Group Project 4: CPU Scheduler

Date: 11:55 pm, November 11th (Monday), 2024
Member's name:
Member 1: Nhi Nguyen ID:030818149
Member 2: An Tran ID:030578689

1. First-Come, First-Served (FCFS) Scheduling (Schedule_fcfs.c)

This is one of the simplest CPU scheduling methods. The program is designed to manage and schedule tasks in the order they arrive using the FCFS strategy. Each task is processed completely before moving to the next task in the list.

- The `Task` structure (assumed to be defined in `task.h`) holds information about each task, including its `name`, `priority`, and `burst` time.
- The `struct node *taskList` is a pointer to the head of a linked list that stores the tasks in the order they are added. The list is managed using functions from `list.h`.
- **`log_error` Function**:
  - Used to print error messages to `stderr` in case of memory allocation failures.
  - Ensures that errors are logged instead of terminating the program abruptly, making the program more robust and easier to debug.
- **`add` Function**:
  - Allocates memory for a new `Task` structure and its `name`.
  - Initializes the task's attributes (name, priority, burst time) and inserts it into the linked list.
- **`pickNextTask` Function**:
  - Returns the first task in the list, adhering to the FCFS scheduling order.
  - If the `taskList` is empty, it returns `NULL`. This ensures that the program does not attempt to process a non-existent task.
- **`schedule` Function**:
  - The main scheduling function that runs tasks in FCFS order until all tasks are completed.
  - Uses a `while` loop to continuously pick the first task, run it, and then remove it from the list.
- The use of `log_error` ensures that memory allocation issues are reported rather than causing abrupt program termination.
- Proper use of `malloc` and `free` to manage dynamic memory is crucial to prevent leaks and ensure efficient memory use.

2. Program Design for Priority Scheduling (Schedule_priority.c)

In this scheduling method, tasks are executed based on their priority, with higher-priority tasks being selected for execution before lower-priority ones. Let's discuss the design and flow of the program, focusing on the main components and how they work together to achieve priority scheduling.

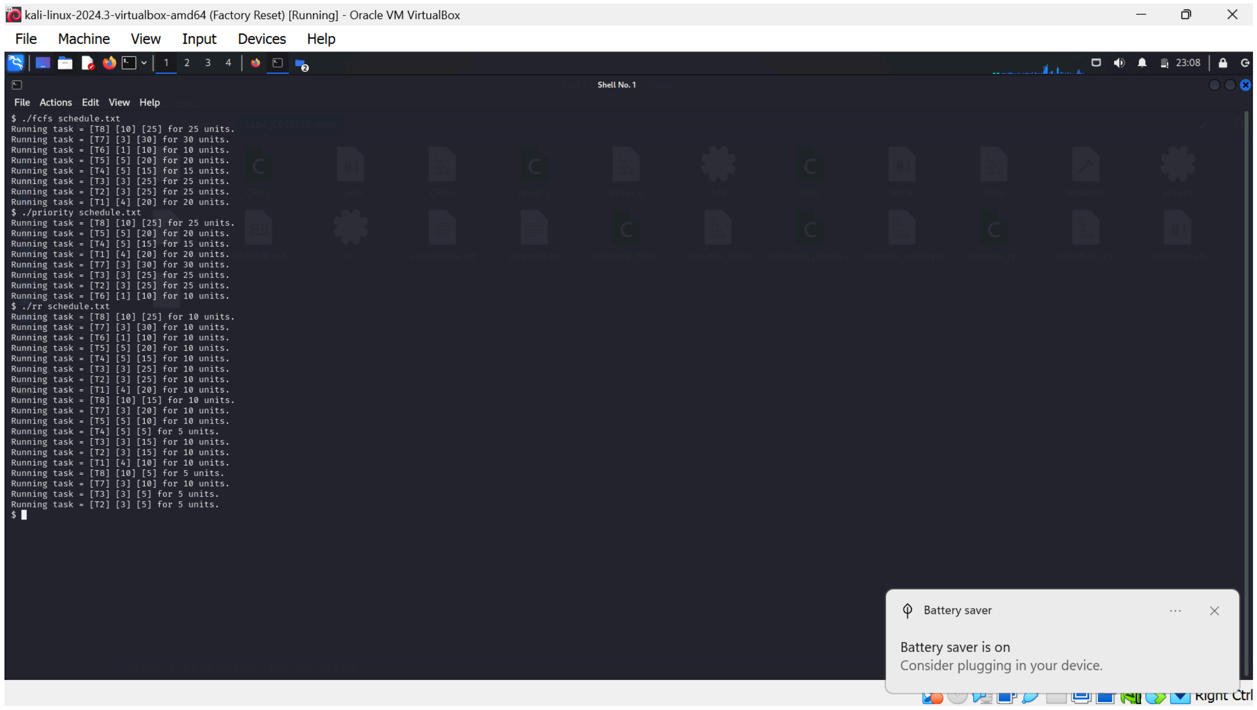
- **Task Structure**:
  - The `Task` structure (assumed to be defined in `task.h`) holds information about each task, including `name`, `priority`, and `burst` time.
- **Linked List**:
  - `struct node *taskList`: A pointer to the head of a linked list that stores the tasks. The list is used to keep track of all the tasks that are waiting
- **`log_error` Function**:
  - Used to print error messages to `stderr` in case of memory allocation failures.
  - Helps ensure that errors are logged instead of causing the program to crash, making debugging easier.
- **`add` Function**:
  - Dynamically allocates memory for a new `Task` and its `name`.
  - Initializes the task's attributes and inserts it into `taskList` using the `insert` function from `list.h`.
- **`pickNextTask` Function**:
  - Selects the task with the highest priority from `taskList`.
  - Iterates through the entire linked list to find the task with the maximum priority value. If multiple tasks have the same highest priority, the first one encountered is selected.
  - **Edge Cases**: Returns `NULL` if `taskList` is empty, preventing further operations on an empty list.
- **`schedule` Function**:
  - The main scheduling function that repeatedly selects the highest-priority task, runs it, and removes it from the list.
  - Uses a `while` loop to continue processing tasks until `taskList` is empty.
  - **Memory Management**: After each task is executed, the memory allocated for the task's `name` and the `Task` structure is freed to prevent memory leaks.

3. Round-Robin Scheduling (Schedule_rr.c)

The code is designed to manage and schedule tasks using the round-robin approach. The main goal is to allocate CPU time fairly among all tasks, preventing any single task from monopolizing the CPU.

- **Task Structure**:
    - The `Task` structure (assumed to be defined in `task.h`) contains details such as `name`, `priority`, and `burst` time.
- **Linked List**:
    - **`struct node *taskList`**: A pointer to the head of a linked list that holds the tasks to be executed.
    - **`struct node *next_node`**: A pointer used to keep track of the next task to be executed in the round-robin cycle.
- **`log_error` Function**:
    - Logs error messages to `stderr` in case of memory allocation failures. This prevents the program from crashing unexpectedly and provides helpful debugging information.
- **`add` Function**:
    - Allocates memory for a new `Task` and its `name`, and initializes the task with the given `priority` and `burst` time.
    - Inserts the new task into `taskList` using the `insert` function from `list.h`.
    - **Memory Management**: If memory allocation fails, the function logs an error and exits gracefully without causing a crash.
- **`pickNextTask` Function**:
    - Implements the round-robin logic by selecting the next task in the list.
    - If `next_node` points to a task with a `next` node, it advances to the next task. Otherwise, it wraps around to the beginning of the list, maintaining the circular nature of round-robin scheduling.
    - Returns the selected task for execution.
- **`schedule` Function**:
    - The main function that repeatedly selects tasks and executes them in round-robin order until all tasks are completed.
    - **Time Quantum Handling**: The time quantum (`QUANTUM`) is defined in `cpu.h` and is used to determine how long each task should run. The function compares `QUANTUM` with the task's `burst` time and uses the smaller value as the time slice.
    - **Task Execution**: The `run` function (from `cpu.h`) simulates running the task for the calculated time slice.
    - **Task Completion**: If a task's burst time becomes zero or less, the task is removed from `taskList` and its allocated memory is freed, ensuring efficient memory management.

Output:

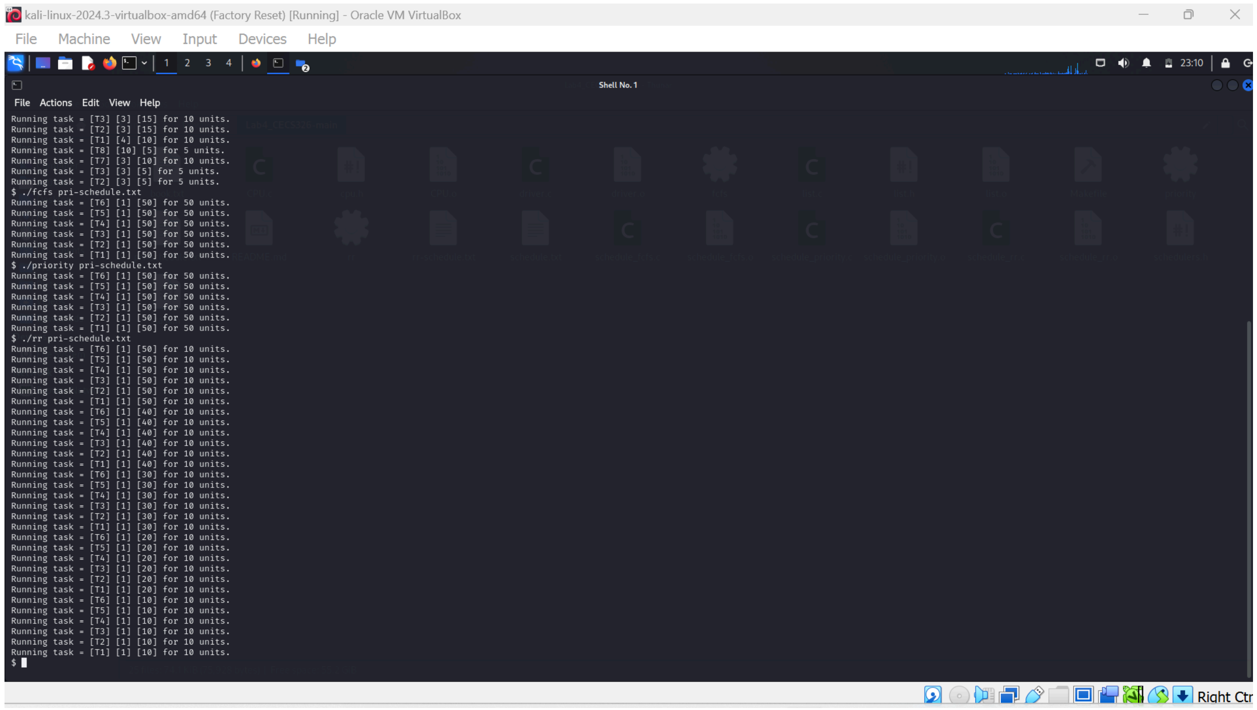


kali-linux-2024.3-virtualbox-amd64 (Factory Reset) [Running] - Oracle VM VirtualBox

```
$ ./fcfs schedule.txt
Running task = [T8] [10] [25] for 25 units.
Running task = [T7] [3] [30] for 30 units.
Running task = [T6] [1] [10] for 10 units.
Running task = [T5] [5] [20] for 20 units.
Running task = [T4] [5] [15] for 15 units.
Running task = [T3] [3] [25] for 25 units.
Running task = [T2] [3] [25] for 25 units.
Running task = [T1] [4] [20] for 20 units.
$ ./priority schedule.txt
Running task = [T8] [10] [25] for 25 units.
Running task = [T5] [5] [20] for 20 units.
Running task = [T4] [5] [15] for 15 units.
Running task = [T1] [4] [20] for 20 units.
Running task = [T7] [3] [30] for 30 units.
Running task = [T3] [3] [25] for 25 units.
Running task = [T2] [3] [25] for 25 units.
Running task = [T6] [1] [10] for 10 units.
$ ./rr schedule.txt
Running task = [T8] [10] [25] for 10 units.
Running task = [T7] [3] [30] for 10 units.
Running task = [T6] [1] [10] for 10 units.
Running task = [T5] [5] [20] for 10 units.
Running task = [T4] [5] [15] for 10 units.
Running task = [T3] [3] [25] for 10 units.
Running task = [T2] [3] [25] for 10 units.
Running task = [T1] [4] [20] for 10 units.
Running task = [T8] [10] [15] for 10 units.
Running task = [T7] [3] [20] for 10 units.
Running task = [T5] [5] [10] for 10 units.
Running task = [T4] [5] [5] for 5 units.
Running task = [T3] [3] [15] for 10 units.
Running task = [T2] [3] [15] for 10 units.
Running task = [T1] [4] [10] for 10 units.
Running task = [T8] [10] [5] for 5 units.
Running task = [T7] [3] [10] for 10 units.
Running task = [T3] [3] [5] for 5 units.
Running task = [T2] [3] [5] for 5 units.
$
```

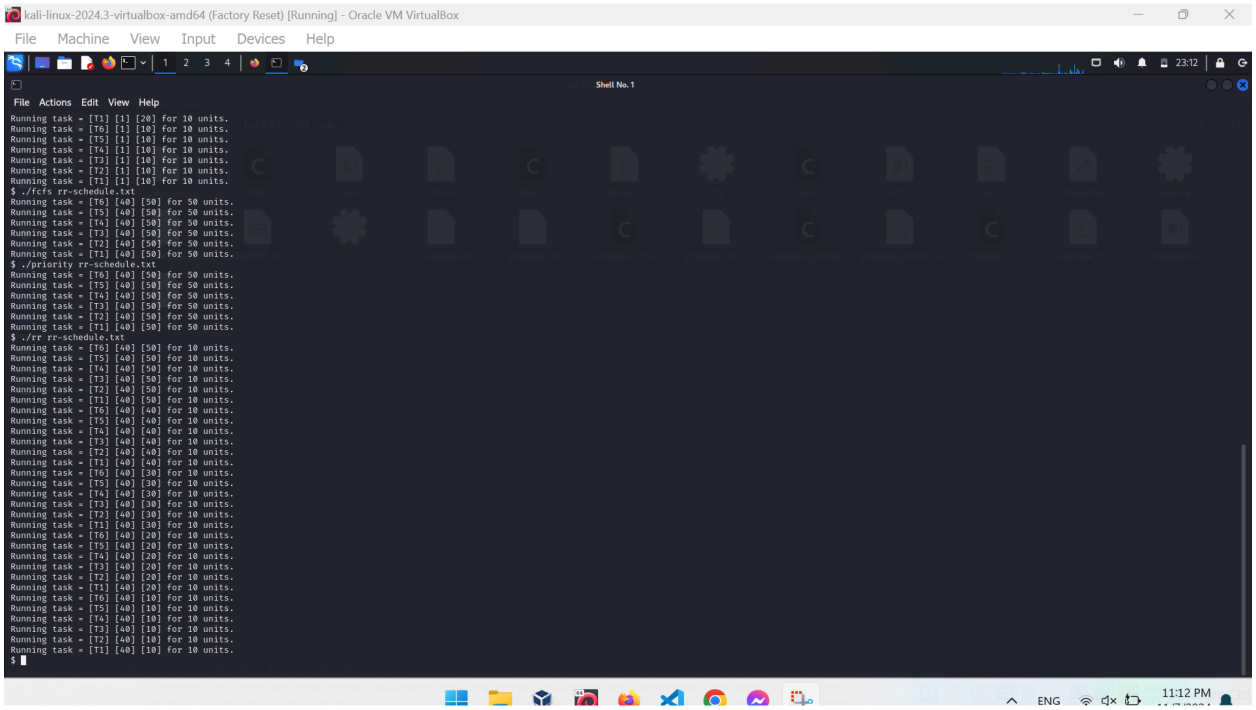
Battery saver

Battery saver is on
Consider plugging in your device.



kali-linux-2024.3-virtualbox-amd64 (Factory Reset) [Running] - Oracle VM VirtualBox

```
Running task = [T3] [3] [15] for 10 units.
Running task = [T2] [3] [15] for 10 units.
Running task = [T1] [4] [10] for 10 units.
Running task = [T8] [10] [5] for 5 units.
Running task = [T7] [3] [10] for 10 units.
Running task = [T3] [3] [5] for 5 units.
Running task = [T2] [3] [5] for 5 units.
$ ./fcfs pri-schedule.txt
Running task = [T6] [1] [50] for 50 units.
Running task = [T5] [1] [50] for 50 units.
Running task = [T4] [1] [50] for 50 units.
Running task = [T3] [1] [50] for 50 units.
Running task = [T2] [1] [50] for 50 units.
Running task = [T1] [1] [50] for 50 units.
$ ./priority pri-schedule.txt
Running task = [T6] [1] [50] for 50 units.
Running task = [T5] [1] [50] for 50 units.
Running task = [T4] [1] [50] for 50 units.
Running task = [T3] [1] [50] for 50 units.
Running task = [T2] [1] [50] for 50 units.
Running task = [T1] [1] [50] for 50 units.
$ ./rr pri-schedule.txt
Running task = [T6] [1] [50] for 10 units.
Running task = [T5] [1] [50] for 10 units.
Running task = [T4] [1] [50] for 10 units.
Running task = [T3] [1] [50] for 10 units.
Running task = [T2] [1] [50] for 10 units.
Running task = [T1] [1] [50] for 10 units.
Running task = [T6] [1] [40] for 10 units.
Running task = [T5] [1] [40] for 10 units.
Running task = [T4] [1] [40] for 10 units.
Running task = [T3] [1] [40] for 10 units.
Running task = [T2] [1] [40] for 10 units.
Running task = [T1] [1] [40] for 10 units.
Running task = [T6] [1] [30] for 10 units.
Running task = [T5] [1] [30] for 10 units.
Running task = [T4] [1] [30] for 10 units.
Running task = [T3] [1] [30] for 10 units.
Running task = [T2] [1] [30] for 10 units.
Running task = [T1] [1] [30] for 10 units.
Running task = [T6] [1] [20] for 10 units.
Running task = [T5] [1] [20] for 10 units.
Running task = [T4] [1] [20] for 10 units.
Running task = [T3] [1] [20] for 10 units.
Running task = [T2] [1] [20] for 10 units.
Running task = [T1] [1] [20] for 10 units.
Running task = [T6] [1] [10] for 10 units.
Running task = [T5] [1] [10] for 10 units.
Running task = [T4] [1] [10] for 10 units.
Running task = [T3] [1] [10] for 10 units.
Running task = [T2] [1] [10] for 10 units.
Running task = [T1] [1] [10] for 10 units.
$
```


kali-linux-2024.3-virtualbox-amd64 (Factory Reset) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Shell No. 1
File Actions Edit View Help

```
Running task = [T1] [1] [20] for 10 units.
Running task = [T6] [1] [10] for 10 units.
Running task = [T5] [1] [10] for 10 units.
Running task = [T4] [1] [10] for 10 units.
Running task = [T3] [1] [10] for 10 units.
Running task = [T2] [1] [10] for 10 units.
Running task = [T1] [1] [10] for 10 units.
$ ./fcfs rr-schedule.txt
Running task = [T6] [40] [50] for 50 units.
Running task = [T5] [40] [50] for 50 units.
Running task = [T4] [40] [50] for 50 units.
Running task = [T3] [40] [50] for 50 units.
Running task = [T2] [40] [50] for 50 units.
Running task = [T1] [40] [50] for 50 units.
$ ./priority rr-schedule.txt
Running task = [T6] [40] [50] for 50 units.
Running task = [T5] [40] [50] for 50 units.
Running task = [T4] [40] [50] for 50 units.
Running task = [T3] [40] [50] for 50 units.
Running task = [T2] [40] [50] for 50 units.
Running task = [T1] [40] [50] for 50 units.
$ ./rr rr-schedule.txt
Running task = [T6] [40] [50] for 10 units.
Running task = [T5] [40] [50] for 10 units.
Running task = [T4] [40] [50] for 10 units.
Running task = [T3] [40] [50] for 10 units.
Running task = [T2] [40] [50] for 10 units.
Running task = [T1] [40] [50] for 10 units.
Running task = [T6] [40] [40] for 10 units.
Running task = [T5] [40] [40] for 10 units.
Running task = [T4] [40] [40] for 10 units.
Running task = [T3] [40] [40] for 10 units.
Running task = [T2] [40] [40] for 10 units.
Running task = [T1] [40] [40] for 10 units.
Running task = [T6] [40] [30] for 10 units.
Running task = [T5] [40] [30] for 10 units.
Running task = [T4] [40] [30] for 10 units.
Running task = [T3] [40] [30] for 10 units.
Running task = [T2] [40] [30] for 10 units.
Running task = [T1] [40] [30] for 10 units.
Running task = [T6] [40] [20] for 10 units.
Running task = [T5] [40] [20] for 10 units.
Running task = [T4] [40] [20] for 10 units.
Running task = [T3] [40] [20] for 10 units.
Running task = [T2] [40] [20] for 10 units.
Running task = [T1] [40] [20] for 10 units.
Running task = [T6] [40] [10] for 10 units.
Running task = [T5] [40] [10] for 10 units.
Running task = [T4] [40] [10] for 10 units.
Running task = [T3] [40] [10] for 10 units.
Running task = [T2] [40] [10] for 10 units.
Running task = [T1] [40] [10] for 10 units.
$
```

Contribution:

schedule_fcfs.c	Nhi
schedule_priority.c	An
schedule_rr.c	An
Readme	Nhi
Report	Nhi
Video	An
Submit + Organize	Nhi

Nhi: 100%
An: 100%