Project

Milestone 2 Description

Milestone On Overview

In this milestone, you are asked to create your own scheduler and manage the usage of the resources between the different processes. **You are provided with 3 program files each representing a program. You are asked to create an interpreter that reads the txt files and executes their code. You are asked to implement a memory and save the processes in it. You are also asked to implement mutexes that ensure mutual exclusion over the critical resources. And finally, you are also asked to implement a scheduler that schedules the processes that we have in our system.**

Detailed Description

Programs

We have 3 main Programs:

- Program 1: Given 2 numbers, the program prints the numbers between the 2 given numbers on the screen (inclusive).
- Program 2: Given a filename and data, the program writes the data to the file. Assume that the file doesn't exist and should always be created.
- Program 3: Given a filename, the program prints the contents of the file on the screen.

Process Control Block

A process control block is a data structure used by computer operating systems to store all the information about a process. In order to schedule your processes, you will need to keep a PCB for every process. The PCB should contain the following information:

- Process ID (The process is given an ID when is being created)
- Process State
- Current Priority
- Program Counter
- Memory Boundaries (Lower and Upper Bounds of the process' space in the memory)

Project

Milestone 2 Description

Program Syntax

For the programs, the following syntax is used:

- **print**: to print the output on the screen. Example: **print x**.
- **assign:** to initialize a new variable and assign a value to it. Example: **assign x y**, where **x** is the variable and **y** is the value assigned. The value could be an integer number, or a string. If **y** is **input**, it first prints to the screen "Please enter a value", then the value is taken as an input from the user.
- **writeFile:** to write data to a file. Example: **writeFile x y**, where **x** is the filename and **y** is the data.
- **readFile:** to read data from a file. Example: **readFile x**, where **x** is the filename.
- **printFromTo:** to print all numbers between 2 numbers. Example: **printFromTo x y**, where **x** is the first number, and **y** is the second number.
- **semWait:** to acquire a resource. Example: **semWait x**, where x is the resource name. For more details refer to section **Mutual Exclusion.**
- **semSignal:** to release a resource. Example: **semSignal x**, where x is the resource name. For more details refer to section **Mutual Exclusion**.

Every line of instruction in the program will be executed in 1 clock cycle.

Memory

One of the steps the OS does in order to create a new process is allocating a space for it in the main memory. The OS is responsible for managing the memory and its allocation.

The memory is of a fixed size. It is made up of 60 memory words. The memory is large enough to hold the un-parsed lines of code, variables and PCB for any of the processes. The memory is divided into memory words, each word can store 1 name and its corresponding data; for example: State:" Ready".

A process should only be created at its arrival time. A process is considered created when its program file is read into lines and it gets assigned a part of the memory for instructions, variables and its PCB.

Assume that each process needs enough space for 3 variables.

Project

Milestone 2 Description

Feel free to separate the lines of code, variables and PCB within the memory if needed as long as they fall within the same data structure meant to represent the memory.

Scheduler

A scheduler is responsible for scheduling between the processes in the Ready Queue. It ensures that all processes get a chance to execute. A scheduling algorithm is an algorithm that chooses the process that will be executed next.

In this milestone, you are required to implement:

- First come first serve scheduling algorithm.
- Round Rubin scheduling algorithm with any quantum as a user input.
- The multilevel feedback model scheduling algorithm. The algorithm works as follows, there are 4 levels, 1 being the highest priority level and 4 being the lowest. The quantum for the first level is 1 and it gets doubled as we move down the levels. The policy used in the last level is Round Rubin.

Your task is to implement the 3 scheduling algorithms from scratch and schedule the processes using the implemented scheduling algorithm.

Mutual Exclusion

A mutex is a directive provided by the OS used to control access to a shared resource between processes in a concurrent system such as a multi-programming operating system by using two atomic operations, semwait and semsignal. Mutexes are used to ensure mutual exclusion over the critical section.

You are required to implement 3 mutexes, one for each resource we have:

- Accessing a file, to read or to write.
- Taking user input
- Outputting on the screen.

Whenever a mutex is used, either a semWait or semSignal instruction is followed by the name of the resource, **userInput, userOutput or file**.

For an illustration, to print on the screen:-

- **semWait userOutput:** any process calls it whenever it wants to print something on the screen to acquire the key of the resource.

- **semSignal userOutput:** any process calls it whenever it finishes printing to release the key of the resource.

NOTE: ONLY ONE process is allowed to use the resource at a time. If a process requests the use of a resource while it is being used by another process, it should be blocked and added to the blocked queue of this resource and the general blocked queue.

NOTE: Unblocked processes are unblocked based on the priority. Meaning if 2 processes are waiting for the same resource, the process that has the highest priority would be the one to get unblocked.

NOTE: Whenever a process is unblocked, it will go to the ready queue corresponding to its priority.

GUI Requirements for the Scheduler Simulation

The GUI will allow users to interact with the simulated operating system for managing processes and resources. The interface will display real-time information on the state of processes, memory usage, and the status of various system resources (e.g., user input, file access, and screen output).

Main Features:

1. **Main Dashboard:**
    - **Overview Section**: Displays key information about the system, such as the total number of processes, the current clock cycle, and the active scheduling algorithm.
    - **Process List**: A list showing all the processes in the system, including:
        - Process ID
        - Process State (Ready, Running, Blocked)
        - Current Priority
        - Memory Boundaries
        - Program Counter
    - **Queue Section**: Displays the Ready Queue, Blocking Queue, and the Running Process with details such as:
        - Process ID
        - Current Instruction being executed

Milestone 2 Description

- Time in the queue

2. **Scheduler Control Panel:**
   - **Select Scheduling Algorithm**:
     - Dropdown list to select between **First Come First Serve**, **Round Robin** (with adjustable quantum), and **Multilevel Feedback Queue**.
   - **Start/Stop/Reset Button**:
     - Start the simulation.
     - Stop the simulation.
     - Reset the simulation (clear all processes and memory).
   - **Adjust Quantum**:
     - Input field for the user to set the quantum for Round Robin scheduling.
3. **Resource Management Panel:**
   - **Mutex Status**: Displays the status of the mutexes, showing which process is holding or waiting for resources:
     - **userInput** (e.g., for taking user input)
     - **userOutput** (e.g., for printing to the screen)
     - **file** (e.g., for file access)
   - **Blocked Queue for Resources**: Shows a list of processes waiting for resources and their priorities.
4. **Memory Viewer:**
   - **Memory Visualization**: A grid or list showing the current allocation of the 60 memory words.
5. **Log & Console Panel:**
   - **Execution Log**: A real-time log of executed instructions, showing the current instruction, process ID, and the system's reaction (e.g., which resource is acquired or released).
   - **Event Messages**: Real-time system status messages, such as when a process is blocked, unblocked.
6. **Process Creation and Configuration:**
   - **Add Process**: A button that allows users to load a new process file. This will prompt for a file path and then load the process, displaying its program instructions.

Milestone 2 Description

- **Process Configuration**: Users can set the arrival time for each process before starting the simulation.

7. **Visualized Execution:**
    - **Execution Step-by-Step**: A button for advancing the simulation one step at a time (one clock cycle).
    - **Auto Execution**: A button for continuous execution until all processes are completed or stopped.

Additional Notes:

- Order in which the processes are scheduled are subject to change (User input).
- The memory is shown every clock cycle in a human readable format.
- Real-time Updates: The GUI should be updated after each clock cycle with the latest data, so the user can easily track which process is currently executing, which resources are being used, and how memory is allocated.

Please make sure that the output is readable, and presentable.

Project

Milestone 2 Description

Submission Guidelines

- The deadline for submission is **Thursday 1 May 2025 at 11:59 PM**
- You are requested to submit the following documents: The below deliverables are the ones that will be described in the deliverables section, and below is just examples of the deliverables and the naming convention (video and report is a MUST, we can add extra deliverables which are codes etc)
    1. A 1-min video to demonstrate the working experiment (please narrate and comment on the results)
        - ☐ **name the Video (MS_02_Team_m_Video.mp4)**
    2. The required project description report
        - ☐ **name the report (MS_02_Team_m_Report.pdf)**
    3. The developed C code of the experiment in a single zip folder
        - ☐ **name the Code (MS_02_Team_m_Code.zip)**
- Please upload your milestone deliverables to your drive as a .zip file with the following naming format:

  (Ex.: CSEN602_S25_MS_0**n**_Team_**m**.zip)

  *where **m** is your team number and **n** is the milestone number you are currently submitting.*

- Submit **ONLY** the sharing link through the below form and ***make sure that you give permission to access***
    - https://docs.google.com/forms/d/e/1FAIpQLSdU4xMPGar5QIiA4wKGdwmrJKfwPYug-vpJHOAEK7Ce2WCMjA/viewform?usp=header