Operating Systems Lab

Final Term Project

Operating System Simulator

Instructor: Waqas Ali

Project Statement

An operating system (OS) is system software that manages computer hardware, software resources, and provides common services for computer programs. In this project you will design a simulation of operating system on your own. The simulation should be made using and following all concepts studied in this course.

Key features the operating system should have:

- Your operating system should have a name of its own that will display on starting the program
  Detail: You should make the program in such a way that when it starts it displays the name of your operating system on terminal using sleep command for some seconds like it is loading.
- The RAM hard drive and the number of cores will be given by the user while starting the OS

```
(base) Mateens-MacBook-Pro:desktop mateencheema$ g++ os.cpp -o OS
(base) Mateens-MacBook-Pro:desktop mateencheema$ ./OS 2 256 8
```

  This instance of OS will have 2GB RAM, 256GB Hard drive and 8 cores you can choose your own convention.

- The resources should be managed in you operating system RAM Hard drive and cores should be kept in mind while making new processes or threads.
- You will define basic applications and tasks that your operating system can perform
  - Basic tasks would be like Notepad (writing into file with auto save function), Calculator, Time, creating a file, Move, Copy, Delete file, Check file info, Minigames (minesweeper etc.)
  - You should have minimum of 15 tasks in your operating system

  Detail: Each task should have a defined task its own separate code file and it should make a new process simple function calling is strictly not allowed the once a process is created it will send the creating process message that will contain memory required in Hard drive and RAM if available then the process

will reply grantee else the process should be terminated each process will have its own terminal so the output is not messed up.

This will help you to enable multi-tasking in your operating system.

- You can also manually create an interruption to stop the specific task or minimize it.
  Detail: Each task should have a button to close the task in between so to make a simulation of interrupt.
- Your operating system should be able to multitask.
- Your operating system would have user mode and kernel mode (in kernel mode you can access hardware resources)

  Detail: In kernel Mode user can close or delete the processes from memory meaning you can close running programs.

Project Working:

You will provide Hardware resources on starting program. The operating system will start and its name will be written on boot screen. After starting some basic tasks like time and calendar should start on their own you can also add other tasks on startup. An instruction guide should show all the tasks a user can perform in your operating system. The user can then select a specific task and it should start working in a new terminal for that you can use EXEC commands. It will take specific location and size in RAM and wait for its turn to execute which will be controlled by you using scheduling techniques if the processes increase the total number of cores you need to schedule the processes. If the user starts any other program that will be added into the RAM too. A ready queue will schedule all the tasks in the RAM on its own and send them for processing. If any interrupt for input in task or some other external interrupt occurs the task should move into blocked section. For that time processor can process some other tasks. After the interrupt for first program ends it can execute again. Each process will execute according to its turnaround time which will be set during coding. User will not provide this on runtime. After task is completed, it should be removed from the RAM. If user saves something it should be saved in the hard disk.

Operating system concepts:

Your project should involve following OS concepts:

- Multitasking
- Context switching
- Resource allocation
- User mode and Kernel mode
- Process creation
- Threads
- EXEC commands
- Scheduling using mutual exclusion, semaphore and condition variable
- Scheduling techniques in ready queue. You should implement multilevel queue with different techniques on each level.

All the above concepts should be used in order to get full marks.

You will get bonus marks for implementing it using graphics library.

Project Full Workflow:

Program starts with asking resources and then operating system loads. Then if you are implementing with graphics or windows form it should have icons for tasks that you can perform. Else in terminal it should have a list of tasks that you can perform. Each task when started should have an option to close it in between or minimize. Closing will close the task and remove it from the RAM and user can select any other task. By minimizing it will stay in the RAM user can select any other task he wants to perform. One option will also be added to select any task that is already in ram and working. So, user can select a task from already running tasks. The running tasks should not exceed the size of RAM.

From the tasks some tasks should involve full time user input like a calculator. Some can run in background. Some should have only processing.

Types of Tasks:

- Full time response like a game. It can only stop by closing or minimizing it. They should occupy space according to their work.
- A background process like a song in background. You can implement it by using a beep sound playing in background for a specific time and it will stop according to its length according to a song. Other process like this will be a copy file task or printing a file

- Some tasks that perform on their own. Like a clock or saving a notepad file automatically

Each task should be able to close or minimize unless it is automatically performed like clock. They can be changed from kernel mode. User can minimize a task anytime and can select from any other task to start and can add more tasks.

Each task will be added in ready queue and multiple tasks can run side by side. Like a game and song. Like writing a notepad file and saving it. Like using calculator and printing a file.

More tasks should be added in the queue and run according to the synchronization.

User can shut down the operating system at any time. It will close all the tasks and close the program. Greeting message can be displayed on booting and shutting down too.

😊😊 Good Luck! 😊😊