Filesystem

In this assignment you will write a user space portable index-allocated file system. Your program will provide the user with 2^{26} bytes of drive space in a disk image. Users will have the ability to create the filesystem image, list the files currently in the file system, add files, remove files, and save the filesystem. Files will persist in the file system image when the program exits.

You may complete this assignment in groups of up to four people. If you wish to be in a group you must add your group to the spreadsheet linked in the Canvas assignment no later than 04/23/23 at at 11:59pm.

Requirements

1. Your program shall print out a prompt of mfs> when it is ready to accept input.
2. The following commands shall be implemented:

Command	Usage	Description
insert	`insert <filename>`	Copy the file into the filesystem image
retrieve	`retrieve <filename>`	Retrieve the file from the filesystem image and place it in the current working directory
retrieve	`retrieve <filename> <newfilename>`	Retrieve the file from the filesystem image and place it in the current working directory using the new filename
read	`read <filename> <starting byte>`	Print <number of bytes> bytes from the file, in hexadecimal, starting at <starting byte>

Command	Usage	Description
	`<number of bytes>`	
delete	`delete <filename>`	Delete the file from the filesystem image
undel	`undelete <filename>`	Undelete the file from the filesystem image
list	`list [-h] [-a]`	List the files in the filesystem image. If the `-h` parameter is given it will also list hidden files. If the `-a` parameter is provided the attributes will also be listed with the file and displayed as an 8-bit binary value.
df	`df`	Display the amount of disk space left in the filesystem image
open	`open <filename>`	Open a filesystem image
close	`close`	Close the opened filesystem image
createfs	`createfs <filename>`	Creates a new filesystem image
savefs	`savefs`	Write the currently opened filesystem to its file
attrib	`attrib [+attribute] [-attribute] <filename>`	Set or remove the attribute for the file
encrypt	`encrypt <filename> <cipher>`	XOR encrypt the file using the given cipher. The cipher is limited to a 1-byte value
decrypt	`encrypt <filename> <cipher>`	XOR decrypt the file using the given cipher. The cipher is limited to a 1-byte value
quit	`quit`	Quit the application

3. The filesystem shall use an index allocation scheme.
4. The filesystem block size shall be 1024 bytes.
5. The filesystem shall have 65536 blocks.
6. The filesystem shall support files up to 2^{20} bytes in size.
7. The filesystem shall support up to 256 files.
8. The filesystem shall support filenames of up to 64 characters.
9. Supported file names shall only be alphanumeric with ".". There shall be no restriction to how many characters appear before or after the ".". There shall be support for files without a "."
10. The directory structure shall be a single level hierarchy with no subdirectories
11. The filesystem shall store the directory in the blocks 0-18.
12. The filesystem shall allocate block 19 for the free inode map
13. The filesystem shall allocate blocks 20-276 for inodes
14. The filestem shall allocate block 277 for the free block map
15. Blocks 278-65535 shall be used for file data.
16. Files shall not be required to be contiguous. Blocks do not have to be sequential.

Command Details

`insert`

`insert` shall allow the user to put a new file into the file system.

The command shall take the form:

```
insert <filename>
```

If the filename is too long an error will be returned stating:

```
insert error: File name too long.
```

If there is not enough disk space for the file an error will be returned stating:

```
insert error: Not enough disk space.
```

`retrieve`

The `retrieve` command shall allow the user to retrieve a file from the file system and place it in the current working directory.

The command shall take the form:

```
retrieve <filename>
```

and

```
retrieve <filename> <newfilename>
```

If no new filename is specified the `retrieve` command shall copy the file to the current working directory using the old filename.

If the file does not exist in the file system an error will be printed that states:

```
Error: File not found.
```

`delete` command

The `delete` command shall allow the user to delete a file from the file system

If the file does exist in the file system it shall be deleted and all the space available for additional files.

`undelete` command

The `undelete` command shall allow the user to undelete a file that has been deleted from the file system

If the file does exist in the file system directory and marked deleted it shall be undeleted.

If the file is not found in the directory then the following shall be printed:

```
undelete: Can not find the file.
```

`list` command

The `list` command shall display all the files in the file system, their size in bytes and the time they were added to the file system

If no files are in the file system a message shall be printed: `list: No files found.`

Files that are marked as hidden shall not be listed

`df` command

The `df` command shall display the amount of free space in the file system in bytes.

`open` command

The `open` command shall open a file system image file with the name and path given by the user.

If the file is not found a message shall be printed:

```
open: File not found
```

`close` command

The `close` command shall close a file system image file with the name and path given by the user.

If the file is not open a message shall be printed:

```
close: File not open
```

`savefs` command

The `savefs` command shall write the file system to disk.

`attrib` command

The `attrib` command sets or removes an attribute from the file.

Valid attributes are:

Attribute	Description
h	Hidden. The file does not display in the directory listing
r	Read-Only. The file is marked read-only and can not be deleted.

To set the attribute on the file the attribute tag is given with a +, ex:

```
mfs> attrib +h foo.txt
```

To remove the attribute on the file the attribute tag is given with a -, ex:

```
mfs> attrib -h foo.txt
```

If the file is not found a message shall be printed:

```
attrib: File not found
```

```
createfs command
```

`createfs` command

`createfs` shall create a file system image file with the named provided by the user.

If the file name is not provided a message shall be printed:

```
createfs: Filename not provided
```

`encrypt` command

The `encrypt` command shall allow the user to encrypt a file in the file system using the provided cipher. This is a simple byte-by-byte XOR cipher. Cyber Chef can help verify your encryption.

The command shall take the form:

```
encrypt <filename> <cipher>
```

The cipher is required to be 256 bits.

`decrypt` command

The `decrypt` command shall allow the user to decrypt a file in the file system using the provided cipher. This is a simple byte-by-byte XOR cipher.

The command shall take the form:

212 lines (132 sloc) | 9.99 KB

The cipher is required to be 256 bits.

Nonfunctional Requirements

1. You may code your solution in C or C++.
2. C files shall end in .c . C++ files shall end in .cpp
3. Your source files must be ASCII text files. No binary submissions will be accepted.
4. Tabs or spaces shall be used to indent the code. Your code must use one or the other. All indentation must be consistent.
5. No line of code shall exceed 100 characters.
6. Each source code file shall have the following header filled out:
7. All code must be well commented. This means descriptive comments that tell the intent of the code, not just what the code is executing.

8. Keep your curly brace placement consistent. If you place curly braces on a new line , always place curly braces on a new end. Don't mix end line brace placement with new line brace placement.
9. Each function should have a header that describes its name, any parameters expected, any return values, as well as a description of what the function does.
10. If your solution uses multiple source files you must also submit a cmake file or a makefile. Submissions without a cmake file or makefile will have a 20 pt deduction.

Grading

This assignment will be graded on the GitHub codespace. The assignment will be graded out of 100 points.

There is no quiz.

Your C code will be compiled with:

```
gcc -Wall -Werror --std=c99 <filename>
```

Your C++ code will be compiled with:

```
g++ -Wall -Werror <filename>
```

Compiler warnings are there to tell you something is not correct. Pay attention to them and you will save yourself a lot of late nights debugging code. Code that does not compile will earn 0. This assignment must be 100% your group's own work. No code may be copied from friends, previous students, books, web pages, etc. All code submitted is checked against a database of previous semester's graded assignments, current student's code and common web sources. Code that is copied from an external source will result in a 0 for the assignment and referral to the Office of Student Conduct.

Hints and Suggestions

Reuse the code from the mav shell. Your parser and main loop logic are already done. It will allow you to concentrate on the new functionality you have to implement and not on reimplementing code you've already written.

Give feedback