==========
CIS3207 Section 04
Assignment 1: Discrete Event Simulator
Author: Garrett Bowser
Due date: 16 September 2019
==========

Overview

This program simulates the way processes are handled by a computer system comprised of a combination of CPU cores and storage disks. The arrivals and departures of these processes to and from each component are abstracted into events. The random arrival and departure of events drive the simulation as they would in a real computer system.

Data Structures

Clock

This program uses an arbitrary time unit to represent relative time spent by a process on each component of the system as well as assign event priority. If there is only a single core in the simulation, the program's global clock is advanced whenever a process completes execution on a CPU. Otherwise, the clock is advanced by one unit per completion of the event-handling loop to simulate parallelism between cores (see Component).

Process

A process is the most basic component of the simulation. A process is represented by a numeric ID, a CPU time, and an IO time. These times represent relative time spent on each respective component type and will change as the process moves through the system.

Event

An event represents a process moving through the system. Events have a type, a process, and a time associated with them. An event’s type determines how it is handled by the event handler. An event's time is used as a priority in the event queue with lower times being processed first.

Component

A component can in this simulation be either a CPU core or a disk. Both components can receive and complete jobs (processes) but only in a system with a single core will time be advanced based on a process' CPU time. This is to simulate parallelism between CPU cores. Similarly, jobs processed on a disk will not advance time based on the process' I/O time as execution does not wait for I/O and disks can run in parallel with one another.

Config

A config is a struct which can read values from the config.ini file distributed with the program and make them accessible to the Simulation. In this manner the parameters and initial conditions of the simulation can be easily modified.

Simulation

A simulation is a struct that allows different parts of the system to interact with one another. The simulation provides event handling and distributes jobs between the components in the simulation. To simulate a constant creation of new processes, a new process is inserted into the event queue after each process is recognized by the system in the form of a system arrival event.

System Overview

The basic structure of the simulation is as follows:

- 1. A process enters the system
    - From here the process is assigned a free core and is dispatched to that core via the EQ (Event Queue)
    - The process is reinserted into the event queue with high priority as a CPU arrival event
- 2. The process arrives, then runs on the CPU for a period of time
    - After processing a CPU arrival event, the system will assign the process a random (configured) CPU time
    - The process is reinserted into the EQ as a CPU finish event with this time as its priority
    - Upon arrival of this CPU finish event,the system will advance time appropriately
- 3. The process has a random (configured) change to exit the system here OR begin IO on a disk
    - If the process exits the system, it is reinserted into the EQ as a system exit event
    - If the process begins I/O, it is reinserted into the EQ as a disk arrival event
    - The process as assigned a random (configured) I/O time between its arrival and completion and sent to the first free disk or the disk with the smallest queue
- 4. After completing I/O, the process is reinserted into the event queue
    - The process is reinserted into the EQ as a system arrival event

Note There are features in this program that are out of the scope of the assignment. They were added out of my own interest and for the sake of working on this project in the future.

Thank you for reading!