Real-Time Scheduling Algorithms Implementation Report

Mustafa Can Çalışkan, 150200097

November 10, 2024

1 Introduction

This report presents the implementation of four different real-time scheduling algorithms: Least Laxity First (LLF), Earliest Deadline First (EDF), Rate Monotonic (RM), and Deadline Monotonic (DM). The source code organization, compilation process, and usage instructions are provided to help users understand and run the project.

2 Project Structure

The project is organized into the following directories and files:

- **include/**: Contains header files defining the functions and data structures used by the scheduling algorithms. These files include:
  - `dm.h, edf.h, llf.h, rm.h, task.h, utils.h`
- **src/**: Contains the C source files for the scheduling algorithms and other functionalities:
  - `dm.c, edf.c, llf.c, rm.c`: Implementation files for the scheduling algorithms.
  - `main.c`: Entry point for the program.
  - `utils.c`: Functions related to utility functions.
- `Makefile`: Contains rules for building the project.
- `tasks.csv`: Contains task information used as input for the schedulers.

3 CSV File Format

The `tasks.csv` file defines the tasks to be scheduled by the implemented algorithms. The headers of the CSV file are as follows:

- **TaskName**: Unique identifier for each task.
- **ReleaseTime**: The time at which the task becomes available for execution.
- **Period**: The time interval at which the task repeats.
- **ExecutionTime**: Amount of time required to complete the task.
- **Deadline**: Time by which the task must be completed.

Each row in the CSV file represents a task with the above attributes, and the schedulers use this information to determine the optimal scheduling order.

4 Main Program Behavior

The `main.c` file serves as the entry point for the program. The main function performs the following steps:

1. Reads the task definitions from the `tasks.csv` file using the `read_csv()` function.
2. Displays the tasks using the `show_tasks()` function.
3. Sets a predefined simulation time.
4. Selects a scheduling algorithm.
5. Calls the selected scheduling algorithm with the tasks, task count, and simulation time as parameters.

This modular approach allows for easy switching between different scheduling algorithms by changing the assignment of the `selected_algorithm` function pointer.

5 Compilation and Execution

The project can be compiled and executed using the provided `Makefile`. The steps are outlined below:

5.1 Build Process

To compile the project, navigate to the project directory and run the following command in the terminal:

```
make build
```

This command will create the `build/` and `object/` directories, compile the source files into object files, and link them to create an executable named `build/scheduler`.

5.2 Running the Scheduler

After successfully compiling the project, you can run the scheduler using:

```
make run
```

This will execute the scheduling algorithms using the tasks defined in the `tasks.csv` file. The results, including the order in which tasks are scheduled, are displayed in the terminal.

5.3 Cleaning the Build

To clean up the build files, use the command:

```
make clean
```

This command removes the object files and the executable, as well as the `build/` and `object/` directories.

6 Conclusion

This project demonstrates the implementation of four widely used real-time scheduling algorithms: LLF, EDF, RM, and DM.