# GTU Department of Computer Engineering

# CSE 312 – Spring 2024

# Homework 2 Report

**Barış Batuhan Bolat**

**210104004029**

# 1. Directory Table and Directory Entries

The file system implementation includes a structure for managing directory entries and tables.

- **Directory Entries**: These are managed using the DirectoryEntry class, which holds information such as filename, parent directory, timestamps for creation and modification, file type, permissions, and password.

```
class DirectoryEntry {
public:
    char filename[MAX_FILENAME_LENGTH];
    char parent[MAX_FILENAME_LENGTH];
    time_t created;
    time_t modified;
    int firstBlock;
    int size;
    int type; // 0: file, 1: directory
    char ownerPermissions[4]; // e.g., "RW" for read and write
    char password[MAX_PASSWORD_LENGTH];
};
```

- **Directory Table**: This is an array of DirectoryEntry objects that holds all entries for files and directories. The table is stored in the file system's metadata.

# 2. Free Blocks Management

Free blocks are managed using a free table that keeps track of which blocks are available for use. The free table is an array of integers where each entry represents a block; 1 indicates the block is free, and 0 indicates it is occupied.

- **Initialization**:

```
vector<int> free_table(number_of_blocks, 1);
for (int i = 0; i < 1 + free_blocks + fat_blocks + directory_blocks; i++) {
    free_table[i] = 0;
}
```

# 3. Handling Arbitrary Length of File Names

File names are managed within the DirectoryEntry class, which has a fixed maximum length defined by MAX_FILENAME_LENGTH. While this approach sets a limit, it simplifies management and avoids the complexity of dynamic memory allocation.

```
#define MAX_FILENAME_LENGTH 255
class DirectoryEntry {
    char filename[MAX_FILENAME_LENGTH];
    // other attributes
};
```

# 4. Handling Permissions

Permissions are managed using a simple string representation within each DirectoryEntry. The ownerPermissions attribute contains permissions such as "R" for read and "W" for write.

- **Checking Permissions**

```
if (strchr(directoryEntries[index].getOwnerPermissions(), 'R') == NULL) {
    std::cout << "ERROR: Read permission is denied for file \"" << child <<
"\"!\n";
    return;
}
```

# 5. Handling Password Protection

Password protection is handled using the password attribute in the DirectoryEntry class. When accessing a file, the system checks if a password is set and verifies it.

- **Password Verification**:

```
if (strcmp(directoryEntries[index].getPassword(), "") != 0) {
   if (password == nullptr || strcmp(directoryEntries[index].getPassword(),
password) != 0) {
       std::cout << "ERROR: Incorrect password for file \"" << child <<
"\"!\n";
       return;
   }
}
```

# 6. Function Names for File System Operations

Here are some key functions from the source code that handle various file system operations:

1. **Creating the File System**:
    - int main(int argc, char *argv[]) in makeFileSystem.cpp: Initializes the file system, creates the superblock, and sets up the free table and FAT table.
2. **Creating a Directory**:
    - void FileSystem::mkdir(char *parent, char *child, SuperBlock superBlock, DirectoryEntry directoryEntries[], int fat_table[], int free_table[]) in fileSystemOper.cpp: Creates a new directory entry.
3. **Deleting a File/Directory**:
    - void FileSystem::del(char *parent, char *child, char *password, SuperBlock superBlock, DirectoryEntry directoryEntries[], int fat_table[], int free_table[]) in fileSystemOper.cpp: Deletes a file or directory after verifying permissions and password.
4. **Reading a File**:
    - void FileSystem::read(char *parent, char *child, char *password, SuperBlock superBlock, DirectoryEntry directoryEntries[], int fat_table[], const char *filename) in fileSystemOper.cpp: Reads a file's content after checking permissions and password.

5. **Writing to a File**:
    - void FileSystem::write(char *parent, char *child, const char *filename, SuperBlock superBlock, DirectoryEntry directoryEntries[], int fat_table[], int free_table[]) in fileSystemOper.cpp: Writes content to a file, updating the FAT table and free table as necessary.
6. **Changing Permissions**:
    - void FileSystem::chmod(char *parent, char *child, char *permissions, char *password, SuperBlock superBlock, DirectoryEntry directoryEntries[]) in fileSystemOper.cpp: Changes the permissions of a file or directory.