

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN, ĐHQG - HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO ĐỒ ÁN**

**MÔN: HỆ ĐIỀU HÀNH**

**ĐỒ ÁN 2:**

**TÌM HIỂU SYSTEM CALLS – CÁC THAO TÁC VỚI FILES**

**Giảng viên lý thuyết:**

**Lê Quốc Hòa**

**Giảng viên thực hành:**

**Chung Thùy Linh**

**Nhóm sinh viên:**

**Hoàng Hữu Minh An 20127102**

**Nguyễn Võ Minh Trí 20127364**

**Nguyễn Thị Minh Thảo 20127412**





## THÔNG TIN THÀNH VIÊN

Họ và tên	MSSV
Hoàng Hữu Minh An	20127102
Nguyễn Võ Minh Trí	20127364
Nguyễn Thị Minh Thảo	20127412

## MỤC LỤC

<b>1. TÌM HIỂU SƠ LƯỢC VỀ CÁC LỚP LIÊN QUAN ĐẾN QUẢN LÝ FILE .....</b>	<b>3</b>
1.1. SƠ LƯỢC VỀ FILESYS.H .....	3
1.2. SƠ LƯỢC VỀ OPENFILE.H .....	3
<b>2. CÀI ĐẶT SYSTEM CALL THAO TÁC VỚI FILE.....</b>	<b>4</b>
2.1. CÀI ĐẶT SYSTEM CALL CREATE FILE .....	4
2.2. CÀI ĐẶT SYSTEM CALL OPEN FILE VÀ CLOSE FILE .....	4
2.3. CÀI ĐẶT SYSTEM CALL READ FILE VÀ WRITE FILE .....	5
2.4. CÀI ĐẶT SYSTEM CALL SEEK FILE .....	7
2.5. CÀI ĐẶT SYSTEM CALL REMOVE FILE .....	7
<b>3. VIẾT CHƯƠNG TRÌNH NGƯỜI DÙNG .....</b>	<b>8</b>
3.1. VIẾT CHƯƠNG TRÌNH CREATEFILE .....	8
3.2. VIẾT CHƯƠNG TRÌNH CAT .....	8
3.3. VIẾT CHƯƠNG TRÌNH COPY .....	8
3.4. VIẾT CHƯƠNG TRÌNH DELETE .....	9
3.5. VIẾT CHƯƠNG TRÌNH CONCATENATE .....	9
<b>4. DEMO.....</b>	<b>10</b>
<b>5. TÀI LIỆU THAM KHẢO. ....</b>	<b>13</b>
<b>6. BẢNG PHÂN CÔNG CÔNG VIỆC .....</b>	<b>13</b>

## 1. Tìm hiểu sơ lược về các lớp liên quan đến quản lý file

### 1.1. Sơ lược về `filesystem.h`

- ✓ Là Cấu trúc dữ liệu đại diện cho hệ thống tệp Nachos.
- ✓ Hệ thống tệp là một tập hợp các tệp được lưu trữ trên đĩa, được tổ chức vào các thư mục. Các hoạt động trên hệ thống tệp phải thực hiện với "đặt tên" - tạo, mở và xóa tệp, cho một tên tệp văn bản. Hoạt động trên một cá nhân tệp "mở" (đọc, ghi, đóng) sẽ được tìm thấy trong `OpenFile` lớp (`openfile.h`).
- ✓ Ta xác định hai triển khai riêng biệt của hệ thống tệp. Phiên bản "STUB" chỉ định nghĩa lại hệ thống tệp Nachos hoạt động như các hoạt động trên hệ thống tệp UNIX gốc trên máy
- ✓ chạy mô phỏng Nachos. Phiên bản còn lại là hệ thống tệp "thực", được xây dựng dựa trên một trình mô phỏng đĩa. Đĩa được mô phỏng bằng UNIX gốc hệ thống tệp (trong tệp có tên "DISK").
- ✓ Trong triển khai thực tế, có hai cấu trúc dữ liệu chính được sử dụng trong hệ thống tệp. Có một thư mục "gốc" duy nhất, danh sách tất cả các tệp trong hệ thống tệp; không giống như UNIX, đường cơ sở hệ thống không cung cấp cấu trúc thư mục phân cấp. Ngoài ra, có một bitmap để phân bổ các khu vực đĩa. Cả thư mục gốc và bitmap đều là chính nó được lưu trữ dưới dạng tệp trong hệ thống tệp Nachos - điều này gây ra một điều thú vị sự cố bootstrap khi khởi chạy đĩa mô phỏng.
- Một số chỉnh sửa:
  - Tạo một hàm `int FindFreeSlot()`: dùng để tìm vị trí trống trong bảng mô tả `OpenFileID`.
  - Tạo `OpenFile **ListFile`; là một bảng mô tả danh sách các files đang mở.
- Vì `filesystem.h` có 2 phiên bản, nên phiên bản còn lại ta làm tương tự.
- Các hàm khởi tạo ở `FileSystem()` tạo ra 2 file `stdin` và `stdout`.
- Ở hàm hủy `~FileSystem()` hủy danh sách bảng mô tả `OpenFileID`.

### 1.2. Sơ lược về `openfile.h`

- ✓ Cấu trúc dữ liệu để mở, đóng, đọc và ghi vào các tệp riêng lẻ. Các hoạt động được hỗ trợ tương tự như những cái UNIX - gõ 'man open' vào lời nhắc UNIX. Có hai cách triển khai. Một là "STUB" trực tiếp biến các hoạt động tệp thành các hoạt động UNIX cơ bản. (xem bình luận trong `filesystem.h`).
- ✓ Khác là triển khai "thực", biến những các hoạt động vào đọc và ghi các yêu cầu khu vực đĩa. Trong quá trình triển khai cơ bản này của hệ thống tệp, chúng tôi không lo lắng về việc truy cập đồng thời vào hệ thống tệp bằng các chủ đề khác nhau.
- Một số chỉnh sửa:
  - Thêm 1 biến `char* filename` nhằm mục đích định danh đối tượng `OpenFile`. Tương tự `filesystem.h`, ta cũng thêm trên 2 phiên bản của `openfile.h`

## 2. Cài đặt system call thao tác với file

### 2.1. Cài đặt system call create file

Cài đặt system call `int Create(char *name)`. Create system call sẽ sử dụng Nachos `FileSystem Object` để tạo một file rỗng. Chú ý rằng filename đang ở trong user space, có nghĩa là buffer mà con trỏ trong user space trỏ tới phải được chuyển từ vùng nhớ user space tới vùng nhớ system space. System call Create trả về 0 nếu thành công và -1 nếu có lỗi

- Mục đích: Tạo file
- Cách cài đặt:
  - Đọc địa chỉ của tham số vào thanh ghi số 4.
  - Chép giá trị của thanh ghi số 4 từ user space sang system space (gọi hàm `User2System()`) chính là tên của file.
  - Kiểm tra tên file có NULL không.
  - Kiểm tra file đã được tạo ra trước đó hay chưa, nếu chưa tiến hành tạo file bằng cách gọi hàm `Create()` trong `FileSystemObject`.
  - Tạo file thành công trả về giá trị 0, ngược lại trả về giá trị -1. Giá trị trả về được lưu trong thanh ghi số 2.

### 2.2. Cài đặt system call open file và close file

Cài đặt system call `OpenFileID Open(char *name)` và `int Close(OpenFileID id)` (mở và đóng file)

#### ❖ System call Open File

- Mục đích: Mở file
- Nguyên lý hoạt động: Truyền vào tên file
  - Đọc địa chỉ của tên file từ thanh ghi số 4
  - Thực hiện sao chép giá trị ở số 4 từ vùng nhớ User sang System bằng phương thức `User2System`. Giá trị chép được chính là tên file mà người dùng yêu cầu.
  - Ở đây sử dụng một vòng lặp tiến hành kiểm tra trong 20 vị ta cấp phát để quản lý `openFileID` có thực sự đã mở trước đó hay không ? bởi vì ở lớp `openFile` ta để thêm một thuộc tính `char* fileName` nhằm mục khi khởi có giá trị trong `fileID` thì có thêm một `fileName` nhằm định danh chúng và lúc này ta tiến hành kiểm tra một lần từ đầu đến cuối xem có tên file nào đã trùng trước đó không nếu có thì tiến hành trả về ID đã mở trước đó xóa các con trỏ cấp phát và dùng syscall.
  - Tiến hành kiểm tra tên file mà người dùng nhập. Nếu tên file không hợp lệ thì thông báo ra dòng thông báo “Tên file không hợp lệ” và trả về giá trị -1, xóa bộ nhớ của `namefile`, tăng bộ đếm chương trình.
  - Mỗi tiến trình Read/Write sẽ được cấp một bảng mô tả file có kích thước là 2-20
    - Tìm vị trí file còn trống trong bảng mô tả file bằng hàm `FindFreeSlot()`
      - Kiểm tra điều kiện nếu bảng file đã đầy → thông báo ra màn hình “Bỏ nho dáy” và trả về giá trị -1 . xóa bộ nhớ của `namefile` và tăng bộ đếm chương .
      - Ta tiến hành mở file

- Mở file thành công truy cập vào thuộc tính fileName cấp bộ nhớ để chuẩn bị sao chép tên fileName đã truyền cho đối tượng mới mở
- Nếu thành công, tiến hành mở file bằng hàm được qua hồ trữ qua thư viện của HDH Linux.
  - Nếu mở file không thành công → Thông báo ra màn hình “Không tồn tại file” và trả về giá trị -1.
  - Ngược lại, thông báo ra màn hình “Mở file thành công”. Trả về vị trí hiện tại của file trong bảng mô tả file.
- Xóa bộ nhớ của namefile và tăng bộ đếm chương trình.
- System call Close
  - Mục đích: Đóng file đang mở
  - Nguyên lý hoạt động:
    - Đọc giá trị ID của file từ thanh ghi số 4
    - Kiểm tra id:
      - Nếu ID < 2 hoặc ID > 20: nghĩa là id truyền vào không nằm trong danh sách file (2 ≤ ID ≤ 20), báo lỗi, trả về lỗi -1 cho người dùng, tăng thanh ghi PC và kết thúc
    - Kiểm tra trạng thái của file
      - Nếu file chưa mở: Báo lỗi và trả về -1, tăng thanh ghi PC và kết thúc.
      - Nếu file đã mở: Giải phóng bộ nhớ của OpenFile\* trong danh sách file, gán vị trí ô nhớ đó trong danh sách file là NULL, trả kết quả 1 về cho người dùng, tăng thanh ghi PC và kết thúc.

### 2.3. Cài đặt system call read file và write file

Cài đặt system call `int Read(char *buffer, int size, OpenFileID id)` và `int Write(char *buffer, int size, OpenFileID id)`. Các system call đọc và ghi vào file với id cho trước. Bạn cần phải chuyển vùng nhớ giữa user space và system space, và cần phải phân biệt giữa Console IO (`OpenFileID 0, 1`) và File

- ❖ System call Read file: `int Read(char *buffer, int size, OpenFileID id)`:
  - Mục đích: Đọc byte from từ open file vào bufer với size và OpenFileID cho trước được lưu trong bảng mô tả và trả về số byte thực sự đã đọc
  - Ý tưởng cài đặt:
    - Trong phần OpenFile ta đã tạo ra một bảng mô tả gồm những file nào đã được mở với những id. Công việc ở đây ta chỉ cần lấy những id này kiểm tra có trong bảng mô tả hoặc là id là stdin hoặc stdout ta sẽ có những cách xử lý khác nhau. Với stdin ta tiến hành đọc những kí tự từ console và trả về số byte thực sự đọc được. Không đọc được file stdout sẽ trả về INT\_MIN
  - Cách cài đặt:
    - Lần lượt đọc các địa chỉ từ user space vào biến virtAdr, độ dài byte cần đọc vào buffersize và đọc openFileID vào fileId.
    - Trước khi để đọc file ta cần kiểm tra id file mà ta muốn đọc có nằm trong bảng mô tả hay không nếu không có trả về giá trị -1.

- Kiểm tra tại id file có tồn tại file đã mở chưa. Nếu chưa tồn tại thì trả về -1.
  - Nếu người dùng đọc file stdin thì ta tiến hành hành đọc từ bang phím bằng lớp synchConsoleIn để đọc các kí tự cuối cùng dùng hàm System2User để trả về số byte có thể đọc được.
  - Tiến hành truyền địa chỉ user space xuống cho kernel lúc này dùng phương thức read() có trong lớp OpenFile để thực hiện đọc file.
  - Lưu số byte đọc được n-buf và tiến hành kiểm tra nếu số byte đọc được lớn 0 thì việc đọc file thành công, sau đó chuyển buffer đã đọc được về cho user bằng System2User và trả về số byte đọc được.
  - Các trường hợp còn lại thì trả về số INT\_MIN để thông báo việc đọc không thành công
  - Kết thúc ta xóa các con trỏ đã gọi trước đó và thoát để tránh rò rỉ bộ nhớ
- ❖ System call Write File: int Write(char \*buffer, int size, OpenFileID id) tương tự như hàm read file
- Mục đích: Ghi số byte from từ open file vào file OpenFileID cho trước được lưu trong bảng mô tả và trả về số byte thực sự đã ghi được
  - Ý tưởng cài đặt:
    - Ở trong phần OpenFile ta đã tạo ra một bảng mô tả gồm những file nào đã được mở với những id. Công việc ở đây ta chỉ cần lấy những id này kiểm có trong bảng mô tả hoặc là id là stdin hoặc stdout ta sẽ có những cách xử lý khác nhau. Với stdout ta tiến hành in kí tự ra console và trả về số byte thực sự ghi được. Không đọc được file stdin sẽ trả về INT\_MIN.
  - Cách cài đặt:
    - Lần lượt đọc các địa chỉ từ user space vào biến virtAdr, độ dài byte cần đọc vào buffersize và đọc openFileID vào fileId.
    - Trước khi để đọc file ta cần kiểm tra id file mà ta muốn đọc có nằm trong bảng mô tả hay không nếu không có trả về giá trị -1.
    - Kiểm tra tại id file có tồn tại file đã mở chưa. Nếu chưa tồn tại thì trả về -1.
    - Nếu người dùng đọc file stdin thì ta tiến hành hành trả về -1 có nghĩa là không đọc file được.
    - Tiến hành truyền địa chỉ user space xuống cho kernel lúc này dùng phương thức write() có trong lớp OpenFile để thực hiện ghi file.
    - Lưu số byte ghi được n-buf và tiến hành kiểm tra nếu số byte đọc được lớn 0 thì việc ghi file thành công, sau đó chuyển buffer đã đọc được về cho user bằng System2User và trả về số byte ghi được.
    - Xét trường hợp là file stdout ta tiến hành lần lượt ghi ra các kí tự ra màn hình và trả về số byte mà thực sự ghi được.
    - Kết thúc ta xóa các con trỏ đã gọi trước đó và thoát để tránh rò rỉ bộ nhớ.



## 2.4. Cài đặt system call seek file

Cài đặt system call `int Seek(int position, OpenFileID id)`. `Seek` sẽ phải chuyển con trỏ tới vị trí thích hợp. `Position` lưu vị trí cần chuyển tới, nếu `pos = -1` thì di chuyển đến cuối file. Trả về vị trí thực sự trong file nếu thành công và `-1` nếu bị lỗi. Gọi `Seek` trên console phải báo lỗi.

- Mục đích: Dời vị trí `seekPosition` trong lớp `OpenFile` đến vị trí byte thích hợp bằng cách thay đổi giá thuộc tính `seekPosition`
- Ý tưởng cài đặt: Dựa vào bảng mô tả `Open file id` ta sẽ biết file nào đang được mở và sau đó kiểm tra file có tồn tại hay chưa, sau tiến hành kiểm tra cách điều kiện để dời vị trí nếu các file cần dời là `stdin` hoặc `stdout` thì không cần dời nếu là file `pos = -1` thì chỉ trả về độ dài
- Cách cài đặt:
  - Đọc các vị trí `pos` cần dời và file id
  - Tiến hành kiểm tra file id có nằm trong bản mô tả hay không
  - Kiểm tra vị trí file id có roongx hấy không nếu không tồn tại trả về `-1`
  - Kiểm tra nếu là file `stdin` `stdout` thì cũng trả về `-1`
  - Kiểm tra xem vị trí `pos = -1` thì trả về độ dài file bằng phương thức `Length()` có sẵn trong lớp `OpenFile`
  - Kiểm tra vị trí con trỏ cần có nằm trong độ dài file hoặc nằm ngoài độ dời là `< -1` thì trả về `-1`
  - Nếu không có tồn tại lỗi nào nói trên tiến hành cập nhật vị trí còn trỏ bằng hàm `Seek(pos)` có trong lớp `OpenFile` về trả về vị con trỏ được dời

## 2.5. Cài đặt system call remove file

Cài đặt system call `int Remove(char *name)`. `Remove` system call sẽ sử dụng `Nachos FileSystem Object` để xóa file. Chú ý: cần kiểm tra file có đang mở hay không trước khi xóa.

- Mục đích: Xóa file
- Cách hoạt động:
  - Đọc địa chỉ của tham số vào thanh ghi số 4.
  - Chép giá trị của thanh ghi số 4 từ user space sang system space (gọi hàm `Use2System()`) chính là tên của file.
  - Kiểm tra tên file có NULL không.
  - Sử dụng một vòng lặp để kiểm tra một từ đầu đến cuối đang mở file hay không bằng bảng mô tả `openFileID` dưới định danh tên mà ta đã tạo ở syscall `open` cũng như thế ta tiến hành kiểm tra từ đầu đến cuối nếu tên file mà ta muốn `remove` có trong trong bảng mô tả đang thì dùng syscall và trả về vị giá trị `-1` và thoát
  - Bằng cách kiểm tra file đó có nằm trong `ListFile[]` không, ta có thể biết được file mà người dùng muốn xóa có đang mở hay không nếu có trả về `-1` kết thúc chương trình => Không thể `remove`.
  - Nếu không, tiến hành xóa file bằng cách gọi hàm `Remove()` trong `FileSystemObject`.
  - Xóa file thành công trả về giá trị 0, ngược lại trả về giá trị `-1`. Giá trị trả về được lưu trong thanh ghi số 2.

### 3. Viết chương trình người dùng

#### 3.1. Viết chương trình createfile

Để kiểm tra system call *Create*. Bạn sẽ dùng tên file cố định, hoặc cho người dùng nhập vào từ console từ *ReadString*.

- Chương trình tạo file.
- Cách thức hoạt động:
  - Tiến hành đọc tên file do người dùng muốn tạo.
  - Gọi system call *Create()*.
  - Kiểm tra file có tạo thành công không
    - Thành công => In ra thông báo “Create file succesfully”
    - Không thành công => In ra thông báo “Create file failed”.

#### 3.2. Viết chương trình cat

Yêu cầu nhập filename, rồi hiển thị nội dung của file đó.

- Chương trình cat
- Cách thức hoạt động:
  - Tiến hành đọc tên file muốn hiển thị nội dung.
  - Mở file, kiểm tra mở file có bị lỗi không. Nếu không, seek đến cuối file để lấy độ dài của file. Sau đó, seek lại đầu file để bằng đầu read.
  - Tiến hành read, kết hợp print từng kí tự cho đến hết file.

#### 3.3. Viết chương trình copy

Yêu cầu nhập tên file nguồn và file đích và thực hiện copy.

- Mục đích: Copy nội dung từ file nguồn sang file đích.
- Cách thức cài đặt:
  - Nhập tên file nguồn và file đích
  - Mở 2 file nguồn và đích thông qua tên mà người dùng nhập vào
  - Kiểm tra *openFileID* thu được từ 2 tiến trình mở file:
    - Nếu *ID\_SouFile* == -1 hoặc *ID\_DesFile* == -1: In ra màn hình “Cannot open source file!” hoặc “Cannot open destination file!”. Gọi hàm *Close()* để đóng 2 file lại.
    - Nếu *ID\_SouFile* != -1 và *ID\_DesFile* != -1:
      - Gọi hàm *Seek(-1, ID\_SouFile)* để lấy kích thước tối đa của source file (*lengthFile*) và *Seek(0, ID\_SouFile)*, *Seek(0, DesFile)* để đưa con trỏ về đầu 2 file.
      - Thực hiện quá trình đọc từng byte từ source file vào biến a thông qua hàm *Read(&a, 1, ID\_SouFile)*, sau đó ghi từng byte vào file destination file thông qua hàm *Write(&a, 1, DesFile)*. Quá trình này sẽ lặp lại, mỗi lần lặp sẽ giảm giá trị *lengthFile* đi 1, lặp đến khi *lengthFile* <= 0 thì đóng 2 file nguồn và đích lại.
- Gọi *Halt()* để tắt máy.



### 3.4. Viết chương trình delete

Để kiểm tra system call Remove.

- Mục đích: Kiểm tra system call Remove
- Cách thức cài đặt:
  - Yêu cầu người dùng nhập tên file muốn xóa.
  - Gọi system call Remove để xóa file theo tên mà người dùng nhập.
  - Kiểm tra việc xóa file có thành công hay không.
    - Thành công => In ra “Delete file succesfully”
    - Không thành công => In ra “Delete file failed”

### 3.5. Viết chương trình concatenate

Để nối nội dung của 2 file, yêu cầu nhập tên file nguồn 1 và file nguồn 2.

- Mục đích: Nối nội dung của 2 file
- Nguyên lý hoạt động:
  - Mở 2 file nguồn thông qua tên mà người dùng nhập vào.
  - Kiểm tra openFileID thu được từ 2 tiến trình mở file
    - Nếu ID\_SrcFile1 == -1 hoặc ID\_SrcFile2 == -1: In ra màn hình “Cannot open file source!”. Gọi hàm Close() để đóng 2 file lại
    - Nếu ID\_SrcFile1 != -1 và ID\_SrcFile2 != -1:
      - Tạo file đích (file giữ nội dung được nối từ nội dung của 2 file nguồn) thông qua tên mà người dùng nhập vào. Sau đó mở file đích
      - Gọi hàm Seek(-1, ID\_SrcFile1) và Seek(-1, ID\_SrcFile2) để lấy kích thước tối đa của source file (lengthFile1 và lengthFile2). Kích thước tối đa của destination file (lengthFileDes) bằng tổng kích thước của 2 file nguồn. Gọi hàm Seek(0, ID\_SrcFile1), Seek(0, ID\_SrcFile2) và Seek(0, ID\_DesFile) để đưa con trỏ về đầu 3 file.
      - Thực hiện quá trình đọc nội dung từ file nguồn thứ nhất và ghi sang file đích. Sau đó tiếp tục đọc nội dung từ file nguồn thứ hai và ghi sang file đích
      - Sau khi copy thành công thì thông báo cho người dùng “Concatenate successfully” và đóng 3 file lại.
  - Gọi Halt() để tắt máy.

## 4. Demo

### 1. Chương trình Create.

- Tạo một file mới có tên là hello.txt

```
tri@ubuntu: ~/nachos/NachOS-4.0/code/test
File Edit View Search Terminal Help
tri@ubuntu:~$ ../build.linux/nachos -x createfile
bash: ../build.linux/nachos: No such file or directory
tri@ubuntu:~$ cd nachos/NachOS-4.0/code/test
tri@ubuntu:~/nachos/NachOS-4.0/code/test$ ../build.linux/nachos -x createfile
----- Create Program -----
File name: hello.txt
Create file sucessfully

Machine halting!

Ticks: total 276107554, idle 276104853, system 2650, user 51
Disk I/O: reads 0, writes 0
Console I/O: reads 10, writes 69
Paging: faults 0
Network I/O: packets received 0, sent 0
```

### 2. Chương trình Cat.

- Hiển thị nội dung trong file hello.txt

```
tri@ubuntu: ~/nachos/NachOS-4.0/code/test
File Edit View Search Terminal Help
tri@ubuntu:~/nachos/NachOS-4.0/code/test$ ../build.linux/nachos -x cat
----- Cat Program -----
Enter File Name: hello.txt
Mo file thanh cong
37-> File content:
Hello everyone. We are FIT HCMUser ^^
Dong file thanh cong

Machine halting!

Ticks: total 573393080, idle 573387753, system 4160, user 1167
Disk I/O: reads 0, writes 0
Console I/O: reads 10, writes 111
Paging: faults 0
Network I/O: packets received 0, sent 0
```

### 3. Chương trình Copy.

- Copy nội dung trong file group.txt sang file copyfile.txt

```

tri@ubuntu: ~/nachos/NachOS-4.0/code/test
File Edit View Search Terminal Help
tri@ubuntu:~/nachos/NachOS-4.0/code/test$ ../build.linux/nachos -x copy
----- Copy Program -----
Enter destination file: copyfile.txt
Enter source file: group.txt
Mo file thanh cong
Mo file thanh cong
Copy file successfully
Dong file thanh cong

Dong file thanh cong

Machine halting!

Ticks: total 1075166413, idle 1075162275, system 4000, user 138
Disk I/O: reads 0, writes 0
Console I/O: reads 23, writes 97
Paging: faults 0
Network I/O: packets received 0, sent 0

```

- Nội dung file copyfile.txt sau khi thực hiện chương trình copy.

```

Open  copyfile.txt  Save  ~/nachos/NachOS-4.0/code/test
Group Member:
Hoang Huu Minh An      - 20127102
Nguyen Vo Minh Tri     - 20127364
Nguyen Thi Minh Thao   - 20127412

L... Plain Text  Tab Width: 8  Ln 1, Col 1  INS

```

#### 4. Chương trình Concatenate.

- Nối nội dung file hello.txt và group.txt

```

tri@ubuntu: ~/nachos/NachOS-4.0/code/test
File Edit View Search Terminal Help
tri@ubuntu:~/nachos/NachOS-4.0/code/test$ ../build.linux/nachos -x concatenate
----- Concatenate Program -----
Enter source file 1: hello.txt
Enter source file 2: group.txt
Mo file thanh cong
Mo file thanh cong
Enter destination file : concatenatefile.txt
Mo file thanh cong
Concatenate successfully!!!
Dong file thanh cong

Dong file thanh cong

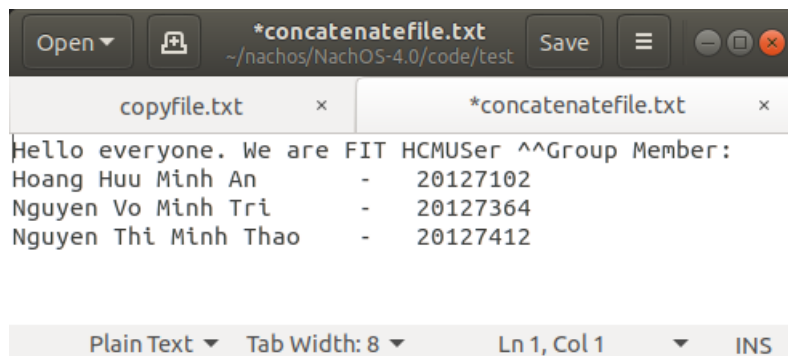
Dong file thanh cong

Machine halting!

Ticks: total 1033317626, idle 1033311247, system 6150, user 229
Disk I/O: reads 0, writes 0
Console I/O: reads 40, writes 144
Paging: faults 0
Network I/O: packets received 0, sent 0

```

➤ Kết quả sau khi nối



```

Hello everyone. We are FIT HCMUser ^^Group Member:
Hoang Huu Minh An      - 20127102
Nguyen Vo Minh Tri     - 20127364
Nguyen Thi Minh Thao   - 20127412

```

## 5. Chương trình Delete.

➤ Xóa file hello.txt

```

tri@ubuntu: ~/nachos/NachOS-4.0/code/test
File Edit View Search Terminal Help
tri@ubuntu:~/nachos/NachOS-4.0/code/test$ ../build.linux/nachos -x delete
----- Delete Program -----
Enter file name: hello.txt
Delete file succesfully

Machine halting!

Ticks: total 336712356, idle 336709523, system 2780, user 53
Disk I/O: reads 0, writes 0
Console I/O: reads 10, writes 73
Paging: faults 0
Network I/O: packets received 0, sent 0

```

## 5. TÀI LIỆU THAM KHẢO.

- Tài liệu do giảng viên cung cấp.
- Chuỗi video hướng dẫn Lập trình Nachos HCMUS - Nguyễn Thành Chung.

## 6. Bảng phân công công việc

Họ và tên	MSSV	Nhiệm vụ	Hoàn thành
Hoàng Hữu Minh An	20127102	System call read, write, seek và chương trình cat.	100%
Nguyễn Võ Minh Trí	20127364	System call create, remove và chương trình copy, concatenate.	100%
Nguyễn Thị Minh Thảo	20127412	System call open, close và chương trình create, delete.	100%

✎ Chúng em xin cảm ơn thầy cô đã đọc bài báo cáo này

✎ Em xin chân thành cảm ơn quý thầy, cô đã tận tình chỉ dạy và trang bị cho em những kiến thức cần thiết về bộ môn Hệ điều hành trong suốt học kỳ qua, làm nền tảng cho em có thể hoàn thành được đồ án cuối kỳ lần này. Em xin kính chúc quý thầy, cô thật nhiều sức khỏe và em mong những học kỳ tới có thể được đồng hành cùng quý thầy cô!