CSE 312 Operating System Homework 2 Report

Şehmus ACAR
161044085

IMPORTANT !!

In my homework c does not allow '\' this char because this char it's special escape char so I decided to use '/' this char instead of '\' this char.

General Structure:

In this homework, I use four main structures for my file system: Superblock, Block, Directory Entry, and FAT table. The Superblock contains all critical information about the file system. Directory Entries are used to hold files and directories. Blocks store data in the files. The FAT table keeps track of which blocks are used by which files.

SUPERBLOCK

```
typedef struct {
    DirectoryEntry directory[MAX_FILES]; // Directory entries
    uint32_t directory_count; // Directory count
    uint16_t fat[FAT_ENTRIES]; // FAT table
    uint32_t block_size; // Block size
    uint32_t total_blocks; // Total number of blocks
    uint32_t free_blocks; // Number of free blocks
} FileSystem; // Super block
```

directory	Indicates the number of directory entries stored in memory.
block_size	It actually means block size.It take the value from user on terminal.Like makeFileSystem 1 mySystem.data.(1 is size_of_blocks)
fat	Contains the FAT table.
directory_count	Indicates the number of directory entries.
total_blocks	Indicates the total number of blocks.
free_blocks	Indicates the number of free blocks.

- And also I write superblock to my filesystem file.
- I store the superblock in block 0

```
fwrite(&fs, sizeof(FileSystem), 1, fp);
```

Directory Table and Directory Entries

Directory entries hold information about files and directories. Each file or directory has a corresponding entry. Below is the structure for `DirectoryEntry`:

```
typedef struct {
    uint8_t name[255]; // File name (variable length)
    uint32_t size; // Size of the file
    uint8_t permissions; // Permissions (R and W)
    time_t creation_time; // Creation date and time
    time_t modification_time; // Last modification date and time
    char password[20]; // Password for protection (if any)
    uint16_t first_block; // Index of the first data block
} DirectoryEntry;
```

name	Name of the file.
size	Size of the file.
permissions	Permissions (R = Read, W = Write).
creation_time	Creation time of the file
modification_time	Last modification date and time of the file.
password	Password for file protection.
first_block	Index of the first data block.

Managing Free Blocks

I manage the free blocks in my file system using a File Allocation Table (FAT). The FAT table tracks the status of each block. If a block is free, it is marked as 0xFFFF in the FAT table. When I add a new file or use a block, the corresponding FAT entry is updated, and the block is allocated. With this method, I effectively track and manage the free blocks within the file system.

```
uint16_t fat[FAT_ENTRIES]; // FAT table
```

Handling Arbitrary Length of File Names

File names are stored in a variable-length array within the *DirectoryEntry* structure. The name field is defined to hold up to 255 characters, allowing for a flexible and reasonably long file name.

```
uint8_t name[255];
```

Handling Permissions

File permissions are handled using a single byte where each bit represents a specific permission. In this simplified file system, only read (R) and write (W) permissions are used.

```
uint8_t permissions;
```

- Read permission: `0x01`
- Write permission: `0x02`
- Both read and write permissions: `0x03`

Handling Password Protection

Password protection for files is managed using a character array within the *DirectoryEntry* structure. If the *password* field is not empty, the file is considered protected, and operations on the file require the password.

```
char password[20];
```

The password can be up to 20 characters long, providing a simple yet effective way to secure files.

File System Initialization Function

This function creates a new file system and writes the superblock information to the file.

```
void initialize_file_system(const char *filename, float block_size) {
```

- This function creates a new file system and writes the superblock information to the file system.
- It includes basic information such as file system size, block size, and number of free blocks.
- Additionally, it fills the rest of the file system with zeros.

Testing File System Initialization Function:

```
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./makeFileSystem 1 fileSystem.dat
File system created: fileSystem.dat, Block size: 1024.00 KB
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./makeFileSystem 0.5 fileSystem.dat
File system created: fileSystem.dat, Block size: 512.00 KB
```

Make Directory Function

```
void make_directory(const char *filesystem_name, const char *path) {
```

- This function creates a new directory at the specified path within the file system.
- It ensures the directory is added to the correct location and verifies the existence of necessary parent directories.

Testing Make Directory Function:

```
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./fileSystemOper fileSystem.dat mkdir /usr/ysa/new_directory
Directory created: /usr/ysa/new_directory
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./fileSystemOper fileSystem.dat mkdir /usr/ysa/new_directory
Directory already exists: /usr/ysa/new_directory
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./fileSystemOper fileSystem.dat mkdir /usr/nonexistent_directory/new_directory
Parent directory does not exist: /usr/nonexistent_directory
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./fileSystemOper fileSystem.dat mkdir /usr/ysa/new_directory2
Directory created: /usr/ysa/new_directory2
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./fileSystemOper fileSystem.dat mkdir /usr/ysa/new_directory3
Directory created: /usr/ysa/new_directory3
```

Remove Directory Function

```
// Function to remove a directory from the file system
void remove_directory(const char *filesystem_name, const char *path) {
```

- This function removes an existing directory at the specified path within the file system.
- It ensures the directory is empty before removing it and updates the directory table and FAT accordingly.

Testing Remove Directory Function :

```
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./fileSystemOper fileSystem.dat rmdir /usr/ysa/empty_directory
Directory not found: /usr/ysa/empty_directory
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./fileSystemOper fileSystem.dat rmdir /usr/ysa/new_directory3
Directory removed: /usr/ysa/new_directory3
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./fileSystemOper fileSystem.dat dir "/"
Directory content: /
Name: /usr, Size: 0, Permissions: rw, Password protected: no
Name: /usr/ysa, Size: 0, Permissions: rw, Password protected: no
Name: /usr/ysa/test4, Size: 102400, Permissions: --, Password protected: no
Name: /usr/ysa/test5, Size: 102400, Permissions: -w, Password protected: no
Name: /usr/test6, Size: 102400, Permissions: r-, Password protected: no
Name: /usr/ysa/new_directory, Size: 0, Permissions: rw, Password protected: no
Name: /usr/ysa/new_directory2, Size: 0, Permissions: rw, Password protected: no
```

File Write Function

```
// Function to write a file to the file system
void write_file(const char *filesystem_name, const char *path, const char *linux_file) {
```

- This function creates a new file at the specified path and writes data to it.
- It checks the existence of the specified directory in the file system.
- Prevents the addition of the same file again.
- Finds an empty block in the FAT table and places the data.

Testing File Write Function:

```
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./fileSystemOper fileSystem.data write "/usr/ysa/test1" test3.txt
Failed to open file system: No such file or directory
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./fileSystemOper fileSystem.dat write "/usr/ysa/test1" test3.txt
File successfully written: /usr/ysa/test1
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./fileSystemOper fileSystem.dat write "/usr/ysa/test2" test3.txt
File successfully written: /usr/ysa/test2
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./fileSystemOper fileSystem.dat write "/usr/ysa/test3" test3.txt
File successfully written: /usr/ysa/test3
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./fileSystemOper fileSystem.dat write "/usr/ysa/test4" test3.txt
File successfully written: /usr/ysa/test4
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./fileSystemOper fileSystem.dat write "/usr/ysa/test5" test3.txt
File successfully written: /usr/ysa/test5
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./fileSystemOper fileSystem.dat write "/usr/test6" test3.txt
File successfully written: /usr/test6
```

File Read Function

```
// Function to read a file from the file system
void read_file(const char *filesystem_name, const char *path, const char *linux_file, const char *password) {
```

- This function reads the file at the specified path and writes its content to a Linux file.
- Checks the existence and permissions of the file in the file system.
- Then reads the file content and writes it to the specified Linux file.

Testing File Read Function:

```
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./fileSystemOper fileSystem.dat read "/usr/ysa/test5" test3.txt
File read: /usr/ysa/test5
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./fileSystemOper fileSystem.dat read "/usr/ysa/test1" test1.txt
File read: /usr/ysa/test1
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./fileSystemOper fileSystem.dat read "/usr/ysa/test2" test2.txt
File read: /usr/ysa/test2
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./fileSystemOper fileSystem.dat read "/usr/ysa/test3" test3.txt
File read: /usr/ysa/test3
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./fileSystemOper fileSystem.dat read "/usr/ysa/test4" test4.txt
File read: /usr/ysa/test4
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./fileSystemOper fileSystem.dat read "/usr/ysa/test5" test5.txt
File read: /usr/ysa/test5
```

```
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ cmp test1.txt test2.txt
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ cmp test1.txt test3.txt
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ cmp test1.txt test4.txt
```

After read Listing Function

```
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./fileSystemOper fileSystem.dat dir "/"
Directory content: /
Name: /usr, Size: 0, Permissions: rw, Password protected: no
Name: /usr/ysa, Size: 0, Permissions: rw, Password protected: no
Name: /usr/ysa/test1, Size: 102400, Permissions: rw, Password protected: no
Name: /usr/ysa/test2, Size: 102400, Permissions: rw, Password protected: no
Name: /usr/ysa/test3, Size: 102400, Permissions: rw, Password protected: no
Name: /usr/ysa/test4, Size: 102400, Permissions: rw, Password protected: no
Name: /usr/ysa/test5, Size: 102400, Permissions: rw, Password protected: no
Name: /usr/test6, Size: 102400, Permissions: rw, Password protected: no
```

File Delete Function

```
// Function to delete a file from the file system
void delete_file(const char *filesystem_name, const char *path) {
```

- This function deletes the file at the specified path from the file system.
- Checks the existence and permissions of the file.
- Updates the FAT table to free the file blocks.

Testing File Delete Function:

```
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./fileSystemOper fileSystem.dat del "/usr/ysa/test1"
File deleted: /usr/ysa/test1
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./fileSystemOper fileSystem.dat del "/usr/ysa/test2"
File deleted: /usr/ysa/test2
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./fileSystemOper fileSystem.dat del "/usr/ysa/test3"
File deleted: /usr/ysa/test3
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./fileSystemOper fileSystem.dat dir "/"
Directory content: /
Name: /usr, Size: 0, Permissions: rw, Password protected: no
Name: /usr/ysa, Size: 0, Permissions: rw, Password protected: no
Name: /usr/ysa/test4, Size: 102400, Permissions: rw, Password protected: no
Name: /usr/ysa/test5, Size: 102400, Permissions: rw, Password protected: no
Name: /usr/test6, Size: 102400, Permissions: rw, Password protected: no
```

Directory Listing Function

```
// Function to list the directory content
void list_directory(const char *filesystem_name, const char *path) {
```

- This function lists the contents of the specified directory.
- Prints the information of files and subdirectories in the directory to the screen.

Testing Directory Listing Function :

```
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./fileSystemOper fileSystem.dat dir "/"
Directory content: /
Name: /usr, Size: 0, Permissions: rw, Password protected: no
Name: /usr/ysa, Size: 0, Permissions: rw, Password protected: no
Name: /usr/ysa/test1, Size: 102400, Permissions: rw, Password protected: no
Name: /usr/ysa/test2, Size: 102400, Permissions: rw, Password protected: no
Name: /usr/ysa/test3, Size: 102400, Permissions: rw, Password protected: no
Name: /usr/ysa/test4, Size: 102400, Permissions: rw, Password protected: no
Name: /usr/ysa/test5, Size: 102400, Permissions: rw, Password protected: no
Name: /usr/test6, Size: 102400, Permissions: rw, Password protected: no
```

Permission Change Function

```
// Function to change file permissions
void change_permissions(const char *filesystem_name, const char *path, const char *permissions) {
```

- This function changes the permissions of the specified file or directory.
- Checks the existence of the file and sets the new permissions.

Testing Permission Change Function:

```
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./fileSystemOper fileSystem.dat chmod "/usr/ysa/test4" -rw
Permissions changed: /usr/ysa/test4
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./fileSystemOper fileSystem.dat chmod "/usr/ysa/test5" -r
Permissions changed: /usr/ysa/test5
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./fileSystemOper fileSystem.dat chmod "/usr/test6" -w
Permissions changed: /usr/test6
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./fileSystemOper fileSystem.dat dir "/"
Directory content: /
Name: /usr, Size: 0, Permissions: rw, Password protected: no
Name: /usr/ysa, Size: 0, Permissions: rw, Password protected: no
Name: /usr/ysa/test4, Size: 102400, Permissions: --, Password protected: no
Name: /usr/ysa/test5, Size: 102400, Permissions: -w, Password protected: no
Name: /usr/test6, Size: 102400, Permissions: r-, Password protected: no
```

Test Read Write After Changing Permissions (*Test4 file -rw*)

```
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./fileSystemOper fileSystem.dat write "/usr/ysa/test4" test3.txt
No write permission for file: /usr/ysa/test4
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./fileSystemOper fileSystem.dat read "/usr/ysa/test4" test3.txt
No read permission for file: /usr/ysa/test4
```

Test Read Write After Changing Permissions (*Test5 file -r*)

```
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./fileSystemOper fileSystem.dat write "/usr/ysa/test5" test5.txt
File already exists: /usr/ysa/test5
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./fileSystemOper fileSystem.dat read "/usr/ysa/test5" test5.txt
No read permission for file: /usr/ysa/test5
```

Test Read Write After Changing Permissions (*Test6 file -w*)

```
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./fileSystemOper fileSystem.dat read "/usr/test6" test6.txt
File read: /usr/test6
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./fileSystemOper fileSystem.dat write "/usr/test6" test6.txt
No write permission for file: /usr/test6
```

Add Password Function

```
// Function to add a password to a file
void add_password(const char *filesystem_name, const char *path, const char *password) {
```

- This function adds a password to the specified file.
- Checks the existence of the file and adds the password to the file entry.

Testing Add Password Function :

```
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./fileSystemOper fileSystem.dat addpw "/usr/ysa/test1" 1234
Password added: /usr/ysa/test1
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./fileSystemOper fileSystem.dat read "/usr/ysa/test1" test4.txt
Incorrect password: /usr/ysa/test1
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./fileSystemOper fileSystem.dat read "/usr/ysa/test1" test4.txt 1234
File read: /usr/ysa/test1
```

File System Dump Function

```
// Function to dump the file system information
void dump_file_system(const char *filesystem_name) {
```

- This function dumps the content of the specified file system to the screen.
- Displays the superblock information, directory entries, and FAT table.
- Reports the current status of all structures in the file system in detail.

Testing File System Dump Function:

```
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./fileSystemOper fileSystem.dat dumpe2fs

File                    | Size (KB)  | Used Block Count     | Password       | Read Perm  | Write Perm | Creation Time
-------------------------------------------------------------------------------------------------------------------------------
/usr                    | 0.0        | 0                    | No             | Y          | Y          | Sat Jun  8 14:52:37 2024

/usr/ysa                | 0.0        | 0                    | No             | Y          | Y          | Sat Jun  8 14:52:37 2024

/usr/ysa/test1          | 100.0      | 100                  | Yes            | Y          | Y          | Sat Jun  8 14:52:37 2024

/usr/ysa/test2          | 75.0       | 75                   | No             | Y          | Y          | Sat Jun  8 14:57:32 2024

-------------------------------------------------------------------------------------------------------------------------------
Total Used Block Count: 175

File System Information:
Total block count: 4096
Free block count: 3921
Directory count: 4
Block size: 1.0 KB
```

1) You can also run it with this command.”./test.sh”

```
acar@DESKTOP-IEVDRBN:/mnt/c/Users/Acar/Desktop/OSHW2$ ./test.sh
File system created: fileSystem.dat, Block size: 1024.00 KB
Directory created: /usr
Directory created: /usr/ysa
File successfully written: /usr/ysa/test1
File successfully written: /usr/ysa/test2
Directory content: /usr/ysa
Name: /usr/ysa, Size: 0, Permissions: rw, Password protected: no
Name: /usr/ysa/test1, Size: 102400, Permissions: rw, Password protected: no
Name: /usr/ysa/test2, Size: 76800, Permissions: rw, Password protected: no

File                    | Size (KB)  | Used Block Count     | Password       | Read Perm  | Write Perm | Creation Time
-------------------------------------------------------------------------------------------------------------------------------
/usr                    | 0.0        | 0                    | No             | Y          | Y          | Sat Jun  8 15:38:16 2024

/usr/ysa                | 0.0        | 0                    | No             | Y          | Y          | Sat Jun  8 15:38:16 2024

/usr/ysa/test1          | 100.0      | 100                  | No             | Y          | Y          | Sat Jun  8 15:38:16 2024

/usr/ysa/test2          | 75.0       | 75                   | No             | Y          | Y          | Sat Jun  8 15:38:16 2024

-------------------------------------------------------------------------------------------------------------------------------
Total Used Block Count: 175

File System Information:
Total block count: 4096
Free block count: 3921
Directory count: 4
Block size: 1.0 KB
File deleted: /usr/ysa/test2
Error: Directory is not empty: /usr/ysa

File                    | Size (KB)  | Used Block Count     | Password       | Read Perm  | Write Perm | Creation Time
-------------------------------------------------------------------------------------------------------------------------------
/usr                    | 0.0        | 0                    | No             | Y          | Y          | Sat Jun  8 15:38:16 2024

/usr/ysa                | 0.0        | 0                    | No             | Y          | Y          | Sat Jun  8 15:38:16 2024

/usr/ysa/test1          | 100.0      | 100                  | No             | Y          | Y          | Sat Jun  8 15:38:16 2024

-------------------------------------------------------------------------------------------------------------------------------
Total Used Block Count: 100

File System Information:
Total block count: 4096
Free block count: 3996
Directory count: 3
Block size: 1.0 KB
File deleted: /usr/ysa/test1
Error: Directory is not empty: /usr

File                    | Size (KB)  | Used Block Count     | Password       | Read Perm  | Write Perm | Creation Time
-------------------------------------------------------------------------------------------------------------------------------
/usr                    | 0.0        | 0                    | No             | Y          | Y          | Sat Jun  8 15:38:16 2024

/usr/ysa                | 0.0        | 0                    | No             | Y          | Y          | Sat Jun  8 15:38:16 2024

-------------------------------------------------------------------------------------------------------------------------------
Total Used Block Count: 0

File System Information:
Total block count: 4096
Free block count: 4096
Directory count: 2
Block size: 1.0 KB
File successfully written: /usr/ysa/test2
Permissions changed: /usr/ysa/test2
File not found: /usr/ysa/test1

File                    | Size (KB)  | Used Block Count     | Password       | Read Perm  | Write Perm | Creation Time
-------------------------------------------------------------------------------------------------------------------------------
/usr                    | 0.0        | 0                    | No             | Y          | Y          | Sat Jun  8 15:38:16 2024

/usr/ysa                | 0.0        | 0                    | No             | Y          | Y          | Sat Jun  8 15:38:16 2024

/usr/ysa/test2          | 75.0       | 75                   | No             | N          | N          | Sat Jun  8 15:38:16 2024

-------------------------------------------------------------------------------------------------------------------------------
Total Used Block Count: 75

File System Information:
Total block count: 4096
Free block count: 4021
Directory count: 3
Block size: 1.0 KB
```