Programming-Assignment-1 Report

Gustavo Hammerschmidt
CS433 - Assignment 1

February 11 2020

1 Name and Description of the submitted files:

1.1 LinkedList.cpp

This class is used for implementing a Linked List, which is used for stocking the processes created. It has basic list-operation methods. It has the display and add functions.

1.2 ReadyQueue.cpp

This class inherits the Linked List class and is used to organize processes, which are to be run, according to their priorities. It organizes the process with greater priorities at the beginning of the list, so they can be removed easily only by remove the head, which is always the highest priority node. The node-adding function inserts node at a position at the list in which their priorities are greater than the next nodes.

1.3 Node.cpp

This class is used as a capsule for a process at the same time that it is used as a component in a linked list. It has a pointer to a next node and a process variable that holds the process encapsulated. It can be used as a component of a linked list or a ready queue. On procedures, node pointers are used as form to easily operate over lists and get access to lists' processes.

1.4 PCB.cpp

This class represents a process with its attributes: ID, state and priority. It only has construction and destruction methods and get and set functions to its attributes.

1.5 tester.cpp

This class has methods that contains the first and second tests procedures and the project introduction. The first method is the one responsible for creating the ready-queue and linked-list lists, it inserts processes in nodes and then adds and removes these nodes from both list while it displays these changes. The second class is responsible for creating a list of processes with random priorities that, based on them, inserts or drops the processes with a higher priority, it repeats the process on a execution loop and exhibits the total execution time and the PCB times-added-dropped table. There is a auxiliary function used to check if an array contains an value, it is on the second test function on the random array creation process, use to select processes randomly.

1.6 Main.cpp

This is the main class of the project. It calls only the functions of the tester class. It calls the introduction, the first test, the second test and finishes the program.

2 Compilation of the program

To compile the program, use the Makefile to create the object classes of each .cpp class. I used the makefile file available on the example code on the server, making small changes to it in order to compile the program and to be able to execute it. It was compiled and executed on linux. I solely recommend executing in Linux.

3 Results of the program

This is my first c++ project. I believe that influenced the overall performance of the program, not only have I learned c++ to make the project, but also learn a lot more of c++ techniques and how to deal with memory management and pointers. The first point I would like to make it clear is that the program was

written on visual studio, and that I had problems, when working on the ubuntu virtual machine, with the files and the libraries used. Second thing is that the performance of my program is not the best due to the fact that I had to make two copies of a same process, and had to administrate them both in order to make the program works, because, previously, the same process could not coexist in two different linked list without merging them, even though I cleared the previous pointers. The total execution time stays in-between 80 to 90 seconds. In my point of view, this performance time did not suffer from the duplicated-process effect, but to the need of creating Nodes(the list components) for a few procedures like: adding processes, changing processes' state and removing the pointers of these nodes. I will also remind that any information not explicitly describe here is well-commented on the program's source code. It is good to remember that, as my first c++ project, this code may look pretty simple or general, but I do affirm that I have not made use of any means of code copy or even used other's code. I have not taken any reference on any procedure involved on my code. When executing on Ubuntu Virtual Machine, the program did not executed well due to core segmentation, but when executed on windows it works perfectly.