

Project 5: FAT12 File System

CS 411

For this project, you'll implement several file system operations in user space for the FAT12 file system. You're given a floppy diskette image file, driver-level source code, file system-level constant and type definitions, and a test suite application program.

Background

1. On the course web site on phoenix: *An Overview of FAT12*. This is your FAT12 file system specification document. Read it. Read it again. Read it a third time. Put it under your pillow at night when you're sleeping.
2. The starter repository on GitHub:
`https://github.com/GoucherCollegeCS411Spring2020/fat12filesystem.git`
3. You'll be using a new repository for this project. Go into Canvas and follow the link to create your Project 5 repo.
4. You can work on phoenix, or your own development system, for this project. Follow these steps to copy the starter repo to your repo:

```
git clone https://github.com/GoucherCollegeCS411Spring2020/fat12filesystem.git
```

```
cd fat12filesystem
```

```
git remote set-url origin <URL of your Project 5 repo>
```

```
git push
```

```
# use
```

```
git status
```

```
# to determine what files need to be added to the repo and explicitly add  
# them. For example:
```

```
git add fsops.c
```

```
# If you're working on a system (phoenix, say) where you haven't used  
# GitHub before, run these commands again:
```

```
# Let's start off by configuring your global git tooling. You'll only need
```

```

# to run the following commands once, unless your blow-up your VM and have
# to start over, or start working from another account. Use identity
# values that make sense for you _and_ that others will understand.

git config --global user.name "Your Name"
git config --global user.email Your.Name@mail.goucher.edu
git config --global push.default simple

# Rather than using geany, you could use emacs, gedit, or vi.

git config --global core.editor geany

# Okay, it can be pretty annoying to have to enter your GitHub credentials
# every single time you interact with your remote, so let's get some
# help. Run one of these two commands:

# 1) Cache credentials in memory for one hour (units are seconds). Adjust
# the timeout as you see fit.

git config --global credential.helper "cache --timeout=3600"

# 2) Store credentials unencrypted on disk permanently
# Default storage file is ~/.git-credentials

git config --global credential.helper store

# If you're paranoid, to delete the credential cache before the timeout
# expires run

git credential-cache exit

```

Tools

Read The Fine Manual for details about the following tools:

1. On phoenix, the Mtools suite may be used to examine your FAT12 image file:

```

mdir -/ -i floppyData.img

mdir -i floppyData.img ::/NEW/SUB

mtype -i floppyData.img ::/NEW/SUB/FILE1.TXT

```

2. You can run the following command to check the integrity of your file system following the use of any of the file system operations that modify the disk image:

```

/sbin/dosfsck -v floppyData.img

```

3. You can use this git command to restore the disk image to its original state:

```
git checkout -- floppyData.img
```

Description

`fd_mount()` and `fd_unmount()` are implemented for you. Refer to the comments in `fsops.c` for additional details about the following functions. Implement the following API-level file system functions in `fsops.c`:

1. `int fd_dir(int showAll)` — list the files in the current working directory. If `showAll` is 0, hidden files should not be listed. Otherwise, hidden files should be listed.

This function return the number of files (and directories) listed.

2. `int fd_cd(const char *dir)` — Change the current working directory. `dir` should name a sub-directory of the current working directory or be the string `".."`, indicating the parent directory. Assume that the parent of the root directory is the root directory.

Return 0 on success. Return -1 if `dir` is not a sub-directory of the current working directory or the string `".."`.

3. `int fd_type(const char *file)` — Display the contents of the file `file` in the current working directory. For the purposes of this function, a directory is not a file.

Returns the number of characters typed on success. Returns -1 if `file` is not a file in the current working directory.

4. `int fd_del(const char *file)` — Deletes the file `file` in the current working directory. For the purposes of this function, a directory is not a file. The blocks that had been in use by this file are marked free, as is the directory entry that had been in use by this file.

Returns the number of blocks freed on success. Returns -1 if `file` is not a file in the current working directory.

5. `int fd_creat(const char *file)` — Creates the file `file` in the current working directory. The file's attributes are all set to 0, the creation, write, and access times are set to the current time in the current time zone, the file's starting block is set to 0, and the file's size is set to 0.

On success, returns 0. Returns -1 otherwise.

6. `int fd_append(const char *file, const char *data, unsigned int len)` — Append `len` characters of data from `data` to the file `file`. For the purposes of this function, a directory is not a file.

On success, returns the number of characters appended. On failure, returns -1.

Although not required, you will find it helpful to also implement the following private helper functions in `fsops.c`:

```
static int fd_dir_root(int showAll)
```

```
static int fd_dir_subdir(int showAll)
```

```
static dirent_t *searchSubdir(const char *name, block_t block,
                             unsigned int *blkindex)

static dirent_t *getFreeRootEntry()
```

Fallback Position

If you're having trouble getting operations involving sub directories working, focus on getting operations working in the root directory.

Testing

The starter project contains a `Makefile`. You may compile the project by running the command `make`

Note that when I test your code, I will be using the original versions of `driver.h`, `driver.c`, `fsops.h`, and `exercise.c`.

Project 5 Turn-in

By the project deadline, email to me the GitHub https URL of your project repository. Remember to document assistance you received from others. This can be done in the email you send me your repo URL, in your `README.md` file, or in comments at the very top of your source code files. The `README.md` file can also be used to list any functionality missing from your project, as well as anything you think I should know.