README

Overview

This project implements a task scheduling system using a directed acyclic graph (DAG) representation. Tasks have dependencies, durations, and are executed based on their dependencies and earliest start times. The system employs multithreading for task execution.

Files

1. **Graph.h**
    - Defines the structure for a directed graph.
2. **Task.h**
    - Defines the structure for a task and functions to create and execute tasks.
3. **TopoSort_Dynamique.h**
    - Defines functions for topological sorting, calculating earliest start times, and dynamic thread allocation.
4. **Graph.c**
    - Implementation of functions related to the graph structure.
    - Includes functions for initializing and freeing graph memory.
5. **Task.c**
    - Implementation of functions related to task creation and execution.
    - Functions for creating tasks, adding tasks to the graph, and executing tasks in a multithreaded environment.
6. **TopoSort_Dynamique.c**
    - Implementation of functions for topological sorting, earliest start time calculation, and dynamic thread allocation.
7. **Main.c**

- The main program that initializes tasks, performs topological sorting, calculates earliest start times, dynamically allocates threads, executes tasks, and prints the results.
- Main function demonstrating the usage of the implemented system.

How to Use

1. Compile the code using a C compiler, for example:

   ```
   gcc Main.c -o task_scheduler
   ```

2. Run the executable:

   ```
   ./task_scheduler
   ```

3. Adjust the parameters in the `main` function of **Main.c** to customize the number of tasks, threads, and other properties.

Dependencies

- This program uses the pthread library for multithreading.