GitHub:

OS_Simulator **PHASE 3**: https://github.com/Brodiejt/CMSC-312-Project/tree/main/OS_Simulator

Please grade the most recent Commit;

Modules added in Phase 3 are highlighted in red

Description:

Basic operating system simulator that generates template programs stored in txt files upon user request and simulates CPU scheduling (Round Robin and Priority Scheduling) on 8 threads for simulation of multiprocessing. After both scheduler have completed all of their jobs, their turnaround times are compared. Additionally, each process is assigned to a certain chunk of contiguous memory inside Main Memory. Interactivity with the Operating System is done with a simulated Command Line Interface where the user can type commands to generate new processes, obtain the status' of processes in execution, and exit the program.

Implementation:

The program assigns the time quantum for the round Robin scheduler to 10 milliseconds and initializes the size of all cues to fit 999 process control objects. Additionally, each process is also assigned a random priority if it is put into the priority queue. To simulate I/O operations, a time limit of 10 milliseconds for each IO operation is set for process control blocks that are in the IO wait queue. The rest of the program is just a simple while loop that executes all programs that have been put in the ready queue, and depending on the exit status of each process they can be put in the IO, the fork generator can be called, or they are terminated and removed from the ready queue. Main Memory is simulated with a Java Object that contains an array of instructions (integers) with pointer to the next available slot in memory. Program also simulates multicore simulation with all 8 threads, and all processes share the same virtual memory.

Compilation and Execution:

In order to compile properly all template files must be generated in "Program" folder."`src\\kernel\\Programs\\"`. All *.java* files must remain where they are. Note: this project was created using eclipse so the class files might also be in the "bin" folder

Usage:

Commands (CLI):

status: Prints the status of all process in execution

quit: Exits the program

g: Generates user process

gtxt: generates a user-defined number of program files (templates)

run: clears memory re-executes all program templates in the Program folder

clean: Deletes all program templates in the “Programs” folder

Class:	Description	Methods + Description
PCB int pc; int pid int[] programStart int programSize int state = 0 static int counter long start; long end priority	used to simulate the process control block and is used to aid the program in context switching between CPU bursts. this class stores basic information about the process including the program counter, process ID, an array of instructions to simulate memory, and start and end times of the process to simulate turn around time. Added priority instance variable to be used in priority queue.	int execute(int , Queue , Queue , Queue , Semaphore) - Function that receives a time quantum three Q objects and a semaphore object to simulate the execution of any given process. this function iterates through the program start array and completes a certain task depending on the opcode at any given array index. This function returns the “exit status” of the process to be used in the main function. void printMemory() - Print the current memory of any given process. int generate_opcode(String) - returns a specific opcode for any given line of code. the 1st digit of the opcode represents a specific keyword operation. -
PCBGenerator	this class is used to generate program templates.	void cleanFolder() - Deletes all text files with a capital F that was created from the generator. void createFile(String path, String program) - Receives a file path and program string that contains several keyword operations Hey text file in path “`src\\kernel\\Programs\\` void generateTXT() - Function that randomizes the number of instructions, the location of critical sections, and type of instructions to create a program template. Each program is only given one critical section and only one fork operation call. PCB forkPCB() - Function used to generate a process control object to be used after a specific process calls fork uses the same implementation as generateTXT; - Modified to allow for multilevel parent child relation ships by having a random chance of

		creating more processes with the fork opcode
Queue	class to be used to create job queue ready queue and in IO queue	Standard queue functions
main	Added priority queue that creates separate ready queue for the cores that use priority scheduling. Simulated with built in JAVA priority que class and comparator Modified system boot to load half of all programs into priority queue for comparisons.	Creates 9 Threads - 1 for Command Line - 8 for CPU processors to be used for scheduling comparisons - 4 for rounds robin and 4 for Priority Scheduling
Semaphore:	Class to be used in phase 3	
Main_Memory - Int[] memory - Int next	Class used to simulate Main Memory of an OS. Uses an array of 125,000 32-bit integers to store instructions (opcodes) and a integer to store the next open block of memory for allocation	loadToMemory(File , PCB) - this function parses each line of a given text file converts that line into a specific opcode and stores it into the instruction memory of the process control block. cleanMemory() - simulates garbage collection
RRCPU	Class changed to extend CPU class	Run() - executes programs
CLI	Command Line Interface class that extends Thread Class. Runs parallel to OS modules, reads input from user, returns info about the OS or creates more processes	Run() - intitializes command line
PCPU	Added class that extends CPU, executes programs with priority scheduling	Run()
CPU	CPU class that extends Thread class in Java.	