SIMPLE OPERATING SYSTEM

**Bowie state university**

**Computer science department**

**Graduate student**

**By- Archanaben Patel**

**5/3/2016**

The aim of the project is to implement a simple operating system (SOS) developed in c++. The SOS is expected provide a specified system allowing for processes and memory management

**OBJECTIVE:**

The aim of the project is to implement a simple operating system. This simple operating system will allow process and memory management.

**INTRODUCTION:**

* The Project was implemented in C++ using Win 32 API.
* The program simulates a simple operating system.
* The program includes 10 processes out of which five are system inbuilt processes and rest of them are user defined processes.
* System built processes are the processes that are already existing in the operating system i.e. windows 10
* User defined processes are the set of instructions given by the programmer within the code to perform an operation or to complete a task.

**DESIGN CONSIDERATIONS:**

* SYSTEM REQUIREMENTS
  + A 32 bit windows Operating system
  + Eclipse IDE for ++ Developers.
  + A Java JRE/JDK is required to install Eclipse.
* PROCESS MANAGEMENT:
  + Queues to hold processes in ready and waiting state.
  + Threads to run the processes i.e. Running state.
* MEMORY MANAGEMENT:
  + A RAM of size 132 bytes. This is the limited size on a process to run.

Memory map to determine the memory size required to run the process

**DESIGN:**

* Main function calls process list menu to select process based on user priorities.
* The selected processes enter the ready queue in the same order as user entered.
* As queue’s method of organizing is shortest job first the processes that entered the ready queue will start running in the sorting order of processing time, each at a time.
* From the ready queue, each process will be taken up by the thread to run the process.
* During running state, process operations are performed and when the task is complete, it gets terminated. The thread now is allowed to take the next process from the ready queue. Thus at each time one process is in the running state.
* The program calculates memory size required to the process by looking up in the process memory map.
* The process memory map has a memory map which has a key and memory size mapped to it. Key value is the process id itself, thus through process memory map the memory size for the process is determined.
* Process operations are performed only when process memory size is within the range of RAM i.e. 132 bytes.

**THE FILES INCLUDED IN THE CODE:**

**Main:**

Program execution starts here. Main cpp calls the show list method to select the processes, after the selection they are added in the ready queue. Then it calls the threads to run the processes.

**Queue:**

This header file has all functions related to queue and sorting the element in queue where they needed to short using structure.

**List Processes:**

This file have all data of 10 processes including their process ID, process control block, process number, process name, processing time.

**Process Management:**

This header has thread implementation part of the code. Each time the thread takes a process, the process is removed from the ready queue. The thread now checks the process number and if the process memory is within the RAM size. If the process memory size is within the RAM size then the process is executed, otherwise it does nothing with the process. This will continue till the queue gets empty.

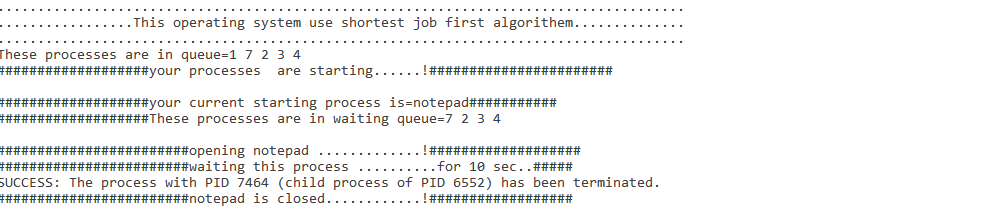
**Process Memory Map:**

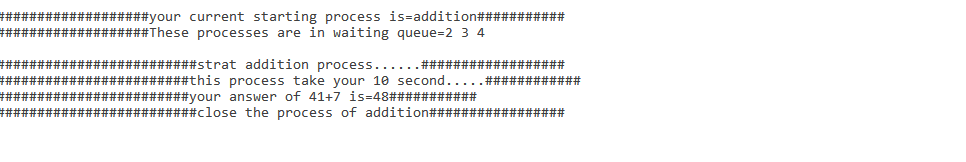
The process memory map has a memory map which has a key and memory size mapped to it. Key value is the process id itself, thus through process memory map the memory size for the process is determined.

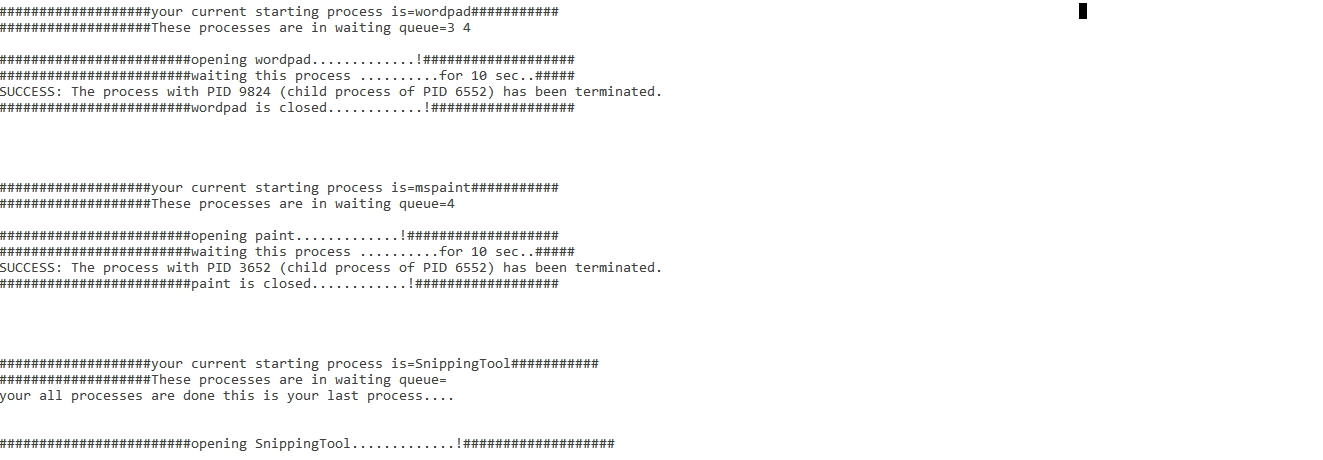
**TEST RESULTS:**

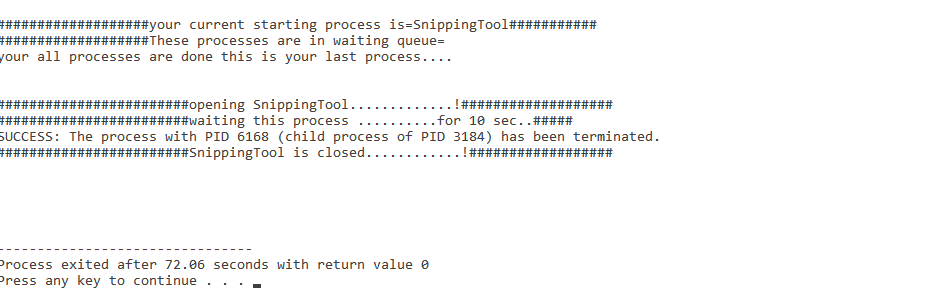
****

****

****

****

****

****