CMPUT 379: Assignment 4, a4w22 Celina Sheng
LEC B1 Winter 2022 ccid: csheng2

Objectives:

In CMPUT 379, we learned about multithreading, synchronization, and deadlocks. Threads are an execution of a sequence of program instructions; their purpose serves to execute a program concurrently.

In this assignment, we hope to write a program that can deal with the Diner Philosopher's problem. We want to execute jobs concurrently using threads.

Design Overview:

There are two main structures created for this assignment. There is the Resource structure, which contains the resource name and number of resources available. And there is the Job structure, which contains the job name, busy time, idle time, a list of resources, a counter for the number of times it has ran, and a counter for the total time it spent waiting.

Overview of the files:
main.cpp:

- There are several global variables shared in the file. These include:
  - A list called RESOURCES, which contains all the resources
  - A list called JOBS, which containing all the jobs
- First I process the input file.
  - During parsing, I grow the RESOURCES and JOBS lists.

Project Status:

The project is incomplete. So far I have parsed the file and I created pthreads, but I have not handled taking the resources.

I made the assumption that each value in a line is delimited by a single space.

While I am unable to complete the assignment, I hope I can explain my thought process to show my understanding:

1. After parsing the file, we want to iterate through every job.
2. I use 1 binary semaphore. Only one job can access the critical section (ie, manipulate the RESOURCES list) at a time.
3. A pthread to execute the resource handling is done for each job. The job checks if all of its resources are available from RESOURCES.
   a. If available, the job consumes the resources
   b. Otherwise, the job waits
4. Repeat step 3 NITER times

Testing and Results:

I created a file called "example.txt" which contains the input from the assignment:

```
# An instance of the Dining Philosophers Problem with 5 people
#
resources A:1 B:1 C:1 D:1 E:1
job j1 50 100 A:1 B:1
```

```
job j2 50 100 B:1 C:1
job j3 50 100 C:1 D:1
job j4 50 100 D:1 E:1
job j5 50 100 E:1 A:1
```

The program runs the jobs in order and skips jobs that can't run because the resource is consumed.

Acknowledgments:

"Introduction to semaphores in C" by CodeVault
(https://www.youtube.com/watch?v=YSn8_XdGH7c&t=211s)