ĐẠI HỌC QUỐC GIA
THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

CSC10007 - Hệ điều hành

BÁO CÁO ĐỒ ÁN

Hệ thống quản lý tập tin FAT32 - NTFS

Họ tên	MSSV
Nguyễn Lê Hồ Anh Khoa	23127211
Hình Diễm Xuân	23127524

Giảng viên hướng dẫn

Cao Xuân Nam

Ngày 13 tháng 4 năm 2025

Mục lục

1 Thông tin nhóm **3**

2 Thông tin đồ án **3**

3 Cài đặt chi tiết **4**
3.1 Lớp FileSystemReader 4
3.2 FAT32 5
3.3 Lớp NTFSReader 6
3.4 Lớp DiskExplorerApp 8

4 Đánh giá **10**
4.1 Bảng tự đánh giá các yêu cầu đã hoàn thành 10
4.2 Đánh giá tổng thể mức độ hoàn thành của bài nộp 11

5 Hướng dẫn sử dụng chương trình **11**

6 Tài liệu tham khảo **15**
6.1 Các tài liệu về các thư viện: 15
6.2 Các tài liệu khác: 16

Lời cảm ơn

Trước tiên, chúng em xin gửi lời cảm ơn chân thành đến các thầy cô trong Khoa Công nghệ Thông tin, Trường Đại học Khoa Học Tự Nhiên, ĐHQG-HCM đã tạo điều kiện cho chúng em thực hiện đồ án lập trình này. Sự hướng dẫn nhiệt tình và kiến thức quý báu từ các thầy cô đã giúp chúng em rất nhiều trong quá trình học tập và nghiên cứu.

Chúng em cũng xin chân thành cảm ơn thầy Cao Xuân Nam, người đã trực tiếp hướng dẫn và hỗ trợ chúng em trong suốt quá trình thực hiện đồ án. Thầy đã không quản ngại thời gian và công sức để hướng dẫn và giải đáp những thắc mắc của chúng em trong suốt quá trình thực hiện.

Ngoài ra, chúng em xin cảm ơn các anh chị và các bạn trong lớp, những người đã cùng nhau chia sẻ kinh nghiệm, ý tưởng và những lời động viên trong suốt thời gian thực hiện đồ án. Sự hỗ trợ và động viên của mọi người đã giúp chúng em vượt qua những khó khăn và thử thách trong quá trình nghiên cứu và phát triển.

Một lần nữa, xin chân thành cảm ơn tất cả mọi người!

Trân trọng,
Đại diện trưởng nhóm
Nguyễn Lê Hồ Anh Khoa

1 Thông tin nhóm

- **Môn học:** Hệ điều hành
- **Lớp:** 23CLC09
- **Giảng viên:** Cao Xuân Nam
- **Danh sách thành viên nhóm:**

STT	Họ tên	MSSV	Email
1	Nguyễn Lê Hồ Anh Khoa	23127211	nntkhoa23@clc.fitus.edu.vn
2	Hình Diễm Xuân	23127524	hdxuan23@clc.fitus.edu.vn

Bảng 1: Bảng phân công công việc

STT	Công việc	Người thực hiện
1	Cài đặt FAT32.	Diễm Xuân
2	Cài đặt NTFS.	Anh Khoa
3	Cài đặt giao diện người dùng.	Anh Khoa
4	Viết báo cáo	Anh Khoa, Diễm Xuân

2 Thông tin đồ án

- **Tên:** Hệ thống quản lý tập tin FAT32 - NTFS.
- **Môi trường lập trình:** Visual Studio Code
- **Ngôn ngữ lập trình:** Python.
- **Các thư viện và công cụ:**
  - **Thư viện:**
    * **datetime:** Thư viện chuẩn của Python để làm việc với thời gian và ngày tháng dùng để chuyển từ đơn vị nano-seconds về thời gian thực trong NTFS.
    * **threading** và **pythoncom**: Thư viện chuẩn của Python giúp chia luồng, tối ưu giao diện
    * **tkinter**,**customtkinter** và **pillow**: Thư viện cài đặt ngoài của Python để tạo giao diện đồ họa cho ứng dụng.
    * **wmi**: Thư viện cài đặt ngoài của Python để lấy tên những phân vùng có trên USB

- **Công cụ:**
  - **ChatGPT**, **DeepSeek**: Nghiên cứu và hỗ trợ tìm kiếm thông tin.
  - **Visual Studio Code**: Trình soạn thảo mã nguồn.
  - **Overleaf**: Trình soạn thảo LaTex để viết báo cáo.

3 Cài đặt chi tiết

3.1 Lớp FileSystemReader

Lớp `FileSystemReader` chịu trách nhiệm đọc và phân tích dữ liệu của Boot Sector từ một thiết bị lưu trữ (partition), từ đó xác định được hệ thống tập tin hiện hành. Các hàm trong lớp được phân tích như sau:

- **Hàm `__init__`**: Hàm khởi tạo mặc định của Python dùng để khởi tạo các biến thành viên của đối tượng. Cụ thể, hàm này nhận tham số `device` (đường dẫn đến thiết bị), sau đó gán giá trị này cho biến thành viên `self.device`. Tiếp theo, nó gọi hàm `read_boot_sector()` để đọc 512 byte đầu tiên của thiết bị, gán kết quả cho biến `self.boot_sector`.
- **Hàm read_boot_sector**: Chức năng chính của hàm này là đọc 512 byte đầu tiên của thiết bị lưu trữ, thường là Boot Sector. Trong hàm:
  - Sử dụng câu lệnh `with open(self.device, "rb") as disk` để mở thiết bị ở chế độ đọc nhị phân.
  - Đọc 512 byte đầu tiên bằng hàm `disk.read(512)`.
  - Bao gồm các khối ngoại lệ để xử lý các lỗi tiềm ẩn:
    - `PermissionError`: Nếu không đủ quyền truy cập, in ra thông báo lỗi và thoát chương trình.
    - `FileNotFoundError`: Nếu thiết bị không tồn tại hoặc không thể truy cập, in ra thông báo lỗi và thoát chương trình.
- **Hàm detect_filesystem**: Hàm này xác định loại hệ thống tập tin của thiết bị (FAT32 hoặc NTFS) bằng cách phân tích các byte ký hiệu nằm trong Boot Sector:
  - Đầu tiên, tạo chuỗi ký tự từ 8 byte bắt đầu tại offset `0x52` để kiểm tra ký hiệu FAT32.
  - Tiếp theo, tạo chuỗi ký tự từ 8 byte bắt đầu tại offset `0x03` để kiểm tra ký hiệu NTFS.
  - Dựa trên việc chuỗi ký hiệu chứa từ khóa "FAT32"hoặc "NTFS", hàm trả về kiểu hệ thống tương ứng. Nếu không thuộc 2 loại trên, trả về "UNKNOWN".
- **Hàm read_little_endian**: Hàm này chuyển đổi một dãy byte sang số nguyên theo định dạng Little Endian:

- Hàm nhận tham số gồm `data` (dữ liệu dạng byte array), `offset` (vị trí bắt đầu đọc), và `length` (số byte cần đọc).
- Sử dụng vòng lặp từ 0 đến `length - 1`, mỗi byte được dịch chuyển bit phù hợp (theo thứ tự little endian) và cộng dồn vào biến `value`.
- Sau khi xử lý xong, hàm trả về giá trị số nguyên kết quả.

3.2 FAT32

Lớp `fat32_reader` chịu trách nhiệm đọc và phân tích hệ thống tập tin FAT32 từ thiết bị hoặc file ổ đĩa. Các hàm của lớp được phân tích như sau:

- **Hàm __init__**: Gọi hàm khởi tạo của lớp cơ sở (FileSystemReader) để đọc Boot Sector từ thiết bị, khởi tạo biến để lưu toàn bộ bảng FAT.
- **Hàm read_fat32_info**: Đọc thông tin từ Boot Sector nếu là FAT32.
- **Hàm parse_short_name**: giải mã tên tập tin ngắn với 8 byte đầu (offset 0-7) là tên file, 3 byte sau (offset8-10) là phần mở rộng. Trả về dạng {tên}.{phần mở rộng}, nếu không có phần mở rộng thì chỉ trả về {tên}. Kiểm tra tên và phần mở rộng có được ghi là chữ thường hay không (tại offset 12 chứa byte đánh dấu chữ thường), rồi chuyển đổi lại cho đúng kiểu chữ khi hiển thị.
- **Hàm clean_filename**: Xóa các khoảng trống đầu cuối, bỏ cái ký tự không hợp lệ cũng như các bit trống
- **Hàm parse_lfn**: giải mã tên tập tin dài. Khởi tạo danh sách rỗng name_parts để lưu các phần nhỏ của tên tập tin dài từ mỗi entry phụ. Duyệt từng phần tử trong entry phụ theo thứ tự ngược vì các entry phụ LFN này được lưu theo thứ tự ngược trong bảng thư mục.Lưu tên dài bằng mã UTF-16LE, chia nhỏ ra 3 đoạn trong mỗi entry phụ:
  - Phần 1: từ offset 1 đến 10 (10 bytes) → 5 ký tự.
  - Phần 2: từ offset 14 đến 25 (12 bytes) → 6 ký tự.
  - Phần 3: từ offset 28 đến 31 (4 bytes) → 2 ký tự.

  Ghép 3 phần lại rồi lưu vào name_part, sau đó nối các chuỗi trong name_part lại rồi lưu vào full_name.
- **Hàm parse_date**: giải mã ngày, tháng, năm. Trả về chuỗi dưới dạng: YYYY-MM-DD. Raw_date có độ dài 16 bit (offset 16-17) trong đó (bit đọc từ phải sang trái): bit 0-4 biểu diễn ngày, bit 5-8 biểu diễn , bit 9-15 biểu diễn .
- **Hàm parse_time**: giải mã thời gian. Trả về chuỗi dưới dạng: HH-MM-SS. Raw_time có độ dài 16 bit (offset 14-15) trong đó (bit đọc từ phải sang trái): bit 0-4 biểu diễn giây, bit 5-10 biểu diễn phút, bit 11-15 biểu diễn giờ.
- **Hàm read_directory**: đọc toàn bộ nội dung của một thư mục, bắt đầu từ một cluster đầu. Nó phân tích các entry trong thư mục để lấy thông tin tên (bao gồm cả tên dài), loại (file/thư mục), cluster bắt đầu, kích thước, ngày và giờ tạo. Hàm duyệt

qua toàn bộ các cluster liên kết với thư mục đó bằng cách tra bảng FAT, và trả về danh sách các mục đã phân tích.

- **Hàm load_fat_table**: đọc toàn bộ bảng FAT từ ổ đĩa, bắt đầu từ vị trí fat_offset đọc đến hết kích thước của bảng.
- **Hàm get_file_clusters**: có nhiệm vụ dựa vào bảng FAT và cluster bắt đầu để lấy danh sách tất cả các cluster của file. Duyệt qua từng cluster cho đến khi gặp cluster kết thúc (0, 1, 0x0FFFFFF8) hoặc lỗi (x0FFFFFF7), nếu cluster hợp lệ thì ta thêm vào danh sách, sau đó tính vị trí của cluster xem có bị vượt quá kích thước của bảng FAT hay không, nếu hợp lệ thì lấy cluster tiếp theo để tính.
- **Hàm read_clusters_data**: Đọc dữ liệu từ nhiều cluster đã biết và trả về nội dung đầy đủ của một file trong hệ thống tập tin FAT32. Đầu tiên hàm sẽ tạo biến file_data để chứa dữ liệu của tập tin, mở ổ đĩa để đọc dưới dạng binary, duyệt qua từng cluster để tính toán offset để đọc đúng nội dung, kích thước, sau đó đọc rồi nối nội dung vào file_data. Sau khi đọc hết tất cả cluster, cắt đúng file_size và trả về.
- **Hàm read_file_content**: Có nhiệm vụ đọc nội dung tập tin. Nó phân tích Boot Sector để tính toán các thông số cần thiết, sau đó tải bảng FAT, truy vết các cluster chứa dữ liệu của tập tin và cuối cùng đọc dữ liệu từ các cluster đó rồi trả về nội dung tập tin.

3.3 Lớp NTFSReader

Lớp `NTFSReader` kế thừa từ lớp `FileSystemReader` và mở rộng khả năng xử lý hệ thống tập tin NTFS thông qua việc phân tích Boot Sector và MFT record. Các hàm của lớp được phân tích như sau:

- **Hàm __init__:** Gọi hàm khởi tạo của lớp cơ sở (`FileSystemReader`) để đọc Boot Sector từ thiết bị.
- **Hàm get_signed_byte:** Chuyển đổi giá trị của một byte không dấu sang dạng có dấu, hỗ trợ xử lý kích thước MFT record.
- **Hàm read_ntfs_info (static):** Phân tích Boot Sector của NTFS và trích xuất các thông tin quan trọng gồm:
  - Bytes per Sector (offset 0x0B, 2 bytes)
  - Sectors per Cluster (offset 0x0D, 1 byte)
  - Total Sectors (offset 0x28, 8 bytes)
  - MFT Cluster Number (offset 0x30, 8 bytes)
  - MFT Record Size (offset 0x40, 1 byte; xử lý giá trị có dấu)
  - Volume Serial Number (offset 0x50, 8 bytes)
- **Hàm parse_ntfs_timestamp:** Chuyển đổi NTFS timestamp, sử dụng thư viện datetime để chuyển đổi từ đơn vị 100 nanosecond(tính theo các đơn vị 100 nanosecond

kể từ thời điểm 1 tháng 1 năm 1601 theo giờ UTC) thành giờ UTC+7 để người dùng dễ dàng đọc.

- **Hàm parse_data_runs:** Phân tích chuỗi byte để trích xuất các "data run"của tập tin không resident. Hàm đọc từng header để xác định số byte biểu diễn độ dài và cluster offset, sau đó sử dụng thông tin này để xác định chuỗi cluster chứa dữ liệu của tập tin.
- **Hàm parse_ntfs_mft_record:** Phân tích MFT record để trích xuất các thông tin sau:
  - Các byte đầu tiên (offset 0 đến 3) của record chứa chuỗi "FILE"để xác nhận đó là một MFT record hợp lệ nếu không thì đó có thể là (record chưa sử dụng/đã xóa/lỗi...).
  - Flags để xác định record thuộc file hay thư mục (offset 22, 2 bytes).
  - **Thuộc tính FILE_NAME:**
    - **Header:** Mỗi thuộc tính bắt đầu với header chứa thông tin về kiểu và độ dài của thuộc tính.
    - **Nội dung:** Sau header, nội dung của FILE_NAME bao gồm:
      - **Parent Reference:** 8 byte đầu tiên thể hiện số record của thư mục chứa file.
      - **Thông tin tên tập tin:** Chứa độ dài (1 byte) và chuỗi tên file được mã hóa bằng UTF-16LE.
  - **Thuộc tính $DATA:** Xác định thông tin về dữ liệu tập tin:
    - **Resident:** Nếu dữ liệu đủ nhỏ, toàn bộ dữ liệu (với kích thước và vị trí được xác định trong header của thuộc tính) được lưu trực tiếp trong record.
    - **Non-resident:** Nếu dữ liệu lớn, không thể chứa trực tiếp, record chỉ chứa các tham chiếu dưới dạng data runs. Data runs liệt kê chuỗi các cluster trên đĩa, cùng với kích thước và offset tương đối, giúp xác định vị trí thật của dữ liệu.
- **Hàm `read_all_mft_records`:** Đọc liên tiếp các bản ghi MFT (Master File Table) từ thiết bị lưu trữ. Cụ thể:
  - Sử dụng kết quả từ hàm `read_ntfs_info` để lấy thông tin: Bytes per Sector, Sectors per Cluster, vị trí MFT (tính bằng cluster), và kích thước mỗi MFT record.
  - Tính toán offset vật lý bắt đầu của vùng MFT trên ổ đĩa.
  - Đọc tuần tự từng MFT record từ thiết bị.
  - Mỗi bản ghi đọc được sẽ được phân tích bằng hàm `parse_ntfs_mft_record` và lưu vào một dictionary với khóa là `record_number`.

- **Hàm `build_tree`:** Xây dựng cây thư mục từ danh sách các bản ghi MFT dựa trên quan hệ cha - con.
  - Mỗi record có trường `parent` chứa số hiệu record của thư mục cha.
  - Hàm khởi tạo trường `children` cho tất cả các record.
  - Duyệt qua từng record, nếu `parent` tồn tại và khác chính nó, sẽ thêm nó vào danh sách `children` của thư mục cha.
  - Trả về record gốc (thông thường có `record_number = 5`) đại diện cho thư mục gốc của hệ thống tập tin.
- **Hàm read_file_content:** Dọc nội dung của tập tin:
  - Nếu file là resident, trả về nội dung trực tiếp từ MFT record.
  - Nếu file là non-resident, sử dụng các data run để xác định và đọc dữ liệu từ các cluster được chỉ định.

3.4 Lớp DiskExplorerApp

Lớp `DiskExplorerApp` kế thừa từ `ctk.CTk` (một phần mở rộng của `tkinter`) và đại diện cho toàn bộ giao diện ứng dụng duyệt đĩa. Lớp này cung cấp các thành phần giao diện và chức năng để chọn đĩa, phân tích hệ thống tập tin FAT32 hoặc NTFS, và hiển thị nội dung cây thư mục.

Thành phần	**Thư viện sử dụng**	**Lý do sử dụng**
`DiskExplorerApp`, các khung chính như `disk_frame`, `main_frame`, `info_frame`, các nút và nhãn	`customtkinter`	Giao diện hiện đại, hỗ trợ dark mode, bo góc, dễ tùy biến hơn so với tkinter gốc
`Treeview`, `Scrollbar`	`tkinter.ttk`	Hiển thị cây thư mục phân cấp; chưa có widget tương đương trong customtkinter
`Toplevel`, `ScrolledText`	`tkinter`	Hiển thị nội dung văn bản dài với khả năng cuộn; phù hợp với file .txt

Bảng 2: So sánh thành phần sử dụng giữa `tkinter` và `customtkinter`

Thành phần / Thư viện	Thư viện sử dụng	Lý do sử dụng
Xử lý đa luồng	`threading`	Cho phép thực hiện các tác vụ như duyệt thư mục hoặc quét file mà không làm đứng giao diện người dùng (GUI)
Tích hợp COM trên Windows	`pythoncom`	COM yêu cầu mỗi luồng phải gọi CoInitialize() trước khi tương tác với WMI. Dùng pythoncom để khởi tạo/gỡ bỏ COM đúng cách trong mỗi luồng.
Xử lý ảnh	`PIL.Image`, `ImageTk`	Đọc, resize, và hiển thị ảnh trong giao diện; `ImageTk.PhotoImage` chuyển ảnh thành định dạng tkinter có thể hiển thị được
Lấy thông tin hệ thống	`wmi`	Giao tiếp với Windows Management Instrumentation (WMI) để lấy tên những partition từ USB (Removable)

Bảng 3: Các thư viện hỗ trợ ngoài `tkinter` trong ứng dụng

- **Thuộc tính** `current_reader`: Một đối tượng đọc hệ thống tập tin, có thể là `FAT32Reader` hoặc `NTFSReader`, được khởi tạo sau khi phát hiện hệ thống tập tin từ đĩa.
- **Thuộc tính** `fs_type`: Chuỗi đại diện cho loại hệ thống tập tin được phát hiện, ví dụ "FAT32" hoặc "NTFS".
- **Thuộc tính** `current_cluster_stack`: Stack (danh sách) dùng để lưu các cluster hiện tại trong quá trình điều hướng hệ thống tập tin FAT32.
- **Thuộc tính** `current_node_stack`: Stack dùng để lưu các node hiện tại trong cây thư mục NTFS.
- **Thuộc tính** `ntfs_records`: Danh sách chứa toàn bộ MFT records đã được phân tích từ hệ thống NTFS.
- **Thuộc tính** `ntfs_root`: Node gốc của cây thư mục NTFS, được xây dựng từ các MFT records.
- **Thuộc tính** `folder_icon, file_icon, txt_file_icon`: Các biểu tượng ảnh đại diện cho thư mục, tập tin chung và tập tin văn bản (.txt), được hiển thị trong cây thư mục.
- **Giao diện người dùng**: Được chia thành ba phần chính:
  - `disk_frame`: Gồm combobox chọn đĩa và nút xác nhận, dùng để chọn đĩa cần phân tích.
  - `main_frame`: Chứa `ttk.Treeview` dùng để hiển thị cây thư mục và các tập tin.

  - `info_frame`: Hiển thị thông tin chi tiết về hệ thống tập tin hiện tại.

- **Hàm** `refresh_disks`: Làm mới danh sách các ổ đĩa, kiểm tra loại hệ thống tập tin và cập nhật giao diện hiển thị nếu người dùng thay đổi ổ đĩa.
- **Hàm** `select_disk`: Được gọi khi người dùng chọn đĩa và nhấn nút xác nhận. Hàm sẽ tạo đối tượng đọc hệ thống tập tin tương ứng, khởi tạo cây thư mục và cập nhật thông tin.
- **Hàm** `populate_fat32_tree`: Hiển thị các thư mục và tập tin từ một cluster trong hệ thống FAT32 lên `treeview`.
- **Hàm** `populate_ntfs_tree`: Hiển thị các node con của một thư mục trong cây NTFS.
- **Hàm** `navigate_back`: Cho phép người dùng quay lại thư mục cha (dùng stack lưu lịch sử).
- **Hàm** `on_item_double_click`: Bắt sự kiện nhấn đúp vào một item trong cây, nếu là thư mục thì điều hướng vào trong, nếu là file `.txt` thì hiển thị nội dung.
- **Hàm** `display_file_content`: Hiển thị nội dung tập tin văn bản `.txt` bằng cửa sổ mới, sử dụng `ScrolledText`.

4 Đánh giá

4.1 Bảng tự đánh giá các yêu cầu đã hoàn thành

Bảng 4: Bảng tự đánh giá đồ án

STT	**Yêu cầu**	**Mức độ hoàn thành**
1	Tự động phát hiện định dạng của từng phân vùng (FAT32 / NTFS).	100%
2	Hiển thị cây thư mục của từng phân vùng được chọn, cho phép người dùng thu gọn hoặc mở rộng các thư mục thông qua thao tác tương tác.	100%
3	Tên thư mục hoặc tập tin hiển thị trong cây thư mục là tên đầy đủ.	100%
4	Khi chọn bất kỳ thư mục/tập tin nào, hệ thống sẽ đọc và hiển thị thông tin cho người dùng	100%
5	Hiển thị nội dung của tập tin được chọn (chỉ áp dụng với định dạng văn bản).	100%
	Tổng cộng	**100%**

4.2 Đánh giá tổng thể mức độ hoàn thành của bài nộp

Bài nộp đã hoàn thành đầy đủ các yêu cầu đề ra trong bài tập. Tất cả các yêu cầu đều đã được cài đặt và kiểm thử thành công. Tổng thể, bài nộp đã hoàn thành 100% các yêu cầu đề ra.

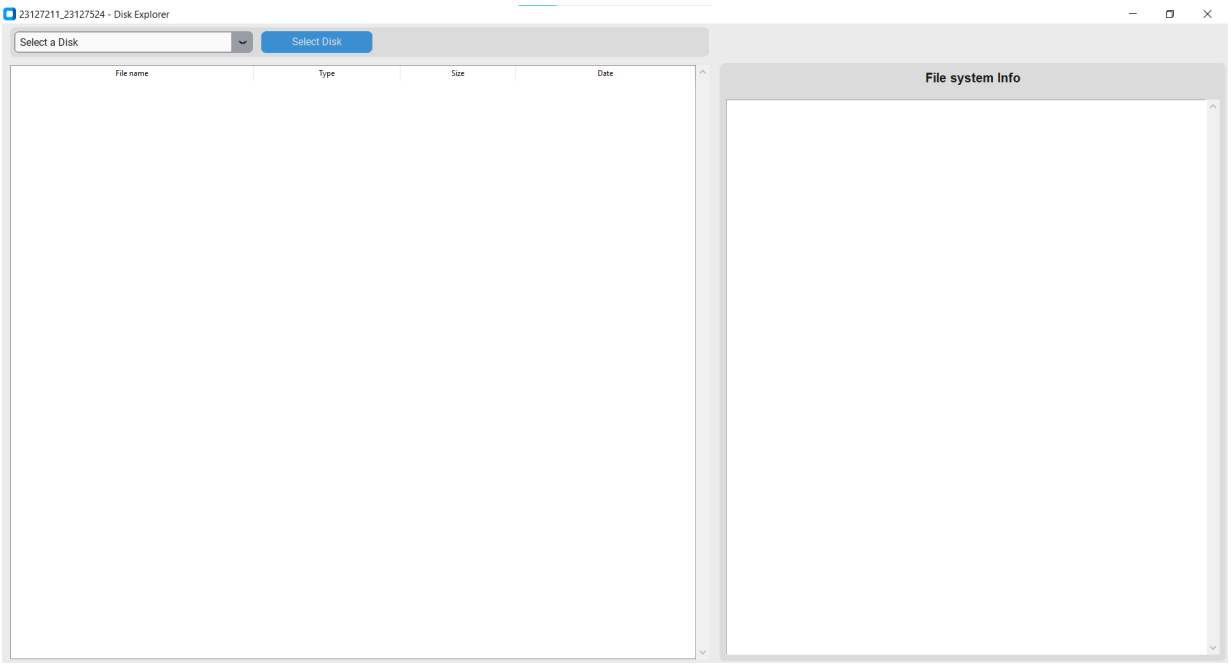
5 Hướng dẫn sử dụng chương trình

Lưu ý:

- Cài đặt ngôn ngữ lập trình Python
- Cài đặt các thư viện ngoài bằng cú pháp sau trên terminal: `pip install customtkinter pillow wmi`

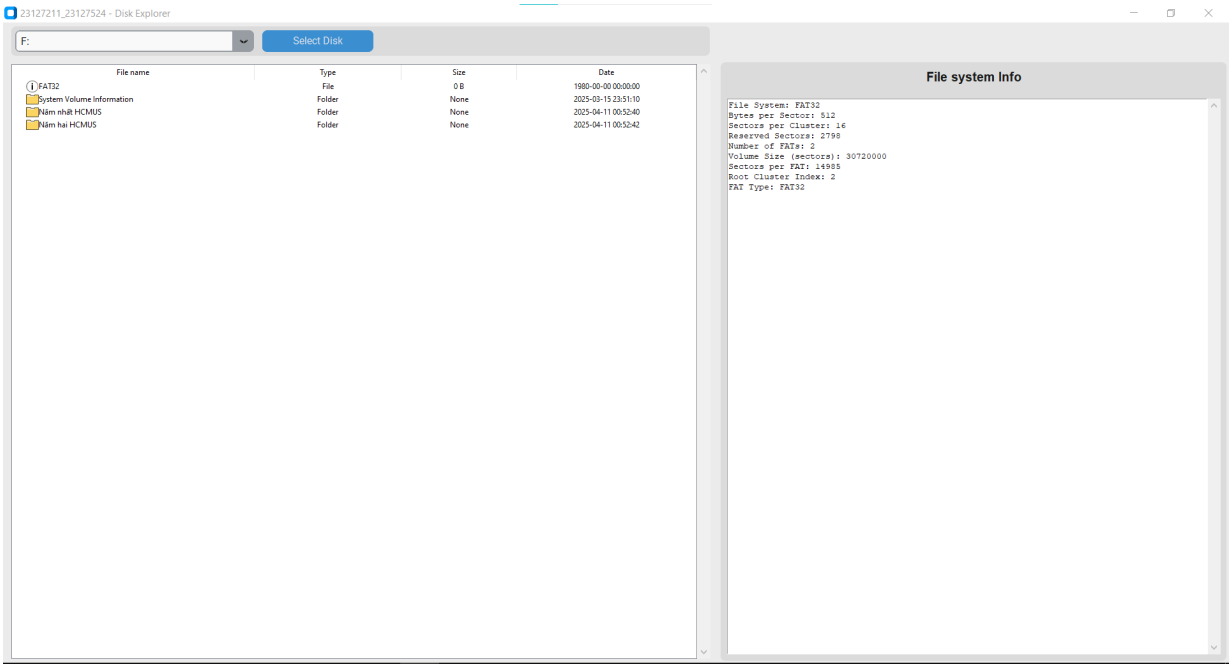
Hướng dẫn chi tiết

- **Bước 1:** Tải toàn bộ thư mục mã nguồn về máy tính
- **Bước 2:** Vào thư mục Source, chạy file main.py bằng câu lệnh py (hoặc python) main.py trên terminal.

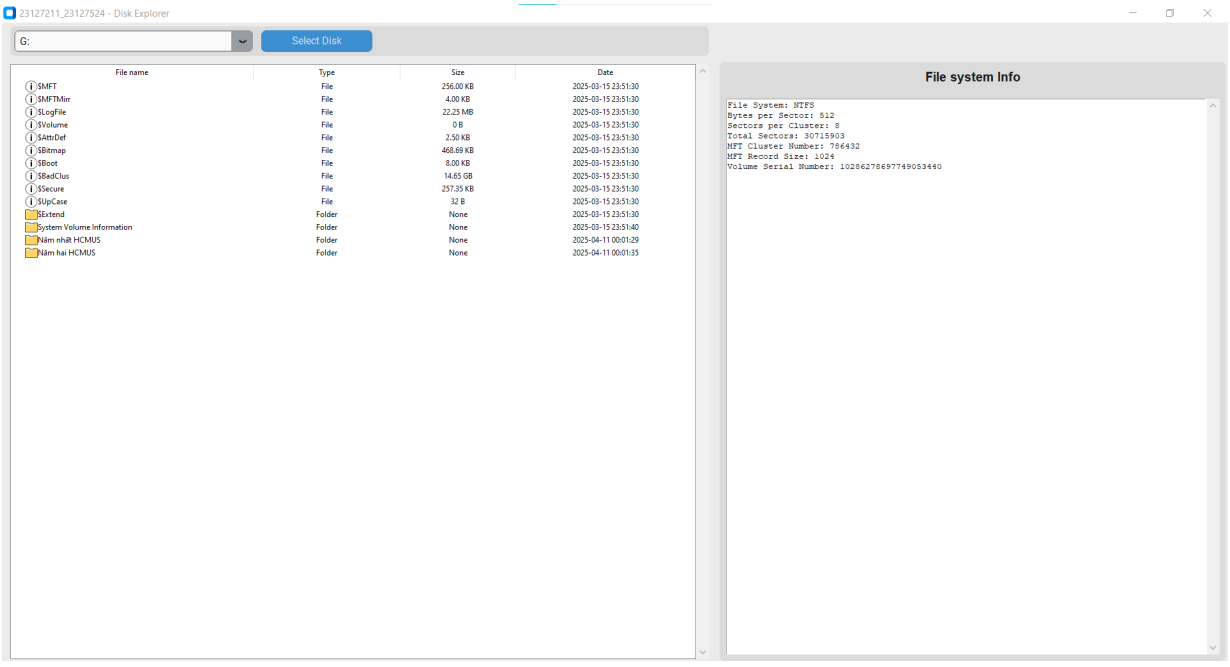


Hình 1: Giao diện khi vừa chạy chương trình.

- **Bước 3:** Giao diện hiện ra, chọn phân vùng từ USB muốn đọc và bấm "Select Disk". Phần bên phải sẽ hiển thị các thông tin quan trọng của partition, phần bên trái sẽ hiển thị cây thư mục.



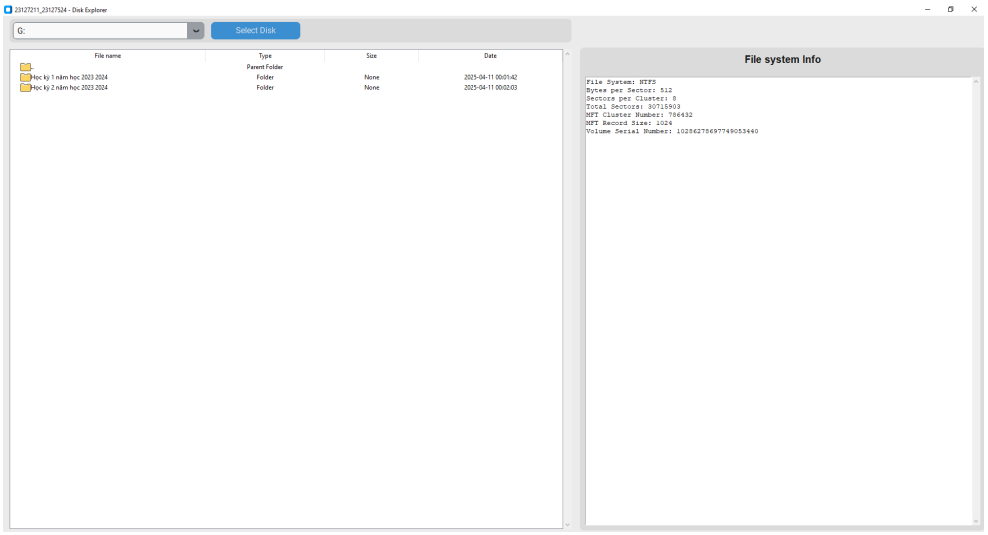
(a) FAT32



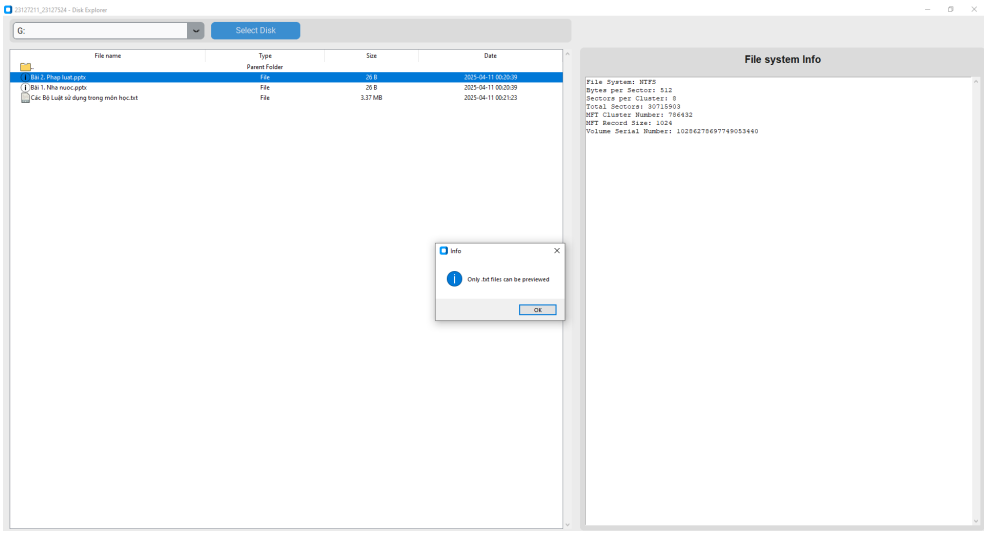
(b) NTFS

Hình 2: Giao diện chương trình sau khi chọn phân vùng

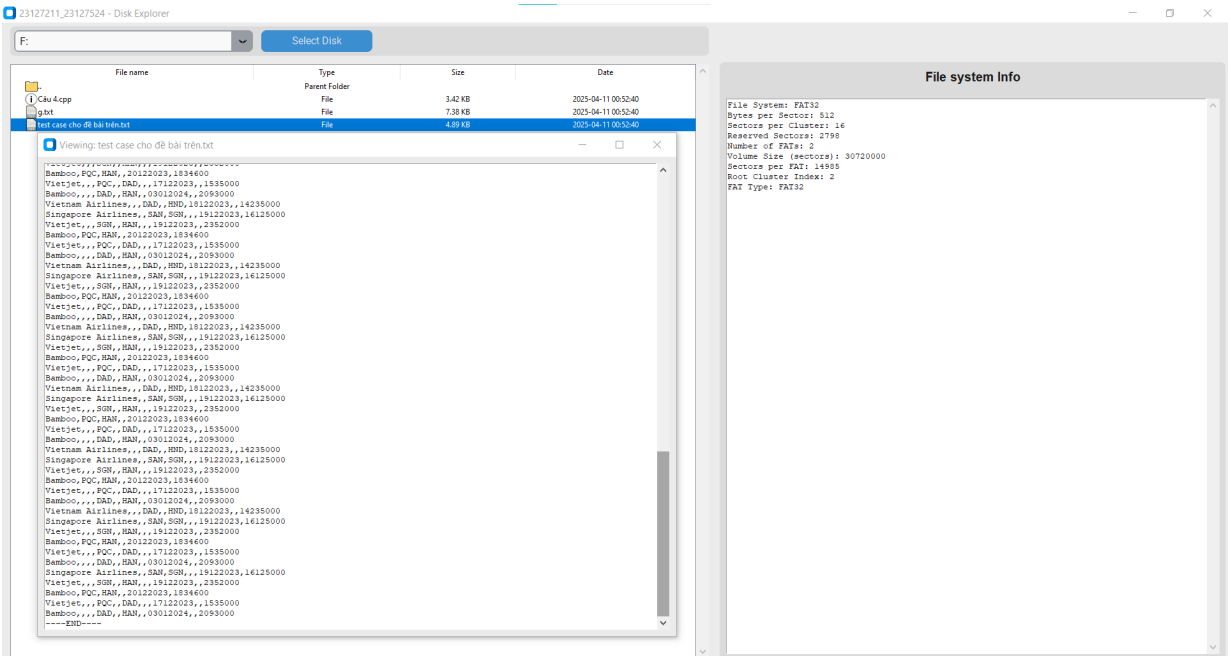
- **Bước 4:** Nhấp đúp chuột vào thư mục/file bạn muốn, nếu là thư mục thì điều hướng vào trong, nếu là file `.txt` thì hiển thị nội dung, nếu là định dạng file khác thì sẽ báo lỗi, nếu là item **..** thì sẽ điều hướng ra thư mục cha.



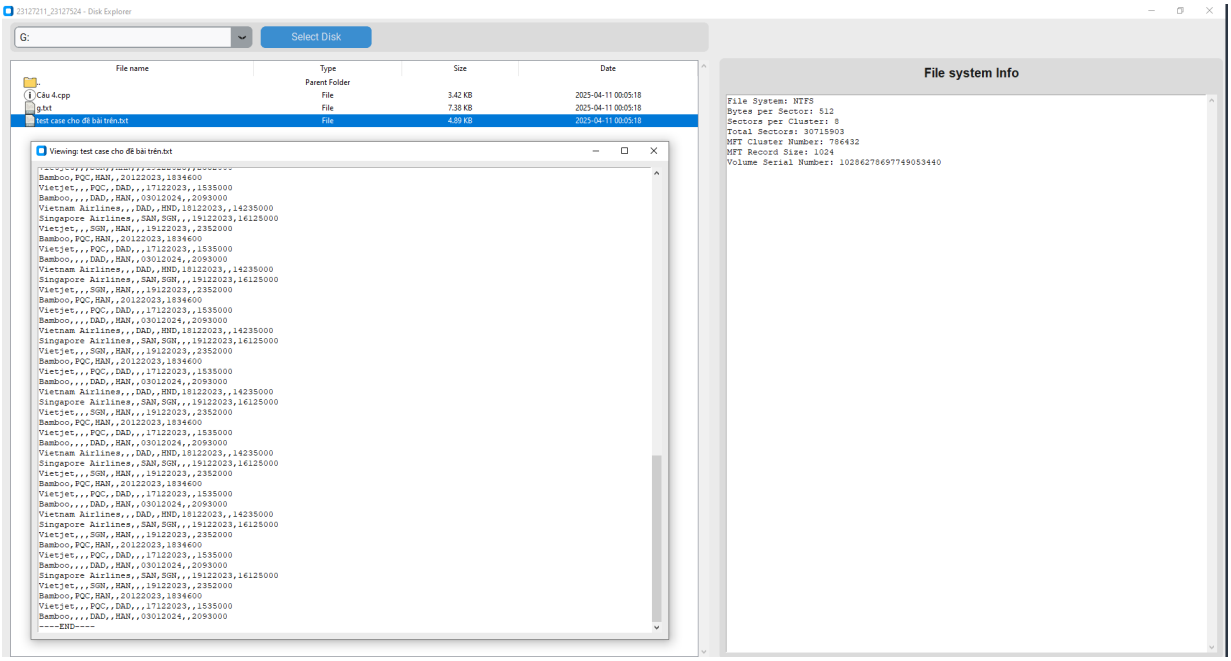
Hình 3: Giao diện sau khi người dùng chọn vào thư mục.



Hình 4: Chương trình sẽ thông báo nếu bạn chọn file khác đuôi .txt.

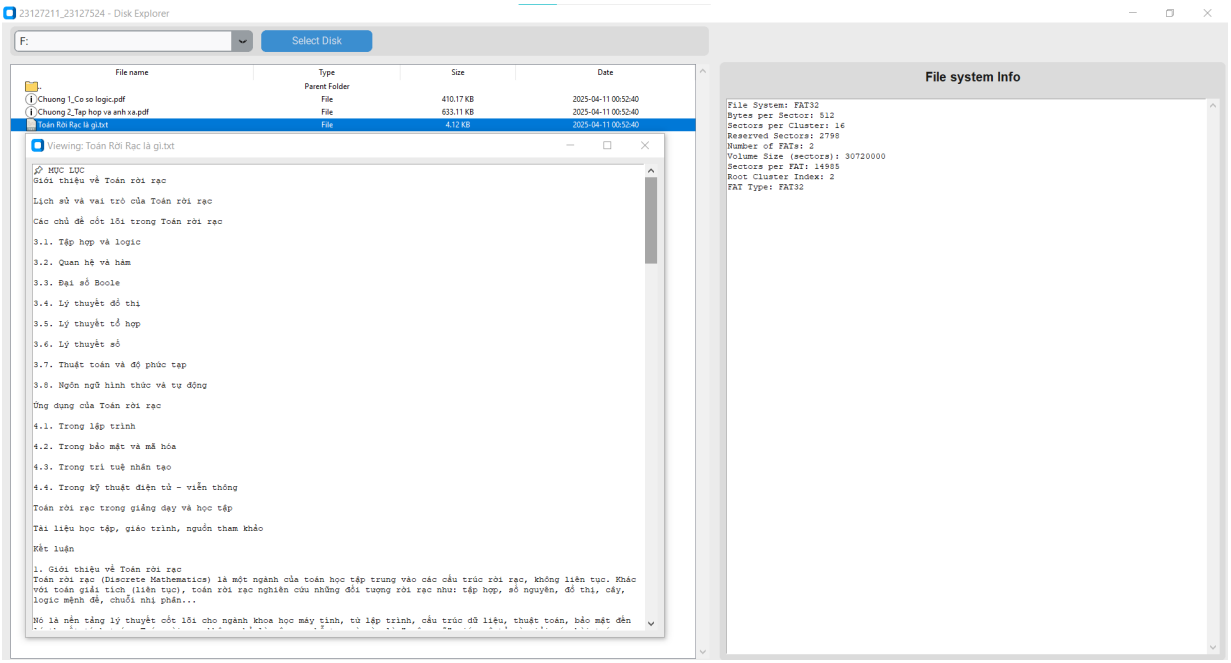


(a) FAT32

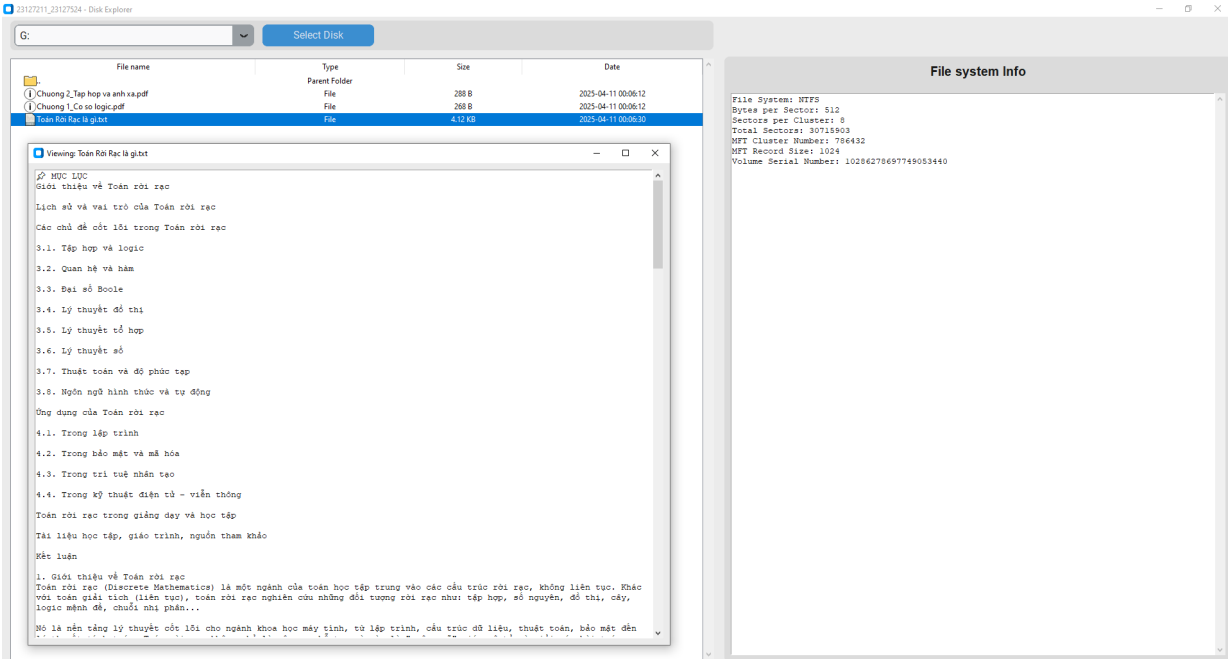


(b) NTFS

Hình 5: Giao diện đọc file đuôi .txt



(a) FAT32



(b) NTFS

Hình 6: Đọc tốt nội dụng file kể cả file tiếng việt

6 Tài liệu tham khảo

Ref_FAT, Ref_NTFS

6.1 Các tài liệu về các thư viện:

datetime, threading, pythoncom, tkinter, customtkinter, pillow, wmi,

6.2 Các tài liệu khác:

8.3 file name in FAT32

Quản lý hệ thống tập tin trên Windows