ITEC 371: Operating Systems

Project 1: RUFS – The RU FileSystem

In doing this project, abide by the RU Honor Code or you will fail the assignment and the course. This is not a group assignment

---

Problem 1
100 points

Goal
Create an application using C++ called RUFS that accepts one command line argument which is the name of the binary file used in this project. The binary file will store a basic file system that can contain directories, text files, and programs.

Specifications for the binary file
A file has the following rules concerning how it is represented in binary:
- A 11 character filename described below.
- The contents of the file (described below)

Filenames: 11 total characters in the following format:
- 8 characters from [A-Z|a-z] or [0-9] or a null terminator to indicate empty space
- A period
- A one character extension consisting of one of the following:
  - t for text tiles
  - p for programs
- A null terminator

A text file contains the following information
- The size of the contents stored in an integer
- Ascii text that can be sent to standard out

A program contains the following information
- Four bytes (or the size of an integer) containing the CPU requirements for the program
- Four bytes (or the size of an integer) containing the Memory requirements for the Program

A directory has the following rules concerning how it is represented in binary:
- A filename is the first part of the directory (see above for the filename format) with an ending of d instead of t or p.
- Next, the number of files/directories in the directory stored as an integer.
- Next, the files stored in the directory and the sub-directories contained in the directory (stored in the order entered by the user).

Lastly, EndX where X is the file name listed earlier (and padded with null terminators)

A root directory beginning at location 0 with the name root\0\0\0\0.d\0 (note \0 is the null terminator) is added to every filesystem created with this program.

Run time arguments
When the application runs it must ask the user for one of the following four commands: CreateDir, CreateFile, EndDir, and quit. Once a command is entered, the application asks for another command until the quit command is entered. Once quit is entered the application stops.

Remember: by default the binary file contains the value root\0\0\0\0.d. For each of the following commands information is added to the binary file in the correct location. The information added by each command is described below.

CreateFile asks for the filename, verifies that it matches the requirements listed above, and then asks the user to enter the contents of the file based on extension. The contents are then stored in the binary file in the current working directory (remember, root is the initial current working directory).

CreateDir will create a sub-directory in the current working directory which will become the new current working directory. The directory filename and contents are stored at the end of the current working directory.

EndDir will set the current working directory to be the previous working directory and add 14 bytes to the binary filesystem (End followed by the 11 character filename).

quit will stop the application. When the application is stopped it prints out the location in the filesystem for what was added by the user.

An example of the program running is listed below(using a 32 bit system):

```
Welcome to RUFS.  Enter one of the following commands:
CreateDir or CreateFile or EndDir or quit
Command>CreateFile
Enter filename>prog1.p
Enter CPU requirements>2
Enter memory requirements>3
Command>CreateDir
Enter directory name>first
Command>CreateFile
Enter filename>test.t
Enter contents>A
Command>EndDir
Command>quit
Binary file structure is:
0:  Directory:  root.d
```

```
11:  Directory root contains 2 files/directories
15: Filename:prog1.p
Type:  Program
Contents: CPU Requirement:  2, Memory Requirement 3
34: Directory first.d
45:  Directory first contains 1 file/directory
49:  Filename:  test.t
     Type:  Text file
60:  Size of text file:  1 byte
64:  Contents of text file:   A
65: End of directory first.d
79: End of directory root.d
```

Implementation note: You can store the entire contents of the file system in memory in data structure and create the file system when the quit option is entered. Or you can write information to the binary file piece by piece as it is entered.

Note: your code must account for extraneous spaces or newlines. If it does not handle these input cases, a grade deduction will be made. Your program will be tested by inputting the commands listed earlier into a file, then redirected into your program via ./NAME < inputFile . If your program does not work in this manner, you will not receive a passing grade on the project.

Submission Requirements:
Your files must be stored in a directory with the format project1_RUEmailLoginName. Also in that directory, include a Makefile that will compile and create an executable called RUFS. Tar/gzip that directory with a filename of project1_RUEMAIL.tar.gz. Submit this file to D2L. If you do not follow these requirements, you will receive a 0 for the project. I have automated grading setup for this project and unless you follow these requirements, it will not work.

Grading Rubric
Submission requirements: If not met, you receive a 0 for your submission. Homework 1 should have provided feedback necessary to successfully submit this project.

Runtime:

To get a C on your submission it must take the input from the specification and produce the output from the specification (albeit it cannot be hardcoded).

To get a B on your submission it must produce correct output for at least ½ of the additional test cases I will use on your program.

To get an A on your submission it must produce correct output for all of the additional test cases I will use on your program.

Design: Can lower your grade by one letter. Deductions are made for choices such as all of your code in one function. Use what you have learned from previous courses and you should have no issues with this deduction.

Comments: Can lower your grade by one letter if your submission doesn't contain comments on:

Variables – purpose, type, scope

Function headers (purpose, parameters, return type, called by / calls)

Explanation for blocks of code (inline comments)

Readme file for program containing name, what it is, how to use it.

Header for every source file containing purpose and functions / classes contained / where to start looking.