Thread Calc

Cyber Solutions Development - Tactical

June 7, 2023

Abstract

This is a mini project that is designed to extend the functionality of your FileCalc submission. It will operate in exactly the same way as FileCalc, but will utilize a threadpool for parallel job processing.

1 Requirements

In this assignment, you will extend the functionality of FileCalc with a threadpool. This should require little to no modification of a file parser, it is suggested that you use a Queue or List data structure.

1.1 Basic Requirements

1. Written in C
2. Everything FileCalc does
3. Utilize a Threadpool to process work concurrently
4. Output files in the same way FileCalc does

1.2 Specific Requirements

1. Utilize a data structure to manage work tasks

   It is recommended that you use a data structure from DataStructures1
2. Utilize a Threadpool to process work from the data structure
3. Utilize getopt to process the following arguments

   -n NUM (number of threads in the pool; defaults to 4)

1.2.1 Memory Management

Your program should not be leaking memory, and should show no memory leaks with:

```
valgrind --leak-check=full ./threadcalc <in_dir> <out_dir> <-n optional
    thread num>
```

1.2.2 Thread Management

You must implement a well designed and fully functional threadpool library. Your program should not have deadlocks, race conditions, or odd failure states. Your program should close appropriately if the user stops it with CTRL-C, and not hang waiting for IO. Your program should exit cleanly after calculating all files in the directory.

Your program should show no errors with the helgrind tool from valgrind:

```
valgrind --tool=helgrind ./threadcalc <in_dir> <out_dir> <-n optional
    thread num>
```

It is possible to have erroneous output on C standard library atomic variables. You should throughly investigate error output and make a reasonable determinination if the bug reported on an atomic variable is actually present.

1.2.3 Assumptions

1. All numbers in EQU spec are little endian
2. All valid files will have a valid Magic Number
3. All Equation IDs are unique (they are unique enough)

1.2.4 FileFormat

1. No Changes from FileCalc - Refer to FileSpec

1.3 File Reading Guidelines

1. If you encounter Errors in the Equ Header (ie invalid magic):

   Log an Error to stderr

   Continue Parsing other files

2. If you encounter Errors in Individual Equations:

   Use the Flags field to indicate it was unsolved

   Continue Parsing other Equations

1.3.1 Project Design

You should design your ThreadCalc project with two major subsections:

1. A generic threadpool library

   Can execute different tasks, is multipurpose

   Does not expose pthread library calls to the threadpool library user

2. An executable ThreadCalc

   Accomplishes the items identified in section 1, Requirements

2 Deliverables

Your code should have the following file structure:

```
4_ThreadCalc
├── src
│   └── source-files
├── include
│   └── header-files
├── references
│   └── documentation
└── CMakeLists.txt
```

Your code should build and compile with the following shell script ran from the ThreadCalc directory:

```
// build.sh
mkdir build
cd build
cmake ..
make
```

The build script has the same requirements as the SimpleCalc build script. You must build the threadpool into library called "threadpool".

3 Notes to grader

The purpose of this project is to bridge the gap between FileCalc and NetCalc. Throwing in threadpools and networking in one project was a little much. This is an opportunity for trainees to design and implement generic and multipurpose code for their threadpool that is useful outside the CalcProjects.

Although you ultimately control the trainees JQR completion status, allow them to complete the projects using whichever means they want (within reason). Engineers need to be able to break down problems, and iterate on solutions. If we say "you will do X and Y and Z to solve this" we risk putting the trainees in a box where they can't fully flex their problem solving muscles.

If the trainee uses C standard library atomic variables, it is possible for false errors to be reported with the helgrind tool. After conducting thourough review, you should make the determination as to the validity of the error and need for correction.

4 JQR Sections Potentially Covered - Mentor/Trainee dependent

By this point, your trainee should absolutely be able to point to specific JQR line items, and where they are in the codebase.

1. These projects are designed to give each trainee room to think creatively. Although the reference specification may target specific JQR requirements, it is up to the trainee to pick which JQR items they want to solve, and it should be indicated somewhere in the trainees documentation. It is up to the mentor to determine which JQRs have been satisfied and which need more work.

2. These JQRs are satisfied in the reference solution, which is designed to be a sane and straightforward solve, but not particularly creative.

- 4.1.16
- 4.1.19
- 4.1.21
- 4.1.23
- 4.1.25
- 4.5.2
- 4.5.4
- 4.6.2
- 4.8.1
- 4.8.2