File Calc - File Specifications

Cyber Solutions Development - Tactical

September 23, 2021

Abstract

Your task is to build file parser that conforms to the attached format and solve the equations, given two directories that contain 'M' files and with each file containing 'N' equations.

1 Requirements

In this assignment, you will build an application that will compute equations that are read in from a binary file.

1.1 Basic Requirements

1. Written in C
2. Take two arguments that are directories: Input and Output directory. Report error if not directory.
3. Successfully parses the binary file. Report error otherwise.
4. Solve eqauations and output solved files in the output_dir in the format described in the FileSpec
5. Handle bad inputs such as bad format as described above, divide by zero, or interger overflow. Report error otherwise.
6. Single binary with the usage statement ./simplecalc input_dir output_dir
7. All files should have the following memory permissions: -,rw-, r--, r--. Report error otherwise.
8. Use the read, write, lseek, open, close, and creat system calls. See man pages for usages.

1.2 Specific Requirements

1.2.1 Required Operators

1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Modulo
6. Shift Left
7. Shift Right
8. AND
9. OR
10. XOR
11. Rotate Left
12. Rotate Right

1.2.2 Data Structures

It is recomended to use an array to hold the data for each file, and dynamic memory allocation on the heap to hold all file data. However, you can solve this problem how you see fit.

1.2.3 Memory Management

However you chose to solve it, your program should not be leaking memory. Your program should show no memory leaks with:

```
valgrind --leak-check=full ./filecalc input_dir output_dir
```

1.2.4 Assumptions

1. All numbers are little endian
2. All numbers are 64-bits in size

   It is not save to assume the signedness of the numbers; refer to the Spec
3. All valid files will have a valid Magic Number
4. All Equation IDs are unique (they are unique enough)

1.2.5 FileFormat

1. Please see the attached FileSpec.pdf file.

1.2.6 Special Requirements

1. Utilize the C preprocessor (or some other means) to write debug output when specified.

1.3 File Reading Guidlines

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

   Log an Error to stderr

   Move the file to a BadFiles directory

   Continue Parsing other files

2. If you encounter Errors in Individual Equations:

   Use the Error field to indicate it

   Continue Parsing other Equations

2 Deliverables

Your code should have the following file structure:

```
2_FileCalc
├── src
│   └── source-files
├── hdrs
│   └── header-files
├── docs
│   └── documentation
└── CMakeLists.txt
```

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

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

The build script has the same requirements as the SimpleCalc build script.

3 Notes to grader

The purpose of this assignment is to start writing complex C code. Use this assignment to achieve the following objectives:

1. Perform file input and output operations using system calls
2. Building on CMake knowledge. Have your mentee experiment with using debug macros and building debug/release builds
3. Code organization. This is complex project, and code organization will be key. Ensure your mentee is leverging good practices for code organization.
4. Memory Management: Your mentee should be utilizing the heap for this project. Make sure your mentee understands valgrind and its purpose, and the important of clean memory management.

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.

/pagebreak

4 JQR Sections Potentially Covered - Mentor/Trainee dependent

NOTE: These projects are designed to give each trainee room to think creatively. Although the reference spec 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.

NOTE: Thse JQRs are satisfied in the reference solution, which is designed to be a sane and straightforward, but not particularly creative.

- 4.1.5 (all)
- 4.1.7 (all)
- 4.1.8 (all)
- 4.1.9 (all)
- 4.1.10
- 4.1.11 (all)
- 4.1.12 (all)

- 4.1.18 (all)
- 4.1.19
- 4.1.20 (all)
- 4.1.21
- 4.1.22
- 4.1.23