MergeFS

this is?

https://github.com/SegaraRai/MergeFS

Software for Windows that mainly mounts multiple directories as a single directory.
The plugin also supports mounting archive files and CUE sheets.

For example, if you have directory A and directory B as follows:

- A
    - abc
    - def
        - ghi
- B
    - def
        - jkl
    - mno
        - pqr

It is software that mounts this in directory C as follows.

- C
    - abc
    - def
        - ghi
        - jkl
    - mno
        - pqr

Probably similar to UnionFS or OverlayFS on Linux (I don't know because I've never used it).

The mount itself uses the Dokany library.

Use

- I want to combine things that have been managed across multiple drives into one
- I want to handle the contents of the file without expanding the archive file
    - Furthermore, I want to work with file changes at the mount destination without making changes to the archive file.

- I don't want to use the disk space just to cut out music files managed by CUE + FLAC for each track.

It is supposed to be used for such purposes.

Detailed explanation

Mount point

The mount destination is called the **mount point .**
In the first example, it's directory C.

The mount point can be a drive or a directory on the NTFS file system. It seems that you need to specify an unused drive letter when specifying a drive, and a directory that already exists when specifying a directory on the NTFS file system. It should be possible to mount it as a network drive depending on the support, but since all the processing around here is delegated to Dokany , please see that.

Mount source

The mount source is called the **mount source** (or simply the source).
In the first example, it's directory A or directory B.

Archive files and CUE sheets can be used as mount sources in addition to directories on the file system. The plug-in handles these mount sources.

Some mount sources are writable (eg directories on the filesystem) and some are non-writable (eg archive files and CUE sheets), and if the first mount source is writable, the mount destination is also writable. Will be. At this time, the written changes are saved in the first mount source or metadata.

Metadata

Metadata is data used to represent information (changes) that cannot be represented by the mount source alone.
For example, in the example of directories A and B above, suppose you mount to C in the order of A and B, and consider deleting C / mno at the mount destination C. The C / mno entity is in B, but B is not the first mount source and cannot be modified. Now we need the metadata to note that the C / mno has been deleted. Other metadata is used to reduce IO, for example when changing file attributes or moving files.
Currently the metadata is stored in its own binary format, but we plan to move it to SQLite 3 in the future.

Mount source priority

Mount source priority is required when there are conflicting files and directories. In other words, if a file or directory with the same name exists in multiple mount sources, the mount source priority is used to determine which one to refer to at the mount destination.
The rules for mounting source precedence are simple, with **the one specified earlier taking precedence** .
For example, if a file with the same name exists in both mount source X and mount source Y, and they are mounted together in Z, Z / example will be X / example if they are mounted in the order of X and Y. Z / example points to Y / example if it is mounted in the order of Y and X.

Change at the mount destination

If the first mount source is writable, then the mount destination is also writable. The changes you make will be reflected in the first mount source.
For example, if you create a new C / stu assuming that you are mounting A and B in the first directory A, B, and C, a new file will be created in directory A as A / stu. Alternatively, if you create a C / mno / vwu, the A / mno directory is first created in A, followed by the A / mno / vwu.

Case sensitivity

Case sensitivity is the case sensitivity of the alphabet.
For example, on Windows, you can usually access a file called example.txt on an NTFS volume as Example.txt. Also, example.dat and Example.dat cannot exist in the same directory. On Linux, on the other hand, instead of normally not being able to access a file called example.txt as Example.txt, example.dat and Example.dat can exist in the same directory.

In MergeFS, whether to distinguish the case of the file name is specified by the option at the time of mounting.
What is specified here is applied in the management of metadata etc. under the jurisdiction of LibMergeFS, but it may not be applied to the source plug-in that is not under the jurisdiction of LibMergeFS depending on the implementation of the source plug-in.
For now, except for the MFPSFileSystem, the case-sensitive specification is effective. The MFPSFileSystem depends on the result of calling the Windows API CreateFile. Basically, it is not case sensitive regardless of the specification.

We are also considering adjusting the case of the file name passed to the source plug-in on the LibMergeFS side so that the source plug-in does not need to support it.

Case preservation

Case preservation is whether to preserve the case when saving the file.
For example, in NTFS, if you save it as Example.txt, it will become Example.txt as it is. On the other hand, in FAT16, if only 8.3 file names are supported, the file names will be converted to all uppercase and saved. In other words, if you try to save it as Example.txt, it will be converted to EXAMPLE.TXT and saved.

In MergeFS, whether case is preserved depends on each mount source.
The MFPSFileSystem, which currently has the only writable mount source, depends on the result of calling the Windows API CreateFile, as in the case of Case sensitivity. Basically, it depends on the file system of the volume where the directory from which the mount source resides resides. For NTFS, for example, case is preserved.

Project structure

This project consists of three types: **LibMergeFS** , which is the core, a client, and a plug-in.

core

It is in the form of a DLL and is responsible for the core functions.

- **LibMergeFS**
  It has the role of bundling each mount source and providing it to Dokany as one mount source.
  Written in C ++.

Client (front end)

It provides the user with the core LibMergeFS functionality in the form of an executable file .

- **It will have a MergeFS**
  GUI and will eventually be the main front end.
  I plan to write it in C #, but it's almost incomplete. I can't write in C #. somebody help.

- **The front end of the MergeFSCC**
  CLI. Originally created for testing core functionality, this will be completed as another front end.
  Written in C ++.

- **MergeFSMC**
  task tray resident front end.
  You can read and mount the configuration file written in YAML.
  Probably the most practical client.
  Written in C ++.

Plugin

It adds various functions to the core functions in the form of DLL.

Source plugin

This is the only plugin format currently available. Allows you to mount various things (archive files and directories on the actual file system).

- **MFPSFileSystem**
  Mounts a directory on the actual file system.
  Written in C ++.
- **Mount the MFPS**
  Archive archive file. The archive file existing in the archive file is also recursively read as a directory. If possible, uncompressed files (tarball, etc.) are read directly without decompressing them in memory, but in the case of normal compressed files, they are first decompressed in memory. (It can be changed with compile-time options.) I am using 7-Zip
  to read the archive file . Written in C ++.
- Mounts music files saved in **MFPSCue CUE + BIN, CUE + FLAC, CUE + WAV.** Mount it as a group of WAV files divided for each track, not as a CD-DA. As with MFP Archive, everything is not expanded in memory and decoded as needed. (It can be changed by compile time option.) I am using libFLAC and libFLAC ++
  for FLAC decoding . Written in C ++.
- **MFPSNull A**
  read-only source with no content.
  You can make it a read-only mount by placing it at the beginning.
  Written in C ++.
- **MFPSMemory** (planned) Will be
  created to mount a writable source on memory.
  I plan to write it in C ++.

How to build

You need a compiler that supports the Windows environment and C ++ 17. I have confirmed compilation with Microsoft Visual Studio 2017 (15.9.7).

1. Clone this repository. Note that you need to clone recursively (including the submodules contained in this repository).
2. Open /MergeFS.sln and build.

Unsupported / unfinished

- Full-featured CLI front end

- GUI front end
- Security attributes
- Options to plugins