Technical Report

User Manual

Compilation via make

Command

```
make
```

Basic client-server

Command

```
./basic_server
./basic_client
```

Summary: Shows a communication between a server and client

Directory listing server

Command

```
./dir_listing_server
./dir_listing_client
```

Summary: Client sends a command to server, server execute commands, and send back response

Basic disk-storage system

Command

```
./disk_server
./disk_client
```

Summary: Simulation of a disk server

Disk commands

```
Available commands are:
[C]reate - Create/initialize disk. 'C [CYL] [SEC]'
```

```
[D]elete - Delete current disk
[I]nfo - Get disk geometry information
[R]ead - Read from disk. 'R [CYL] [SEC]'
[W]rite - Write to disk. 'W [CYL] [SEC] [DATA]'

Please initialize disk with CREATE command: 'C [CYL] [SEC]'
```

File system server

Command

```
./fs_basic_server
./fs_basic_client
```

Summary: Simulation of a basic file system server

Basic fs commands:

```
[F]ormat filesystem of disk size [CYLINDER] [SECTOR]: 'F [C] [S]'
[C]reate file: 'C [NAME]'
[D]elete file: 'D [NAME]'
[L]ist current dirs/files, flag=0 minimal, flag=1 full: 'L [FLAG]'
[R]ead a file: 'R [NAME]'
[W]rite data to file: 'W [NAME] [DATA]'
[I]nformation of file system: name, valid, size (in bytes), etc
[U]nformat a filesystem and deletes disk

Please format filesystem with 'F' command
```

Directory structure

Commands

```
./fs_full_server
./fs_full_client
```

Summary: Simulation of a full file system with nested directory structure

Full filesystem commands

```
mkfs [CYLINDER] [SECTOR]        Create filesystem size of cylinder x sector
rmfs                            Remove filesystem
mkdir [NAME]                    Create a directory entry
rmdir [NAME]                    Remove a directory
mk [NAME]                       Create a file entry
```

```
rm [NAME]                       Remove a file
read [NAME]                     Read file data
write [NAME] [DATA]             Write data to file
append [NAME] [DATA]            Append data to file
cd [PATH]                       Change directory to PATH
ls                              List path contents
pwd                             List path contents
info                            Display current filesystem information

Please create and format filesystem with 'mkfs' command
```

Technical manual

Basic client-server implemenation

Unix socket is used to establish client/server communication

Directory listing server implementation

Fork is used to execute command and send back response

Basic disk-storage filesystem

mmap() is used to map the physical disk file into memory for read/write operations

Filesystem server implementation

- FatCell
  - Each FatCell is a linked-list node that points to the next disk block
  - Each FatCell is a 1:1 representation to a disk block
  - Three states for each element: FREE (-2), END (-1), USED (>= 0)
    - FREE: indicates that the element, or disk block is free to use
    - END: indicates the end of the linked-list node
    - USED: 0 or greater to the next disk block in the linked-list
- FAT data structure
  - An array of int elements, called FatCell
  - Each int element is a direct correspondence to a disk block plus starting offset
  - Maximum number of elements is the number of total disk blocks
  - Keeps a set of free FatCell
- Entry
  - A 128 bytes binary meta data representation in a disk block
  - Name of size MAX_NAME
  - Type of 0 (directory) or 1 (file)
  - dot: disk block number for current entry
  - dotdot: parent disk block number for current entry
  - size: size (in bytes) of entry (starts at 128 bytes) plus additional data
  - created: time of creation

    - last_accessed: time of last read operation
    - last_updated: time of last write operation
- DirEntry
    - Adds additional metadat to entry
    - dir_head: the head node for sub directory linked-list
    - file_head: the head node for file contents linked-list
- FileEntry
    - data_head: head node for file data blocks linked-list
    - data_size: total bytes of all data in data blocks linked list (stops at nul terminate)
- FatFS
    - Composition of Fat class and Disk class
    - Will query disk instance for total number of blocks
        - Format using Fat instance into disk
    - Creating directory:
        - Find target DirEntry in root entry via dir_head linked-list
        - Query Fat for free disk block
        - Write DirEntry into disk block
        - Add disk block to dir_head linked-list
    - Creating file:
        - Find target DirEntry in root entry via dir_head linked-list
        - Query Fat for disk block
        - Write FileEntry to disk block
        - Add disk block to file_head linked-list
    - Write data to file
        - Find FileEntry in target DirEntry's file_head linked-list
        - Query Fat for free disk block
        - Write a data to disk block
        - Add disk block to FileEntry's data_head linked-list
        - If there's still more data, repeat
    - Read data from file
        - Find FileEntry in target DirEntry's file_head linked-list
        - Iterate through FileEntry's data_head linked-list
        - For each block in data_head linked-list, read a disk block
        - Return entire data to user