Overview

A program that implements a discrete simulator (single-threaded) that implements a five-level multilevel feedback queue.

Introduction

This document holds the technical design of the CSCI312\_A6b\_Schmidt program and serves as a pseudo lessons learned for the changes made to the initial design. The program is intended to simulate a multilevel feedback queue by taking in an input text file and outputting a trace text file.

## Scope

Simulate a five-level multilevel feedback queue.

### PROCESSING

The user enters the program name, an ASCII file address, and a desired name for the soon-to-be created ASCII file.

The program then reads the ASCII file for the processes’ information and loads it into the readyQue LinkedList and turning the accompanying data in m\_Proccess objects. Then the readyQue is turned into an Array and sorted based off arrival time. The readyQueArray is then turned back into a LinkedList.

The program then takes the first available process and puts it the first quantum. If there is more time left after running quantum 1, it places it in quantum two. Once the next process is ready to execute, it is placed in a quantum and the process continues until all processes are executed.

### DATA

The logical and physical data structure of files should be defined in detail.

Data structure definitions must include the:

description of each element, e.g. name, type, dimension;

relationships between the elements, i.e. the structure;

range of possible values of each element;

initial values of each element.

ProcessHeap

ProcessHeap is only the process for managing and running the heap. It holds several LinkedLists<m\_Process> that holds the processes for the heap.

Creating an instance of a ProcessHeap requires input of a StreamReader and a string which it uses to run the simulation. It returns nothing.

m\_Process

Contains the information for the processes: m\_ProccessNumber, m\_ArrivalTime, and m\_ExecutionDuration for use by the ProcessHeap.

### COMPONENTS

Assignment6Execution

* Holds the static main and initiates the simulator.

ProcessHeap

* Holds the simulator process

|  |
| --- |
| **ProcessHeap** |
|  |
| +ProcessHeap(StreamReader, string): constructor  -RunProcesses(m\_Process[], string): void |

m\_Process

* Holds the information for the processes.

|  |
| --- |
| **m\_Process** |
| -m\_ProccessNumber: int  - m\_ArrivalTime: int  - m\_ExecutionDuration: int |
| +m\_Process(int, int, int) : constructor  +setExecutionDuration(int): int  +getExecutionDuration(): int  +getArrivalTime(): int  +getProcessNumber(): int  +CompareTo(m\_Process): int |

### TESTING

Present one or more named scenarios that will be utilized to test the application.

The testing plan should be repeatable.

Describe the scenario in detail, the steps required to execute the test, the input data, the output data, and the success criteria.

Present a summary of the testing scenarios before the details of each scenario.

|  |  |  |
| --- | --- | --- |
| Scenario | Description | Pass/Fail |
| 1st jobs.txt run | Run simulation based off input file | Fail |
| 2nd jobs.txt run | Run simulation based off input file | Pass |
| 3rd jobs.txt run | Run simulation based off input file |  |

Jobs.txt input:

1 4 8

2 9 5

3 10 3

4 1 9

Scenario #1- jobs.txt test

|  |  |  |
| --- | --- | --- |
| Step | Description | Input/Output |
| 1. | Enter input address | Input: .txt file to be simulated |
| 2. | Enter desired file name | Output: .txt file with trace information |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| EXPECTED OUTPUT | | To process in order of 4,1,2,3 |
| ACTUAL OUTPUT | | Program broke because I had a bool backwards for running the StreamReader. |
| RESULTS – The desired output of a .txt file with the findings was produced. | | Fail |

Scenario #2- wap.txt test

|  |  |  |
| --- | --- | --- |
| Step | Description | Input/Output |
| 1. | Enter input address | Input: .txt file to be simulated |
| 2. | Enter desired file name | Output: .txt file with trace information |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| EXPECTED OUTPUT | | To process in order of 4,1,2,3 |
| ACTUAL OUTPUT | | Time: 0204402012334  Proc: 4414122333241 |
| RESULTS – The desired output of a .txt file with the findings was produced. | | Pass(I think) It looks like when it switched to the next que that it added it to the front of the que. |

Scenario #3 – job.txt test

##### 

|  |  |  |
| --- | --- | --- |
| Step | Description | Input/Output |
| 1. | Enter input address | Input: .txt file to be simulated |
| 2. | Enter desired file name | Output: .txt file with trace information |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| EXPECTED OUTPUT | | To process in order of 4,1,2,3 |
| ACTUAL OUTPUT | | Time: 0204402012334  Proc: 4414122333241 |
| RESULTS – The desired output of a .txt file with the findings was produced. | | Pass(I think) It did the same thing twice. It appears to be running the processes the right amount of times. |

##### *Scenario #4 – job.txt test*

|  |  |  |
| --- | --- | --- |
| Step | Description | Input/Output |
| 1. | Enter input address | Input: .txt file to be simulated |
| 2. | Enter desired file name | Output: .txt file with trace information |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| EXPECTED OUTPUT | | To process in order of 4,1,2,3 |
| ACTUAL OUTPUT | | Time: 00110201122244221  Proc: 41412432312341241 |
| RESULTS – The desired output of a .txt file with the findings was produced. | | Pass: It turned out that I had it set so that when it pulled something from the readyque that it was putting it in the quantum that n currently was instead of the first quantum. Now it looks more right. |