File Calc

Cyber Solutions Development - Georgia

September 3, 2020

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. Handle bad inputs such as bad format as described above, divide by zero, or interger overflow. Report error otherwise.
5. Single binary with the usage statement ./simplecalc input_dir output_dir
6. All files should have the following memory permissions: -,rw-, r--, r--. Report error otherwise.
7. 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. Left shift
7. Right shift
8. And
9. Or
10. XOR
11. Rotate Left
12. Rotate Right

1.2.2 Data Structures

Use an array to hold the data for each file. Use dynamic memory allocation on the heap to hold all file data.

1.2.3 Memory Management

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
3. All numbers should be treated as signed (int64_t)

2 Deliverables

Your code should have the following file structure:

```
FileCalc
├── src
│   └── source-files
├── hdrs
│   └── file-calc.h
├── 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
```

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 JQR Sections Covered

- 3.1.3 (all)
- 3.1.4 (a)
- 3.1.5 (all)
- 3.1.6 (a, b)
- 3.1.7 (all)

- 3.1.8 (a, b, c, d, e, f)
- 3.1.9
- 3.1.10 (b, d, f)
- 3.1.11 (all)
- 3.1.18 (a, b)
- 3.1.19 (all)
- 3.1.20 (all)
- 3.1.21 (only if debug macros are used)
- 3.1.22 (all)