Progress Report

Number of Implementations	Implementations	Signature
1.	*User Class*	
2.	*File Class*	
3.	*Directory Class*	
4.	*Simple File Manager Class*	

Main Function:

```
int main() {
    SimpleFileManager fileManager;

    while (true) {
        cout << "Simple File Management System" << endl;
        cout << "1. Create User" << endl;
        cout << "2. Create Directory" << endl;
        cout << "3. Create File" << endl;
        cout << "Enter your choice: ";
```

```
Simple File Management System
1. Create User
2. Create Directory
3. Create File
Enter your choice: _
```

- Here is the output and the first impression of main function. We use various class and its functions. We will learn them gradually.

User Class:

It represents a user with a name. It provides functionality to create, copy, move, and assign users.

- Here's the "**User**" class:

```
#include <bits/stdc++.h>
using namespace std;

class User {
public:
    User(string name) : name(name) {}

    // Copy constructor
    User(const User &other) : name(other.name) {}

    // Copy assignment operator
    User &operator=(const User &other) {
        if (this != &other) {
            name = other.name;
        }
        return *this;
    }

    // Move constructor
    User(User &&other) noexcept : name(std::move(other.name)) {}

    // Move assignment operator
    User &operator=(User &&other) noexcept {
        if (this != &other) {
            name = std::move(other.name);
        }
        return *this;
    }

    string getName() const {
        return name;
    }

private:
    string name;
};
```

- **Explanation:**

The "**User**" class manages user data in the file system. It stores user names privately and provides constructors and operators for efficient object handling. With methods like '***getname()***', it facilitates easy access to user information. This class serves as a crucial component for user management, ensuring data integrity and code clarity.

File Class:

It represents a file with a name, content, owner (user), and directory. It supports copy and move operations.

- Segment 1

```
class File {
public:
    File(string name, string content, const User &owner, const string &directory)
        : name(name), content(content), owner(owner), directory(directory) {}

    // Copy constructor
    File(const File &other)
        : name(other.name), content(other.content), owner(other.owner), directory(other.directory) {}

    // Move constructor
    File(File &&other) noexcept
        : name(std::move(other.name)), content(std::move(other.content)), owner(other.owner), directory(std::move(other.directory)) {}

    // Move assignment operator
    File &operator=(File &&other) noexcept {
        if (this != &other) {
            name = std::move(other.name);
            content = std::move(other.content);
            directory = std::move(other.directory);
        }
        return *this;
    }
```

In this segment of code, the following functionalities are implemented:

- **Constructor:** Initializes a **"File"** object with provided attributes. Here the constructor takes four parameters, ***'name'***, ***'content'***, ***'owner'*** & ***'directory'***.
- **Copy Constructor:** A copy constructor is defined to create a copy of a **"File"** object and it initializes the new **"File"** object with the same attributes as the original object being copied.
- **Move Constructor:** Transfers resources efficiently from one **"File"** object to another and retains the ownership and content from the source object.
- **Move Assignment Constructor:** Enables assignment from a temporary **"File"** object, optimizing resource management.

- Segment 2

```
    string getName() const {
        return name;
    }

    string getContent() const {
        return content;
    }

    const User &getOwner() const {
        return owner;
    }

    string getDirectory() const {
        return directory;
    }

    void updateDirectory(const string &newDirectory) {
        directory = newDirectory;
    }

private:
    string name;
    string content;
    const User &owner;
    string directory;
};
```

In this segment, some member functions and member variables are defined.

Member Functions:

- **getName ( ):** Returns the name of the file.
- **getContent ( ):** Returns the content of the file.
- **getOwner ( ):** Returns a reference to the owner or *user* of the file.
- **getDirectory ( ):** Returns of the directory of the file.
- **updateDirectory (** const string & newDirectory**):** Updates the directory of the file with the provided new directory.

Member Variables:

- **name:** Stores the name of the file.
- **content:** Stores the content of the file.
- **owner:** Stores a reference to the owner or *user* of the file.
- **directory:** Stores the directory of the file.

Directory Class:

It represents a directory with a name and a list of files. It allows moving files to another directory and adding files to itself.

- Segment 1

```
class Directory {
public:
    Directory(string name) : name(name) {}

    void moveFile(const string &fileName, Directory &destination) {
        auto it = find_if(files.begin(), files.end(),
                          [fileName](const File &file)
                          { return file.getName() == fileName; });

        if (it != files.end()) {
            destination.addFile(*it);
            it->updateDirectory(destination.getName()); // Update the directory information in the moved file
            files.erase(it);
            cout << "File " << fileName << " moved from " << name << " directory to " << destination.getName() << " directory." << endl;
        } else {
            cout << "File " << fileName << " not found in " << name << " directory." << endl;
        }
    }
```

Here some following works are done:

- **Constructor:** Initializes a **"Directory"** object with a given name.
- **'moveFile ( )' Function:** Moves a file from the current directory to another directory.

❖ Segment 2

```
    void addFile(const File &file) {
        files.push_back(file);
    }

    string getName() const {
        return name;
    }

    const vector<File> &getFiles() const {
        return files;
    }

private:
    string name;
    vector<File> files;
};
```

- **'addFile ( )' Function:** Adds a file to the directory.
- **'getName ( )' Function:** Returns the name of the directory.
- **'getFiles ( )' Function:** Returns the list of files in the directory.