**Operating System Programming Assignment**

Chevaughn Gibson 1900396, Gail-Ann Archer 2002407, Rushawn Reid 2003911

Faculty of Engineering and Computing, University of Technology, Jamaica

CIT3002, Operating Systems

Mr. Philip Smith

November 20, 2022

**Documentation**

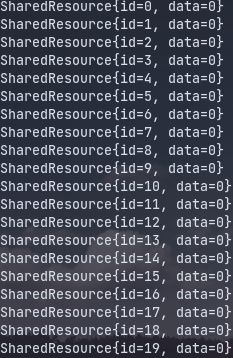
**Team Breakdown Structure**

Gail-Ann Archer

Chevaughn Gibson: Implement scheduling algorithm in Java

Rushawn Reid

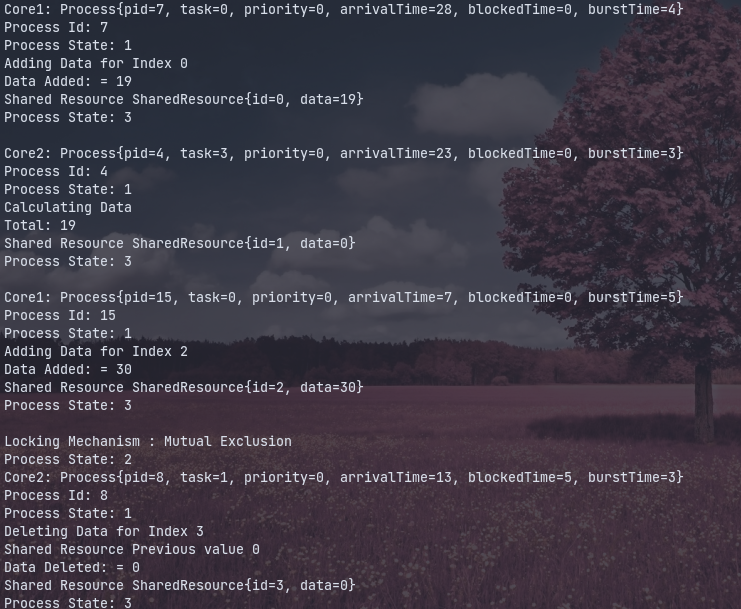
**Screenshots of executing program**

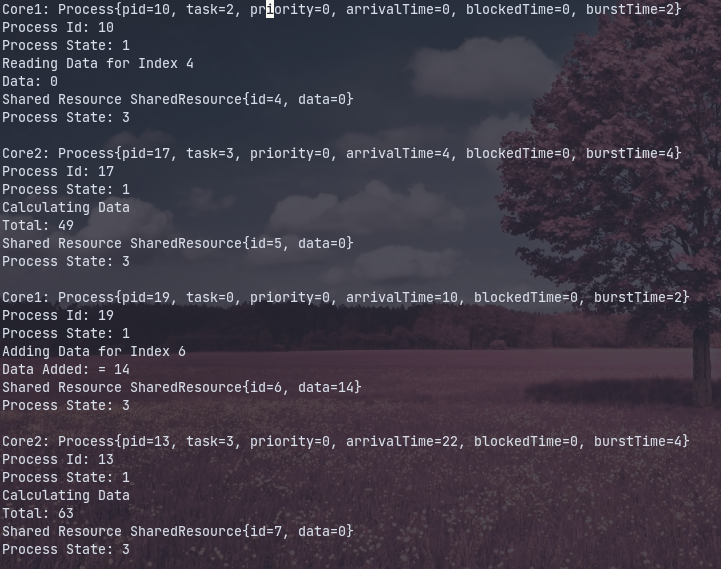


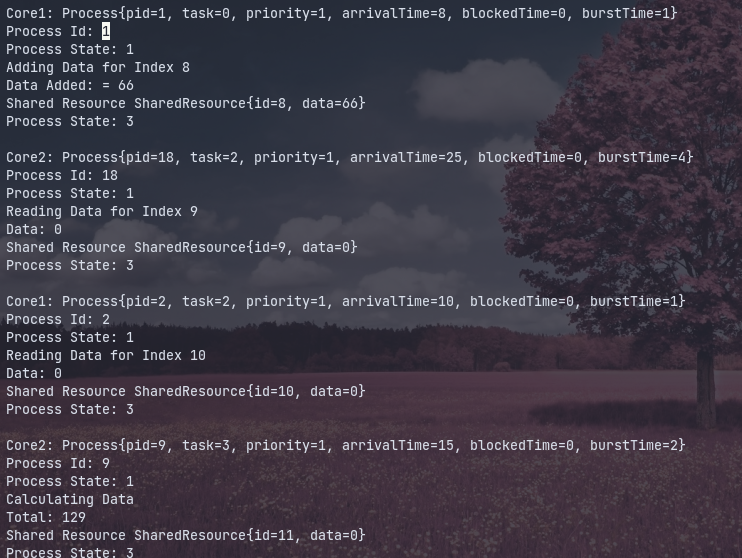
Printing the loaded Shared Resources

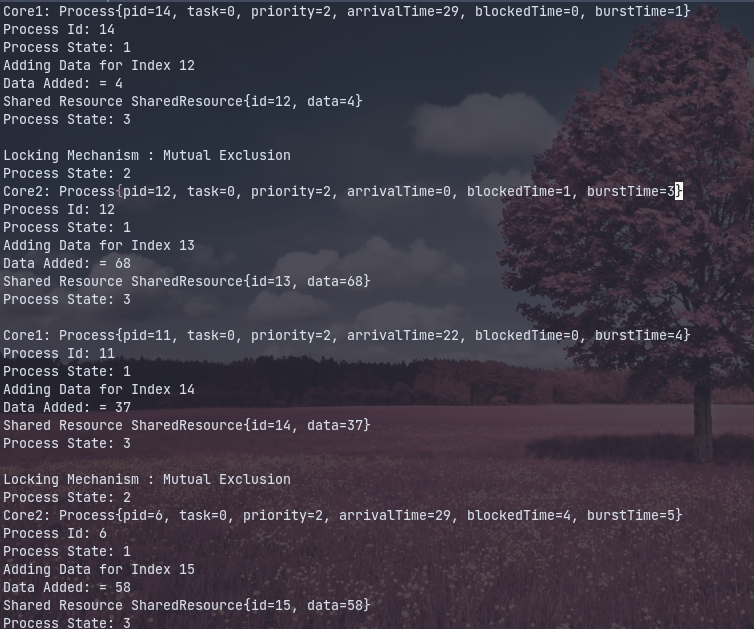
Printing the auto generated processes

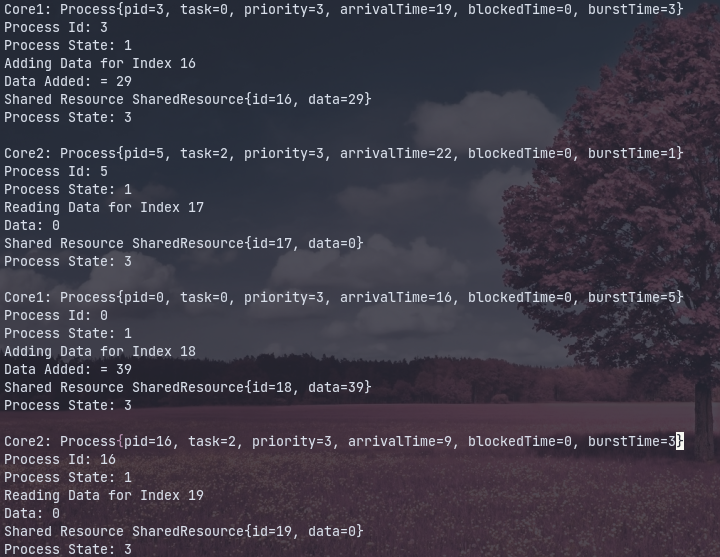
Executing Processes



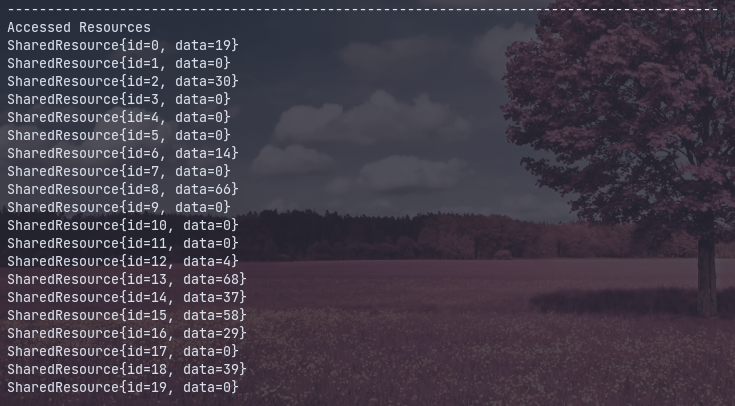




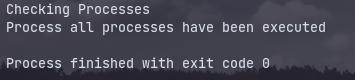




Printing Accessed Resources



Checking if there are remaining processes



**Description of how simulation works**

This program was designed to simulated the priority non preemptive scheduling algorithm.

We have 5 classes: Main, CPU, Process, SharedResource, ProComparator.

Within the Main class, there is generated a Process object called process, a Shared Resource List called srl and a Priority Queue called pq. pq is set to use the attribute priority as the tool to determine the priority of each value within itself. This is done the ProComparator class

Line 16 we call a function that adds 20 SharedResources initalized to 0 and adds them to a list.

Line 17 generates and random processes and loads them into pq.

Line 19 and 20 generates 2 cpu objects core1 and core2 respectively.

Line 22 creates an srlIndex which is the value that will control the index value for srl throughout the execution of the program.

Line 24 creates loop to execute while pq still has processes loaded.

Line 26 checks if pq is empty.

Line 32 and 33 creates 2 processes, process1 and process2. process1 is set to the value at the front of the queue and that value is then removed. The same operation occurs for process2.

Line 37 checks if process1 has a higher priority than process2 if true execute process1 first else execute process2 first.

If the condition is true then core1stages the process by setting the its process attribute to be equal to process1.

Line 40 then calls core1’s executeProcess method which takes a SharedResource List, the srlIndex and process1’s get Task. exectuesProcess returns the updates srlIndex.

**Stepping into Cpu at line 40**

Line 26 creates a int of srlIndex which is initialized to the function parameter index.

Line 27 sets the process to running

next the program check whether the task is either 0,1,2,3 with 0 being Add, 1 Delete, 2 Read, 3 Calculate total data from srl and performs the actions. Add sets a random value to the current srl index, 1 Delete sets the data to 0, 2 Reads the value and 3 Calculates the total.

Line 68 sets the process to terminated

**Stepping out of Cpu to Main**

Line 42 stages the process for core2

Line 43 and 44 checks if should be a lock, this will occur when either process1 task either equals 1 or 0 or process2 either equals 1 or 0.

If true Line 46 sets the process to Blocked

Line 47sets the blockedtime of process2 to the burst time of process1

This simulates the locking of the process due to mutual exclusion.

If line 37 was false then process2 would execute first and perform the same aforementioned action where process1 would be replaced with process2.

**Stepping out of the while loop**

the program prints all values in the srl and pq if there exists any.