TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ
KHOA CÔNG NGHỆ THÔNG TIN


# BÁO CÁO ĐỒ ÁN GIỮA KỲ

**Bộ môn:** An toàn và phục hồi dữ liệu

**Giảng viên:** Thái Hùng Văn

Đặng Trần Minh Hậu

**Nhóm thực hiện:**

20127308: Phan Minh Sáng

20127318: Phan Trí Tài

20127569: Tô Đình Phương Nam

*Hồ Chí Minh, ngày 25 tháng 11 năm 2023*

# Bảng phân công công việc

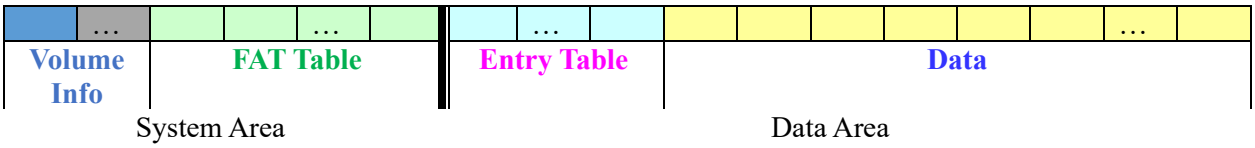
| STT | Công việc | Phụ trách |
|---|---|---|
| | PHẦN LÝ THUYẾT | |
| 1 | Tổ chức Volume | Cả nhóm |
| 2 | Tổng hợp các tiêu chí | Cả nhóm |
| 3 | Xây dựng mô hình hệ thống tập tin MyFS.DRS | |
| | - Bảo mật thông tin<br>- Bảo vệ dữ liệu<br>- Phục hồi dữ liệu | Phan Trí Tài |
| | - Mã hóa dữ liệu<br>- Tổ chức hệ thống thư mục | Phan Minh Sáng |
| | - Truy xuất dữ liệu theo từng cụm 512 byte<br>- Vùng dành riêng để backup | Tô Đình Phương Nam |
| 4 | Thiết kế kiến trúc tổ chức hệ thống tập tin MyFS.DRS | |
| | - Quản lý Bảo Mật và Phân Quyền<br>- Quản Lý Tính Toàn Vẹn Dữ Liệu<br>- Phục Hồi Dữ Liệu Đã Xóa<br>- Người dùng giao tiếp (Luồng Hoạt Động Chung) | Phan Trí Tài |
| | - Tổ Chức Hệ Thống Thư Mục<br>- Xử lý nghiệp vụ (Luồng Hoạt Động Chung) | Phan Minh Sáng |
| | - Truy Xuất Dữ Liệu Theo Đơn Vị 512 Byte<br>- Vùng Dành Riêng để Backup<br>- Ghi log và thống kê (Luồng Hoạt Động Chung) | Tô Đình Phương Nam |
| | PHẦN THỰC HÀNH | |
| 1 | Tạo / định dạng volume MyFS.DRS (với kích thước do người dùng tự nhập hoặc lựa chọn trong danh mục) | Phan Minh Sáng |
| 2 | Thiết lập /Đổi /Kiểm tra password truy xuất MyFS – nếu có thể thì hỗ trợ password động /passkey | Phan Minh Sáng |
| 3 | Liệt kê danh sách các file trong MyFS | Phan Minh Sáng |
| 4 | Đặt /đổi password truy xuất cho 1 file trong MyFS | Phan Trí Tài |
| 5 | Chép (Import) 1 file từ bên ngoài (từ hệ thống tập tin hiện hữu trên hệ điều hành đang sử dụng) vào MyFS | Phan Trí Tài |
| 6 | Chép (Outport) 1 file trong MyFS ra ngoài | Tô Đình Phương Nam |
| 7 | Xóa 1 file trong MyFS (cho phép người dùng chọn dạng không thể phục hồi hoặc có thể) | Tô Đình Phương Nam |

**Đánh giá mức độ:**

| Công việc | Hoàn thành |
|---|---|
| Phần Lý thuyết | 100% |
| Phần thực hành | 80% |
| - Câu 4 | 80% |
| - Câu 7 | 80% |

# Phần Lý thuyết

## Tổ chức Volume




- Tham khảo từ kiến trúc FAT32, Volume sẽ gồm 2 phần chính là System Area và Data Area.
- System Area bao gồm:
  - Volume Info: chiều dài và giá trị mật khẩu của Volume, số lượng entry, kích thước của bảng entry, …
  - FAT Table: trạng thái của các cluster (0 tương ứng với cluster trống, 1 tương ứng với cluster đang chứa dữ liệu, 2 tương ứng với cluster đã xóa nhưng có thể phục hồi được chỉ bị đè khi các cluster trống đã hết)
- Data Area bao gồm:
  - Entry Table: Entry chứa dữ liệu như vị trí của nó trong bảng FAT Table, chiều dài và giá trị mật khẩu tập tin, tập tin này có phải là thư mục hay không, kích thước của tập tin, chiều dài và giá trị tên của tập tin.
  - Data: chứa dữ liệu của tập tin.

## Tổng hợp các tiêu chí

1. **Bảo mật thông tin:**
   - Mã hóa dữ liệu quan trọng với một khóa duy nhất cho từng tập tin.
   - Xác thực và kiểm soát quyền truy cập của người dùng để ngăn chặn truy cập trái phép.
2. **Bảo vệ dữ liệu:**
   - Xây dựng cơ chế kiểm soát tính toàn vẹn của dữ liệu để phòng tránh hư hỏng hoặc mất mát thông tin.
   - Sử dụng checksum hoặc mã hash để kiểm tra tính toàn vẹn của các tập tin quan trọng.
3. **Phục hồi dữ liệu đã xóa:**
   - Thiết kế một hệ thống đánh dấu trạng thái xóa để theo dõi tập tin đã bị xóa.
   - Cung cấp khả năng khôi phục dữ liệu đã xóa bằng cách sử dụng bản sao lưu hoặc hệ thống log.
4. **Mã hóa dữ liệu:**
   - Sử dụng một hàm băm an toàn như SHA-256 để tạo ra khóa từ mật khẩu người dùng.

- Mỗi lần mã hóa sẽ tạo ra một bản mã khác nhau dựa trên khóa và nội dung, đảm bảo tính duy nhất và khó phá.

5. **Tổ chức đa dạng tập tin:**
    - Hỗ trợ tổ chức các tập tin và thư mục theo cấu trúc cây thư mục phân cấp để dễ quản lý.
    - Cung cấp các tính năng như tìm kiếm, sắp xếp, và lọc để dễ dàng định dạng tập tin quan trọng.
6. **Truy xuất dữ liệu theo từng cụm 512 byte:**
    - Xây dựng các hàm ReadBlock và WriteBlock để đọc và ghi dữ liệu theo từng đơn vị 512 byte.
    - Đảm bảo tính hiệu quả và linh hoạt trong việc quản lý và truy xuất dữ liệu.
7. **Vùng dành riêng để backup:**
    - Tạo một vùng lưu trữ trong file MyFS.DRS để chứa các bản sao lưu của các tập tin và dữ liệu quan trọng.
    - Áp dụng các chiến lược sao lưu định kỳ để đảm bảo tính an toàn và khả dụng của dữ liệu backup.

## Xây dựng mô hình hệ thống tập tin MyFS.DRS

1. **Bảo mật thông tin:**

- Kiểm soát quyền truy cập:
    - Xác thực và kiểm tra quyền truy cập của người dùng.
    - Áp dụng hệ thống quyền truy cập để ngăn chặn truy cập trái phép.

2. **Bảo vệ dữ liệu:**

- Kiểm tra tính toàn vẹn:
    - Sử dụng các giải thuật băm hoặc checksum để kiểm tra tính toàn vẹn của dữ liệu.
    - Thực hiện kiểm tra định kỳ để phòng tránh hư hỏng hoặc mất mát dữ liệu.

3. **Phục hồi dữ liệu đã xóa:**

- Quản lý trạng thái xóa:
    - Sử dụng một hệ thống đánh dấu trạng thái xóa để theo dõi tập tin đã bị xóa.
    - Dữ liệu được giữ lại trong một vùng dự trữ cho việc phục hồi.

4. **Mã hóa dữ liệu:**

- Quy trình mã hóa:
    - Mỗi lần mã hóa tạo ra bản mã khác nhau dựa trên khóa và nội dung.
    - Mã hóa theo khóa được tạo từ mật khẩu người dùng.
- Không lưu lại mật khẩu:
    - Không lưu trữ mật khẩu trực tiếp. Thay vào đó, sử dụng hàm băm an toàn để tạo khóa từ mật khẩu.

5. **Tổ chức đa dạng tập tin:**

- Cấu trúc thư mục phân cấp:
  - Hỗ trợ cấu trúc thư mục phân cấp để tổ chức tập tin.
  - Dữ liệu quan trọng có thể được đặt trong các thư mục an toàn.

6. **Truy xuất dữ liệu theo từng cụm 512 byte:**

- Hàm ReadBlock / WriteBlock:
  - Thiết kế hàm ReadBlock và WriteBlock để truy xuất dữ liệu theo từng đơn vị 512 byte.
  - Đảm bảo hiệu suất và linh hoạt trong việc quản lý và truy xuất dữ liệu.

7. **Vùng dành riêng để backup:**

- Vùng lưu trữ backup:
  - Tạo một vùng dành riêng trong MyFS.DRS để chứa các bản sao lưu của dữ liệu quan trọng.
  - Áp dụng chiến lược sao lưu định kỳ để đảm bảo tính an toàn và khả dụng của dữ liệu backup.

## Thiết kế kiến trúc tổ chức hệ thống tập tin MyFS.DRS

1. **Quản lý Bảo Mật và Phân Quyền:**

- Module mã hóa:
  - Chịu trách nhiệm mã hóa và giải mã dữ liệu.
  - Sử dụng một hàm băm an toàn để tạo khóa từ mật khẩu người dùng cho mỗi tập tin.
  - Đảm bảo rằng chỉ người dùng có quyền mới có thể giải mã tập tin.
- Module quản lý quyền truy cập:
  - Xác thực và kiểm soát quyền truy cập của người dùng.
  - Áp dụng cơ chế phân quyền để đảm bảo an toàn dữ liệu.

2. **Quản Lý Tính Toàn Vẹn Dữ Liệu:**

- Module kiểm tra tính toàn vẹn:
  - Tạo checksum hoặc mã hash cho dữ liệu để kiểm tra tính toàn vẹn.
  - Quản lý và kiểm tra tính toàn vẹn định kỳ.

3. **Phục Hồi Dữ Liệu Đã Xóa:**

- Module quản lý trạng thái xóa:
  - Theo dõi trạng thái xóa của tập tin.
  - Dữ liệu đã xóa được giữ lại trong vùng dự trữ để có thể phục hồi.

4. **Tổ Chức Hệ Thống Thư Mục:**

- Module quản lý thư mục:
  - Hỗ trợ cấu trúc thư mục phân cấp.
  - Cho phép người dùng tự do tổ chức và quản lý tập tin.

5. **Truy Xuất Dữ Liệu Theo Đơn Vị 512 Byte:**

- Module truy xuất dữ liệu:
  - Xây dựng hàm ReadBlock và WriteBlock để truy xuất dữ liệu theo từng đơn vị 512 byte.

- Đảm bảo hiệu suất và linh hoạt trong việc quản lý và truy xuất dữ liệu.

6. **Vùng Dành Riêng để Backup:**

- Module quản lý backup:
  - Xác định và quản lý vùng dành riêng để lưu trữ bản sao lưu.
  - Áp dụng chiến lược sao lưu định kỳ và kiểm tra tính toàn vẹn của bản sao lưu.

**Luồng Hoạt Động Chung:**

1. **Người dùng giao tiếp:**

- Tương tác với hệ thống thông qua giao diện người dùng.
- Thực hiện các thao tác như tạo, mở, xóa, di chuyển tập tin.

2. **Xử lý nghiệp vụ:**

- Chịu trách nhiệm thực hiện các thao tác của người dùng.
- Tương tác với các module bảo mật, quản lý quyền truy cập, kiểm tra tính toàn vẹn, và các module khác.

3. **Ghi log và thống kê:**

- Module log và thống kê:
  - Ghi lại các sự kiện quan trọng như truy cập, sửa đổi, backup.
  - Cung cấp thống kê về việc sử dụng hệ thống.

# Phần Thực hành

## Tạo / định dạng volume MyFS.DRS

```
void Volume::initVolume(string name, int volumeSize) {
    FILE *fp = nullptr;
    errno_t err = fopen_s(&fp, name.c_str(), "wb+");
    if (fp) {
        fseek(fp, volumeSize - 1, SEEK_SET);
        fwrite("", 1, 1, fp);
        fclose(fp);
    }
    err = fopen_s(&fp, name.c_str(), "rb+");
    if (fp) {
        string pw1, pw2;
```

- Bước 1: Tạo volume trống fp
- Bước 2: Sử dụng biến err để mở volume kiểm tra việc mở file có lỗi hay không
- Bước 3: Tạo một vùng dữ liệu rỗng có kích thước bằng với biến volumeSize
- Bước 4: Yêu cầu người dùng nhập password cho volume 2 lần
- Bước 5: Ghi Volume info và FAT table vào volume với các giá trị mặc định

## Thiết lập /Đổi /Kiểm tra password truy xuất MyFS

```
        case 2:
            cout << "Please enter the old password: ";
            getline(cin, pw);
            if (v->checkPassword(pw)) {
                cout << "Please enter the new password: ";
                getline(cin, pw);
                v->SetVolPass(pw);
                fseek(fp, 0, SEEK_SET);
                v->writeVolInfo(fp);
                cout << "SUCCESS!\n";
            }
            else
                cout << "WRONG PASSWORD!\n";
            system("pause");
            break;
```

```
bool VolumeInfo::checkPassword(string pw) {
    SHA256 sha256;
    string pwToCheck = sha256(pw);
    return (this->volPass.compare(pwToCheck) == 0);
}
```

Thiết lập password đã được thực hiện ở bước tạo volume

- Bước 1: Yêu cầu người dùng nhập password hiện tại của volume
- Bước 2: Hash password người dùng vừa nhập và so sánh với giá trị hash của password của volume. Nếu 2 giá trị giống nhau thì tiếp tục bước 3, không thì kết thúc.
- Bước 3: Yêu cầu người dùng nhập mật khẩu mới.
- Bước 4: Đổi password trong volume info, sau đó khi ghi volume info mới vào volume.

## Liệt kê danh sách các file trong MyFS

```
vector<Entry*> Volume::listEntry(FILE *f, VolumeInfo *v, vector<char> fatTable) {
    vector<Entry*> res;
    int size = v->volumeInfoSize() + fatTable.size();
    for (int i = 0; i < v->NumberOfEntry(); i++) {
```

```
void showEntries(const vector<Entry*>& entries) {
    int x = 0;
    int y = 0;
    system("cls");
    for (int i = 0; i < entries.size(); ++i) {
```

```
int RecursivePrint(Entry* e, int x, int y) {
    vector<Entry*> subEntryList = e->SubEntryList();
    if (subEntryList.size() == 0)
        return y - 1;
    for (int i = 0; i < subEntryList.size(); ++i) {
```

- Bước 1: Đọc các entry từ volume bằng hàm listEntry.
- Bước 2: Sử dụng hàm showEntries để đọc và in ra các entry từ listEntry. Nếu entry có thuộc tính _IsFolder là True (1) thì sử dụng hàm RecursivePrint để in ra các tập tin trong đó.

## Đặt /đổi password truy xuất cho 1 file trong MyFS

```
case 6:
    isExist = false;
    cout << "Nhap ten file can doi mat khau: ";
    getline(cin, changePWFile);
    if (entries.size() == 0)
        entries = Volume::listEntry(fp, v, fatTable);

    for (int i = 0; i < entries.size(); ++i) {
        if (entries[i]->Name() == changePWFile) {
            isExist = true;
            cout << "Nhap mat khau cua file: ";
            getline(cin, pw);
            //mat khau dung
            if (entries[i]->checkPassword(pw)) {
                cout << "Nhap mat khau moi: "; getline(cin, pw);
                entries[i]->changePassword(pw);
                cout << "Doi mat khau thanh cong!\n";
            }
            else
                cout << "Sai mat khau!\n";
        }
    }
```

- Bước 1: Yêu cầu người dùng nhập tên tập tin cần xóa.
- Bước 2: Đọc các entry từ volume và kiểm tra thử có tập tin hoặc thư mục nào trùng tên với tên người dùng nhập vào. Nếu không tìm ra thì thông báo lỗi. Còn tìm được thì đến bước 3.
- Bước 3: Yêu cầu người dùng nhập mật khẩu tập tin. Nếu trùng thì tiếp tục bước 4, không thì thông báo lỗi.
- Bước 4: Nhập mật khẩu mới cho tập tin và ghi đè vào entry tương ứng.
- Bước 5: Ghi đè entry vào volume. (Chưa hoàn thành)

## Chép (Import) 1 file từ bên ngoài vào MyFS

```
bool Volume::saveFileToData(FILE *&fp, VolumeInfo *&v, vector<char> &fatTable, string sourcePath, string targetName,
    string password) {
    //Entries cua cac file hoac folder doc duoc khi import
    vector<Entry*> entries;
    //Ten File/Folder
    string sourceName = targetName;
    //Thong tin day du cua File/Folder
    WIN32_FIND_DATAA fileData;
```

- Bước 1: Yêu cầu người dùng nhập địa chỉ tập tin muốn import, tên muốn lưu vào volume, mật khẩu của tập tin. Sau đó sử dụng hàm saveFileToData để lưu dữ liệu vào volume.
- Bước 2: Mở thử tập tin người dùng muốn import xem thử có lỗi hay không. Nếu lỗi thì xuất ra thông báo lỗi. Ngược lại thì tiếp tục bước 3.
- Bước 3: Tạo entry từ các thông tin mà người dùng nhập vào.
- Bước 4: Tìm các cluster trống trong FAT table.
- Bước 5: Ghi dữ liệu vào các cluster còn trống.

## Chép (Outport) 1 file trong MyFS ra ngoài

```
bool Volume::exportFile(FILE *fp, VolumeInfo *v, const vector<char> &fatTable, const vector<Entry*> &listEntry,
    string sourcePath, string targetPath) {
    bool isFound = false;
    Entry* exportingEntry = nullptr;
    queue<Entry*> s;
    //Path trong volume
    string path;
    //Path de ghi ben ngoai Windows
    string outputPath;
    for (int i = 0; i < listEntry.size(); i++) {
```

- Bước 1: Nhập địa chỉ của tập tin hoặc thư mục muốn outport và địa chỉ muốn lưu ở bên ngoài.
- Bước 2: Sử dụng hàm exportFile để xuất tập tin. Tìm kiếm địa chỉ tập tin hoặc thư mục người dùng muốn outport. Nếu không tìm được thì thông báo lỗi. Còn nếu tìm được thì tiếp tục bước 3.
- Bước 3: Yêu cầu người dùng nhập mật khẩu. Nếu không sai thì thông báo lỗi. Còn nếu đúng thì tiếp tục bước 4.
- Bước 4: Đọc thông tin của tập tin hoặc thư mục trong các cluster, sau đó xuất ra tập tin cho người dùng.

## Xóa 1 file trong MyFS

```
bool Volume::deleteFile(FILE* fp, VolumeInfo* v, vector<char>& fatTable, string name, vector<Entry*> listEntry) {
    bool isFound = false;
    Entry* e = nullptr;
    for (int i = 0; i < listEntry.size(); i++) {
        if (name == listEntry[i]->Name()) {
            isFound = true;
            e = listEntry[i];
            break;
```

- Bước 1: Nhập địa chỉ của tập tin hoặc thư mục muốn xóa. Đọc các entry trong volume.
- Bước 2: Sử dụng hàm deleteFile để xóa tập tin. Tìm kiếm địa chỉ tập tin hoặc thư mục người dùng muốn xóa. Nếu không tìm được thì thông báo lỗi. Còn nếu tìm được thì tiếp tục bước 3.
- Bước 3: Yêu cầu người dùng nhập mật khẩu. Nếu không sai thì thông báo lỗi. Còn nếu đúng thì tiếp tục bước 4.

- Bước 4: Giảm số lượng entry đi 1 trong Volume info và ghi vào volume. Đổi các giá trị bị xóa trong FAT table thành 2 tương ứng với cluster đã xóa nhưng có thể phục hồi được chỉ bị đè khi các cluster trống đã hết. (Chưa hoàn thành vì việc ghi đè volume info vào volume bị lỗi)
- Bước 5: Cập nhật lại cái cluster đã xóa.

# Tài liệu tham khảo


https://www.youtube.com/watch?v=owJ6wOZaLP0

Slide C1 và C3 trong Text: https://drive.google.com/drive/folders/1VLOniRuq99SmdtybVmLZgjJFykI_lGlc