and throws random interrupts while a process is simulating. Has multilevel parent-child relationship with cascading termination.

Banker’s algorithm implemented for deadlock avoidance.

public void run()
Thread number for each thread will be assigned to a process to be simulated with a round robin scheduler.

public static Log(String activity)
Prints scheduler activity

OrdinalPipe class
Simulates a ordinal pipe and has a nested Message class

Priority Scheduler class
public synchronized void PriorityScheduling ()
Implements a priority scheduling algorithm. Processes in ready queue are sorted based on priority, through a comparator. Processes with lowest priorities are serviced earlier.

Banker’s algorithm implemented for deadlock avoidance.

public void run()
Thread number for each thread will be assigned to a process to be simulated with a priority scheduler.

public static Log(String activity)
Prints scheduler activity

OrdinalPipe class
Simulates a ordinal pipe and has a nested Message class

MQLScheduling class
public synchronized void MLQScheduling()
Splits new queue into three queues each with differing priorities. Two queues uses round robin and the third uses priority. Works well with large numbers.

public static <T> Lists <List<T> getBatches(List<T> collection, int batchSize)
Helper method to partition new queue into 3 different queues.

clock class
Used for comparing schedulers. Measures the time for each scheduler to process queues till each and everyone is terminated. User will have to run program twice to see comparison.

CSHandler Class
Implements semaphores to handles critical sections in processes.

public void waitSem(Semaphore semaphore, pcb process, int permit)
Makes processes wait for permit

public void signalSem(Semaphore semaphore, pcb process, int permit)
Signifies processes in wait queue that a permit is available

Semaphore class (nested in CSHandler)
public void wakeup (pcb process)
Removes a process from semaphore's wait queue

public void block (pcb process)
Places processes wanting to execute their critical section's code in the semaphore's wait queue

cpu Class
Simulates two cpus. Each cpu will have a scheduling algorithm and interprocess method.

IODevice
Has three IO devices, keyboard, mouse, and monitor which are chosen at random to interrupt simulating processes for a randomized amount of cycles.

public boolean shouldInterrupt()
A simulating process will have a 50% of being interrupted during any of its instructions.

pcb Class
Process control block for a process. Updated to include memory of a process, as well as PIDs for child processes.

public string toString()
Prints the pcb for all parent and child processes.

cpu Class
2 cpu objects will be made and be assigned two schedulers (round robin and priority) in Phase 3.

ProcessState enum
Contains the states of a process.