

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

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



BÁO CÁO ĐỒ ÁN 1

Môn: Mạng Máy Tính

Giáo Viên Hướng Dẫn

Thầy LÊ GIANG THANH
Thầy NGUYỄN THANH QUÂN
Thầy LÊ HÀ MINH

LÊ NGÔ SONG CÁT- BUI NGOC KIỀU NHI

MỤC LỤC

I. THÔNG TIN THÀNH VIÊN	2
II. PHÂN CHIA CÔNG VIỆC VÀ MỨC ĐỘ HOÀN THÀNH.....	2
III. TỔNG QUAN CÀI ĐẶT	3
1. Cài đặt trong một số file có sẵn:	3
2. Syscall và hàm hỗ trợ:	3
3. Các giá trị thanh ghi:	4
IV. CÁC SYSTEMCALL THAO TÁC VỚI FILES	4
1. Syscall Create:	4
2. Syscall Open:	5
3. Syscall Close:	5
4. Syscall Read:.....	5
5. Syscall Write:	6
6. Syscall Seek:	6
7. Syscall Remove :.....	6
8. Syscall SocketTCP:	7
9. Syscall Connect:	7
10. Syscall Send:	7
11. Syscall Receive:	8
12. Syscall CloseSocket:.....	8
V. CÁC CHƯƠNG TRÌNH THAO TÁC VỚI FILES	8
1. Chương trình createfile:	8
2. Chương trình cat:.....	10
3. Chương trình copy:.....	11
4. Chương trình delete:.....	13
5. Chương trình concatenate:	14
6. Chương trình echoclient_client:	16
7. Chương trình fileclient:	17
VI. TÀI LIỆU THAM KHẢO	19

BÁO CÁO ĐỒ ÁN 1

I. THÔNG TIN THÀNH VIÊN

STT	MSSV	Họ và tên
1	21127495	Lê Ngô Song Cát
2	21127659	Bùi Ngọc Kiều Nhi

II. PHÂN CHIA CÔNG VIỆC VÀ MỨC ĐỘ HOÀN THÀNH

STT	Hàm	Người thực hiện	Tự đánh giá mức độ hoàn thành
1	Syscall Create	21127659	100%
2	Syscall Open, Close	21127659	100%
3	Syscall Read, Write	21127495	100%
4	Syscall Seek	21127495	100%
5	Syscall Remove	21127495	100%
6	Syscall SocketTCP	21127495	100%
7	Syscall Connect	21127495	100%
8	Syscall Send	21127659	100%
9	Syscall Receive	21127659	100%
10	Syscall Close	21127495	100%
	Report	21127495	100%
		21127659	

III. TỔNG QUAN CÀI ĐẶT

1. Cài đặt trong một số file có sẵn:

- exception.cc: xử lý SystemCall và các exception khác ở mức user, ví dụ như lỗi trang, ở đây chỉ có 'halt' SystemCall được viết.
- syscal.h: SystemCall interface: các thủ tục ở kernel mà chương trình người dùng có thể gọi.
- ksyscall.h: Cài đặt các lời gọi hệ thống để chương trình thực hiện.

2. Syscall và hàm hỗ trợ:

a) SC_ReadString

- Input: int length.
- Output: NULL.
- Công dụng: Đọc ký tự hoặc chuỗi ký tự mà người dùng nhập vào từ bàn phím.
- Cài đặt:
 - Bước 1: Cấp phát một vùng nhớ có kích thước length + 1 cho biến buffer kiểu char*, nếu không thể cấp phát (buffer = NULL) thì return, ngược lại thực hiện bước 2.
 - Bước 2: Duyệt từng ký tự có độ dài length mà người dùng nhập vào, cho đến khi gặp ký tự '\0' hoặc '\n'.
 - Bước 3: Kiểm tra trường hợp tràn buffer.
 - Bước 4: Đọc dữ liệu từ thanh ghi 4 và lưu vào biến virtAddr, gọi hàm System2User() để tạo vùng nhớ ảo tại virtAddr, với chuỗi ký tự buffer và kích thước length.

b) SC_PrintString:

- Input: NULL.
- Output: Chuỗi ký tự.
- Công dụng: In chuỗi ký tự ra màn hình.
- Cách cài đặt:
 - Bước 1: Đọc dữ liệu từ thanh ghi 4 và lưu vào biến virtAddr.
 - Bước 2: Dùng hàm User2System() để lấy từ vùng nhớ ảo virtAddr chuỗi ký tự mà ta đã đọc vào.
 - Bước 3: Truyền từng ký tự lên hệ thống để in ra màn hình.

c) Hàm increasePC():

- Công dụng: Để tránh tạo thành vòng lặp vô hạn sau mỗi lần system call của chương trình khi máy tính nạp lệnh tại một chỗ.
- Ý tưởng: Tăng giá trị Program Counter (PC) sau mỗi System Call (SC) để khi nạp lệnh tiếp theo thì chương trình không lặp lại SC cũ mà thực hiện lệnh mới.
- Cài đặt: Lưu giá trị của PC hiện tại thành PC trước, nạp giá trị của lệnh cần thực hiện cho PC hiện tại, nạp giá trị tiếp sau đó cho PC kế.

d) Hàm System2User():

- Sao chép dữ liệu từ hệ thống đưa cho người dùng

e) Hàm User2System():

- Sao chép dữ liệu từ người dùng đưa cho hệ thống.

f) Lớp FileSystem:

- Thêm hàm tạo OpenFile(int f, char* fileName, int fileType) vào lớp OpenFile vì lớp OpenFile chứa thêm thông tin tên file (char* fileName) và chế độ mở file (int fileType)
- Trong lớp FileSystem, tạo thêm thuộc tính là openingFile là FileTable chứa thông tin các file đang được mở.
- Do có thêm dữ liệu chế độ đọc file nên ta ghi đè hàm Open để tạo nên tham số fileType.
- Hàm int FileDescriptorFile(): duyệt qua FileTable để tìm xem có vị trống trong bảng để mở thêm file mới không. Nếu có thì trả về index trống, không thì trả về -1.
- Hàm OpenFile* GetFileDescriptor (int index): trả về con trỏ OpenFile của file đang được mở ở vị trí 'index' nếu có hoặc trả về NULL.
- Hàm OpenFile* GetFileDescriptor (char* fileName): trả về con trỏ OpenFile của file 'fileName' đang được mở nếu có hoặc trả về NULL.
- Hàm int GetFileDescriptorID(char* fileName): trả về giá trị fileID của file 'fileName' tương ứng nếu có hoặc trả về -1.
- Hàm OpenFile* AssignFileDescriptor(int index, char* fileName, int filetype): tạo fileID mới với tên file là 'fileName' và mở ở chế độ 'fileType'.
- Hàm bool RemoveFileDescriptor(int index): xóa con trỏ OpenFile ở vị trí 'index' (đóng file).

g) Lớp Socket Table:

- Lớp SocketTable gồm hai thuộc tính là mảng bool _inUse chứa thông tin socket đang được mở, mảng integer _socketID chứa các giá trị socketID.
- Hàm tạo socket SocketTCP() : dùng biến freeIndex duyệt mảng tìm vị trí trống để chứa giá trị socketID. Nếu tìm thấy index trống thì tạo socket mới bằng hàm socket(AF_INET, SOCK_STREAM, 0) và thay đổi giá trị _inUse tại index đó thành 1: đã có socketID được tạo ở vị trí đó và trả về giá trị socketID đó.
- Hàm đóng socket CloseSock(int socketID): đóng socket bằng hàm hỗ trợ của thư viện và đi đến index chứa giá trị socketID đó trong SocketTable để xóa giá trị.

3. Các giá trị thanh ghi:

- Register 2: Lưu mã syscall và lưu kết quả trả về của mỗi syscall nếu có.
- Register 4: Lưu tham số thứ nhất.
- Register 5: Lưu tham số thứ hai.
- Register 6: Lưu tham số thứ ba.
- Register 7: Lưu tham số thứ tư.

IV. CÁC SYSTEMCALL THAO TÁC VỚI FILES

1. Syscall Create:

- Mô tả cài đặt:
 - Input: NULL.
 - Return: 0 (thành công) hoặc -1 (thất bại)
 - Công dụng: Tạo một file rỗng.
- Cách cài đặt:

- Bước 1: Đọc dữ liệu từ thanh ghi 4 và lưu vào biến virtAddr.
- Bước 2: Từ virtAddr, chuyển đổi vùng nhớ từ System sang User và lưu vào biến fileName.
- Bước 3: Kiểm tra tính hợp lệ của fileName, nếu không hợp lệ thì đến bước 4.
- Bước 4: Nếu thỏa mãn các ràng buộc thì file được tạo với tên là fileName.
- Bước 5: Trả về giá trị -1 nếu file không được tạo, 0 nếu tạo file thành công.

2. Syscall Open:

- Mô tả cài đặt:
 - Input: NULL.
 - Return: 0 (thành công) hoặc -1 (thất bại)
 - Công dụng: Mở file được yêu cầu.
- Cách cài đặt:
 - Bước 1: Đọc dữ liệu từ thanh ghi 4, 5 và lưu vào 2 biến tương ứng là virtAddr và fileType.
 - Bước 2: Từ virtAddr, chuyển đổi vùng nhớ từ System sang User và lưu vào biến fileName.
 - Bước 3: Kiểm tra tính hợp lệ của fileName, nếu không hợp lệ thì return -1.
 - Bước 4: Lấy địa chỉ id của file nếu file đã được mở, nếu không thì chuyển sang bước 5.
 - Bước 5: Tìm vị trí trống trong bảng (gồm 20 phần tử chứa danh sách file opening), nếu còn vị trí trống để cấp cho file thì return vị trí đó nếu không thì trả về -1.
 - Bước 6: Mở file, có hai chế độ mở file là đọc-ghi file và chỉ được phép đọc. Trong quá trình mở, nếu không thành công thì return -1, nếu thành công thì return 0.

3. Syscall Close:

- Mô tả cài đặt:
 - Input: NULL.
 - Return: 0 (thành công) hoặc -1 (thất bại)
 - Công dụng: Đóng file.
- Cách cài đặt:
 - Bước 1: Đọc fileId từ thanh ghi 4.
 - Bước 2: Kiểm tra fileId có hợp lệ (nằm trong khoảng [0, MAX_FILES = 20]), nếu có thì đóng file và return 0, nếu không đóng được hoặc fileId không hợp lệ thì return -1.

4. Syscall Read:

- Mô tả cài đặt:
 - Input: int virtAddr, int length, OpenFileID fileID.
 - Return: 0 (thành công) hoặc -1 (thất bại)
 - Công dụng: Đọc các ký tự trong file vào buffer với kích thước size.
- Cách cài đặt:
 - Bước 1: Khởi tạo biến cnt (số bytes đọc được) = -1.

- Bước 2: Xét giá trị fileID. Nếu fileID = 0 thì thực hiện tương tự như cách ReadString. Nếu fileID = 1 thì bỏ qua vì đây là Console Output. Các trường hợp còn lại thì đến bước 3.
- Bước 3: Cấp phát vùng nhớ có kích thước length + 1 cho biến buffer kiểu char*, cnt sẽ được tính bằng cách đọc buffer với kích thước length ở fileID.
- Bước 4: Nếu cnt <= 0 thì báo lỗi file rỗng, nếu không thì buffer[cnt] = 0 và đưa dữ liệu buffer vào vùng nhớ ảo virtAddr với kích thước cnt.
- Bước 5: Return cnt.

5. Syscall Write:

- Mô tả cài đặt:
 - Input: int virtAddr, int length, OpenFileID fileID.
 - Return: 0 (thành công) hoặc -1 (thất bại)
 - Công dụng: Ghi vào file.
- Cách cài đặt:
 - Bước 1: Khởi tạo biến cnt (số bytes đọc được) = -1.
 - Bước 2: Xét giá trị fileID. Nếu fileID = 0 thì bỏ qua vì đây là Console Input. Nếu fileID = 1 thì thực hiện tương tự như cách PrintString. Các trường hợp còn lại thì đến bước 3.
 - Bước 3: Lấy dữ liệu từ vùng nhớ virtAddr (dùng User2System), độ dài length và gán vào biến buffer kiểu char*, và xác định file mở để ghi thông qua fileID.
 - Bước 4: Nếu không thể lấy được dữ liệu từ vùng nhớ (buffer = NULL) thì return -1, nếu được thì kiểm tra kiểu file để mở, nếu mở ở chế độ chỉ đọc thì return -1.
 - Bước 5: Ghi dữ liệu từ buffer vào file nếu hợp lệ, ngược lại báo lỗi.
 - Bước 6: Return cnt.

6. Syscall Seek:

- Mô tả cài đặt:
 - Input: int pos, int fileID.
 - Return: 0 (thành công) hoặc -1 (thất bại)
 - Công dụng: Di chuyển con trỏ tệp đến một vị trí pos được chỉ định.
- Cách cài đặt:
 - Bước 1: Mở fileID, kiểm tra con trỏ file có tồn tại hay không, nếu không (filePtr = NULL) thì return -1, nếu hợp lệ thì đến bước 2.
 - Bước 2: Kiểm tra pos:
 - Nếu pos = -1 thì pos = fileLength (cuối file).
 - Nếu pos không nằm trong khoảng [0, fileLength] thì báo lỗi không thể tìm thấy, và pos = -1, nếu nằm trong khoảng đó thì đặt con trỏ file ở vị trí pos.
 - Bước 3: Return pos.

7. Syscall Remove :

- Mô tả cài đặt :
 - Input: NULL.

- Return: 0 (thành công) hoặc -1 (thất bại)
- Công dụng: Sử dụng Nachos FileSystem Object để xóa file.
- Cách cài đặt:
 - Bước 1: Đọc tham số ở thanh ghi 4 và lưu vào biến virtAddr
 - Bước 2: Lấy dữ liệu từ vùng nhớ virtAddr (dùng User2System), độ dài MAXLENGTH = 1024 và gán vào biến fileName kiểu char*.
 - Bước 3: Kiểm tra fileName rỗng thì return, nếu không thì kiểm tra file có đang mở hay không, nếu có thì báo lỗi, ngược lại thì xóa file.

8. Syscall SocketTCP:

- Mô tả cài đặt:
 - Input: NULL.
 - Return: socketID (thành công) hoặc -1 (thất bại)
 - Công dụng: Tạo socket để kết nối.
- Cách cài đặt:
 - Bước 1: Dùng hàm SocketTCP() đã cài đặt trong SocketTable để tạo một socket mới và trả về giá trị socketID nếu thành công hoặc -1 nếu không thể tạo socket.
 - Bước 2: Nếu tạo không thành công, ghi vào thanh ghi kết quả giá trị -1. Ngược lại, ghi vào thanh ghi kết quả giá trị socketID vừa được tạo.

9. Syscall Connect:

- Mô tả cài đặt:
 - Input: int socketID, char *ip, int port.
 - Return: 0 (thành công) hoặc -1 (thất bại)
 - Công dụng: Kết nối đến server theo thông tin địa chỉ IP và port được truyền vào.
- Cách cài đặt:
 - Bước 1: Đọc giá trị socketID từ thanh ghi thứ 4, địa chỉ chuỗi chứa IP của server ở thanh ghi thứ 5 và cổng port để kết nối từ thanh ghi thứ 6.
 - Bước 2: Đọc giá trị của chuỗi chứa IP của server bằng hàm User2System.
 - Bước 3: Kết nối với server với địa chỉ IP đã đọc được ở cổng port bằng hàm connect() được hỗ trợ bởi thư viện <sys/socket.h>.
 - Bước 4: Ghi vào thanh ghi kết quả -1 nếu kết nối thất bại và 0 nếu kết nối thành công.

10. Syscall Send:

- Mô tả cài đặt:
 - Input: int socketID, char *buffer, int len.
 - Return: Số byte gửi đi (thành công) hoặc -1 (thất bại)
 - Công dụng: Gửi dữ liệu từ socket.
- Cách cài đặt:
 - Bước 1: Đọc giá trị socketID từ thanh ghi thứ 4, địa chỉ chuỗi chứa thông tin cần gửi ở thanh ghi thứ 5 và độ dài chuỗi đó 'length' từ thanh ghi thứ 6.
 - Bước 2: Đọc chuỗi cần gửi đi bằng hàm User2System.

- Bước 3: Gửi đi dữ liệu truyền vào bằng hàm send() được hỗ trợ bởi thư viện <sys/socket.h> và lưu lại số byte đã gửi được vào biến byteSent.
- Bước 4: Ghi vào thanh ghi kết quả số byte đã gửi đi được hoặc -1 nếu gửi không thành công.

11. Syscall Receive:

- Mô tả cài đặt:
 - Input: int socketID, char *buffer, int len.
 - Return: Số byte nhận về (thành công) , 0 (kết nối bị đóng) hoặc -1 (thất bại)
 - Công dụng: Nhận dữ liệu từ socket.
- Cách cài đặt:
 - Bước 1: Đọc giá trị socketID từ thanh ghi thứ 4, địa chỉ chuỗi chứa thông tin cần gửi ở thanh ghi thứ 5 và độ dài chuỗi đó 'length' từ thanh ghi thứ 6.
 - Bước 2: Đọc chuỗi truyền vào bằng hàm User2System.
 - Bước 3: Đọc dữ liệu được gửi từ socket vào chuỗi buffer bằng hàm read() được hỗ trợ bởi thư viện <sys/socket.h> và lưu lại số byte đã đọc được vào biến bytesRead.
 - Bước 4: Ghi vào thanh ghi kết quả số byte đã đọc được hoặc -1 nếu gửi không thành công.

12. Syscall CloseSocket:

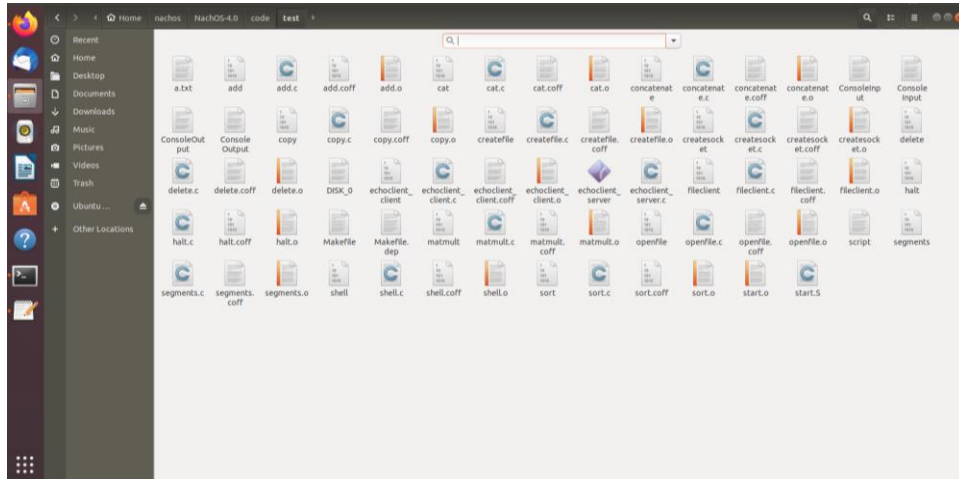
- Mô tả cài đặt:
 - Input: int socketID.
 - Return: 0 (thành công) hoặc -1 (thất bại)
 - Công dụng: Đóng socket.
- Cách cài đặt:
 - Bước 1: Đọc giá trị socketID được truyền vào từ thanh ghi thứ 4.
 - Bước 2: Kiểm tra nếu giá trị được truyền vào nhỏ hơn 0 thì ghi -1 vào thanh ghi kết quả.
 - Bước 3: Dùng hàm Close của lớp SocketTable đã được cài đặt ở trên để đóng socket.
 - Bước 4: Ghi vào thanh ghi kết quả -1 nếu đóng không thành công và 0 nếu đóng thành công.

V. CÁC CHƯƠNG TRÌNH THAO TÁC VỚI FILES

1. Chương trình createfile:

- Yêu cầu: Viết chương trình **createfile** để kiểm tra SystemCall Create, yêu cầu nhập tên file muốn tạo.
- Cách thực hiện:
 - Bước 1: Người dùng nhập tên file cần tạo.
 - Bước 2: Kiểm tra tên file đã tồn tại hay chưa. Nếu đã có rồi thì yêu cầu nhập tên file khác, nếu chưa thì chuyển sang bước 3.
 - Bước 3: Tạo file.

- Hình ảnh demo chương trình: Tạo file hello.txt trong thư mục test.
 - Ban đầu, trong thư mục test không có file hello.txt



- Chạy chương trình **createfile** và nhập tên file cần tạo là hello.txt

```

Terminal
File Edit View Search Terminal Help
songcat@songcat-virtual-machine:~/nachos/NachOS-4.0/code/test$ ../build.linux/nachos -x createfile
Input file name to create
hello.txt
    
```

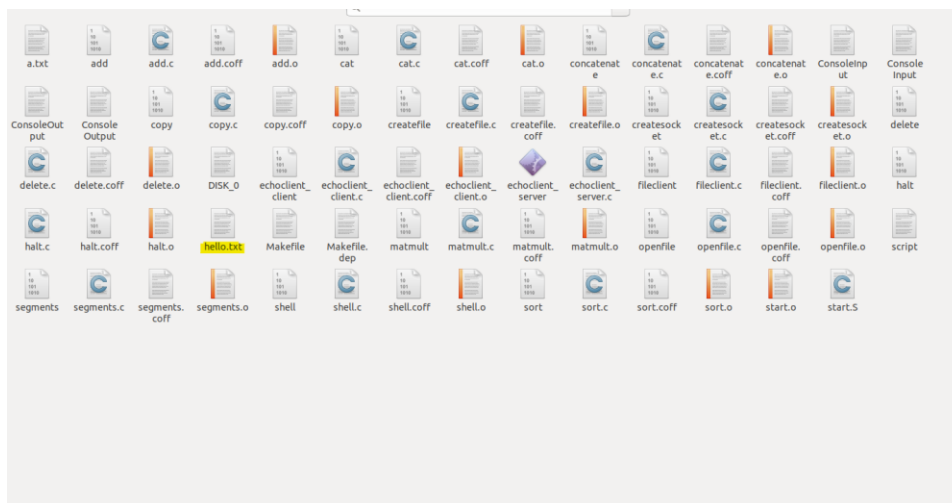
- Chương trình kết thúc và thư mục test có thêm file hello.txt rồi

```

Created file, shutting down...

Shutdown, initiated by user program.Machine halting!

Ticks: total 355580529, idle 355577893, system 2580, user 56
Disk I/O: reads 0, writes 0
Console I/O: reads 10, writes 67
Paging: faults 0
Network I/O: packets received 0, sent 0
songcat@songcat-virtual-machine:~/nachos/NachOS-4.0/code/test$
    
```



2. Chương trình cat:

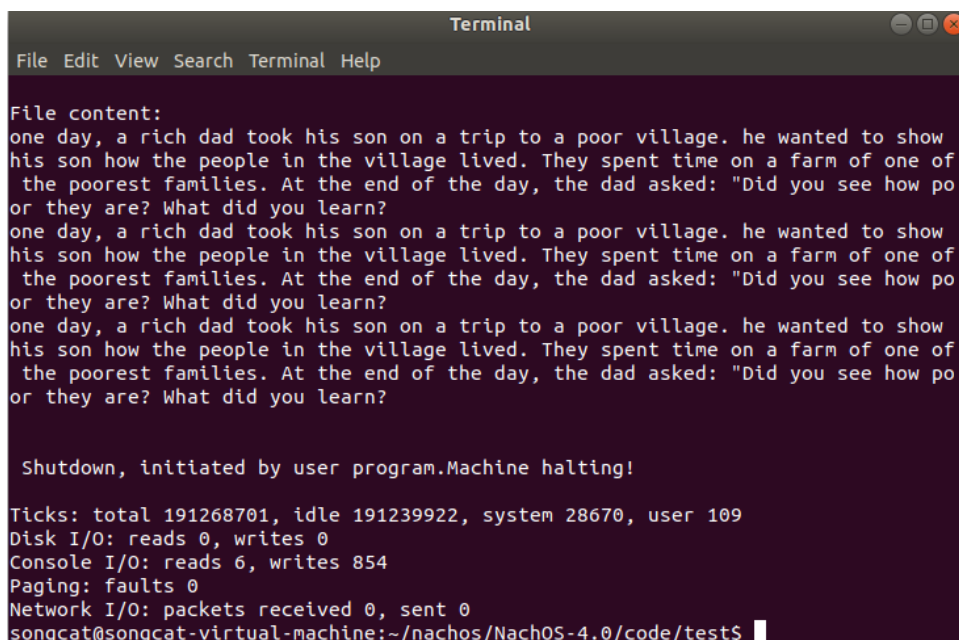
- Yêu cầu: Viết chương trình **cat**, yêu cầu nhập tên file muốn đọc, hiển thị nội dung file.
- Cách thực hiện:
 - Bước 1: Người dùng nhập tên file muốn đọc.
 - Bước 2: Mở file và lấy độ dài của file bằng cách tìm vị trí cuối file. Nếu mở file không thành công thì thông báo lỗi, ngược lại chuyển sang bước 3.
 - Bước 3: Đọc nội dung file với độ dài vừa lấy được và in ra màn hình.
- Hình ảnh demo chương trình: Đọc file 'a.txt' có trong thư mục test
 - Ban đầu, file 'a.txt' có nội dung như hình bên dưới



- Chạy chương trình **cat** và nhập tên file là 'a.txt'

```
songcat@songcat-virtual-machine:~/nachos/NachOS-4.0/code/test$ ../build.linux/nachos -x cat
Input file to cat: a.txt
```

- Chương trình kết thúc và nội dung file được hiển thị lên màn hình console



```

Terminal
File Edit View Search Terminal Help

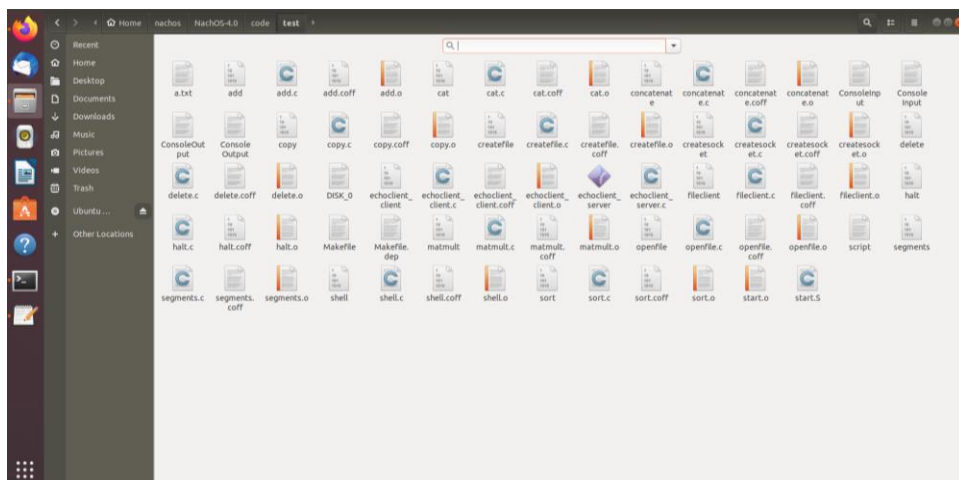
File content:
one day, a rich dad took his son on a trip to a poor village. he wanted to show
his son how the people in the village lived. They spent time on a farm of one of
the poorest families. At the end of the day, the dad asked: "Did you see how po
or they are? What did you learn?
one day, a rich dad took his son on a trip to a poor village. he wanted to show
his son how the people in the village lived. They spent time on a farm of one of
the poorest families. At the end of the day, the dad asked: "Did you see how po
or they are? What did you learn?
one day, a rich dad took his son on a trip to a poor village. he wanted to show
his son how the people in the village lived. They spent time on a farm of one of
the poorest families. At the end of the day, the dad asked: "Did you see how po
or they are? What did you learn?

Shutdown, initiated by user program.Machine halting!

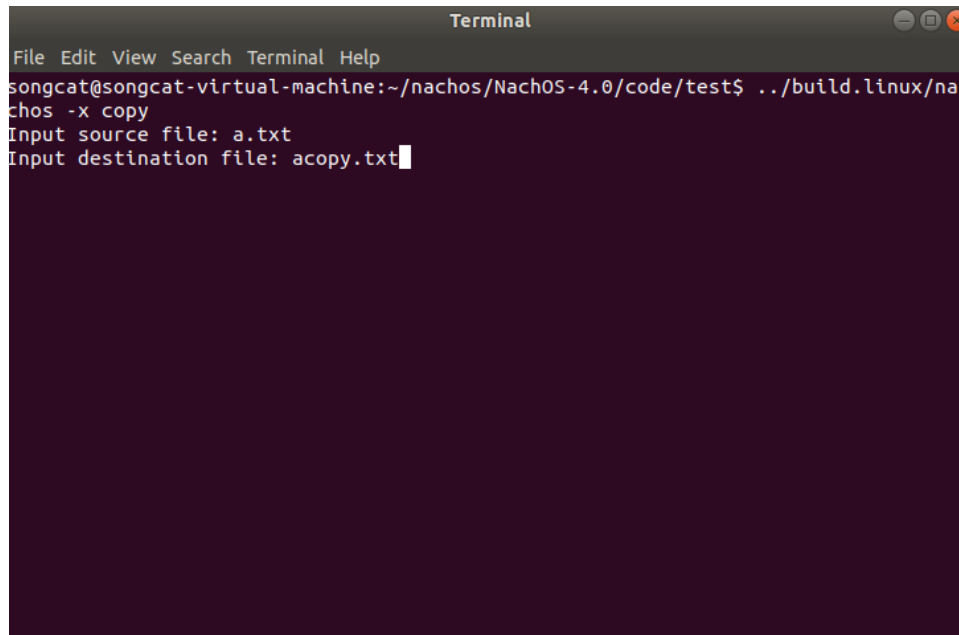
Ticks: total 191268701, idle 191239922, system 28670, user 109
Disk I/O: reads 0, writes 0
Console I/O: reads 6, writes 854
Paging: faults 0
Network I/O: packets received 0, sent 0
songcat@songcat-virtual-machine:~/nachos/NachOS-4.0/code/test$
    
```

3. Chương trình copy:

- Yêu cầu: Viết chương trình **copy**, yêu cầu nhập tên file nguồn và file đích và thực hiện copy.
- Cách thực hiện:
 - Bước 1: Người dùng nhập tên file nguồn (src) và file đích (dst).
 - Bước 2: Mở src, nếu không mở được thì thông báo lỗi và kết thúc.
 - Bước 3: Thực hiện đọc file src giống như chương trình cat.
 - Bước 4: Mở file dst, nếu không mở được thì tạo một file dst mới, nếu không tạo được thì dùng chương trình.
 - Bước 5: Ghi dữ liệu đọc được từ src sang dst và kết thúc chương trình.
- Hình ảnh demo chương trình: Tạo ra file ‘acopy.txt’ từ nội dung của file ‘a.txt’
 - Ban đầu chưa có file ‘acopy.txt’



- Chạy chương trình **copy** và nhập tên file cần copy là 'a.txt' và file mới được tạo ra để chứa nội dung đã sao chép được là 'acopy.txt'.

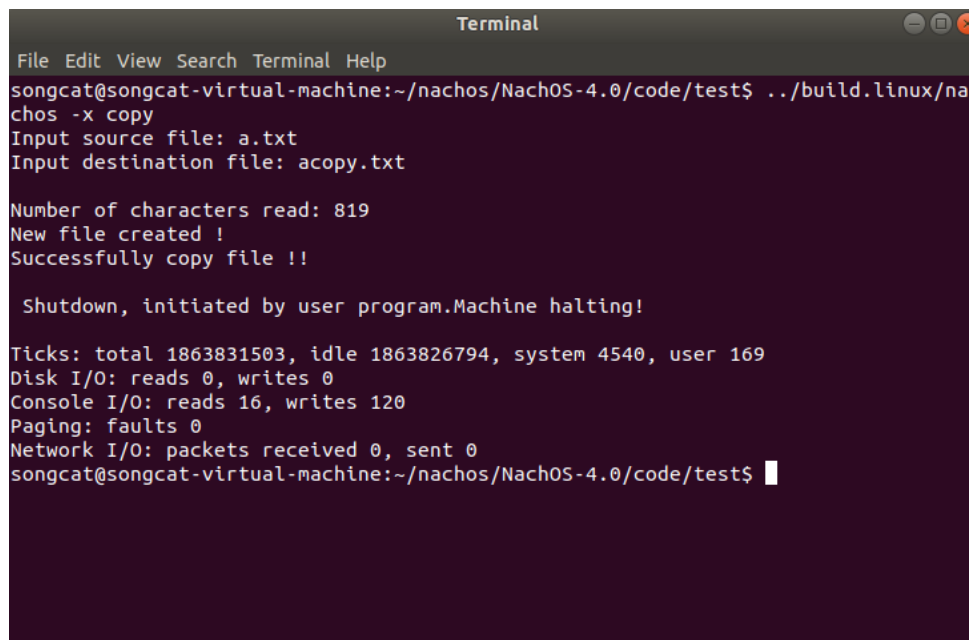


```

Terminal
File Edit View Search Terminal Help
songcat@songcat-virtual-machine:~/nachos/NachOS-4.0/code/test$ ../build.linux/nachos -x copy
Input source file: a.txt
Input destination file: acopy.txt
    
```

- Chương trình kết thúc và đã tạo được file cần copy là 'acopy.txt' với nội dung giống với file 'a.txt'

Nội dung hiển thị trên màn hình console:



```

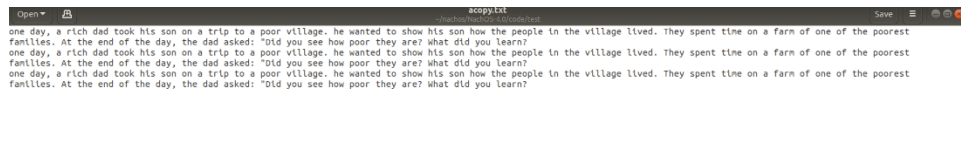
Terminal
File Edit View Search Terminal Help
songcat@songcat-virtual-machine:~/nachos/NachOS-4.0/code/test$ ../build.linux/nachos -x copy
Input source file: a.txt
Input destination file: acopy.txt

Number of characters read: 819
New file created !
Successfully copy file !!

Shutdown, initiated by user program.Machine halting!

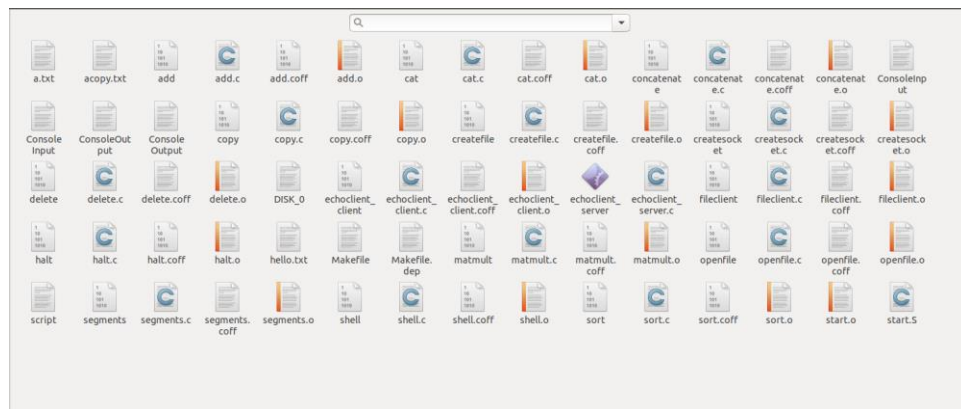
Ticks: total 1863831503, idle 1863826794, system 4540, user 169
Disk I/O: reads 0, writes 0
Console I/O: reads 16, writes 120
Paging: faults 0
Network I/O: packets received 0, sent 0
songcat@songcat-virtual-machine:~/nachos/NachOS-4.0/code/test$
    
```

Nội dung file 'acopy.txt' vừa được tạo mới:

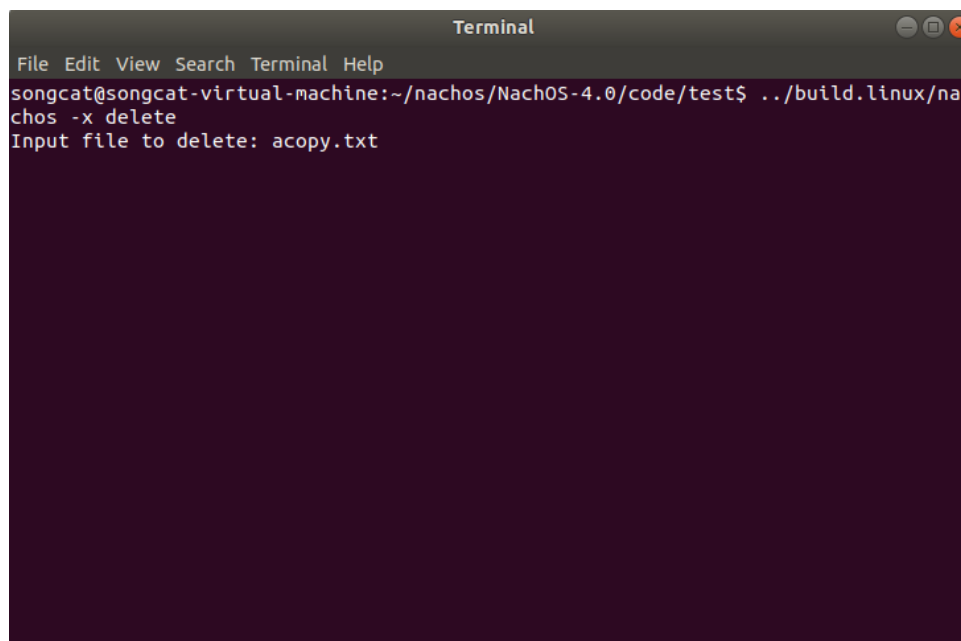


4. Chương trình delete:

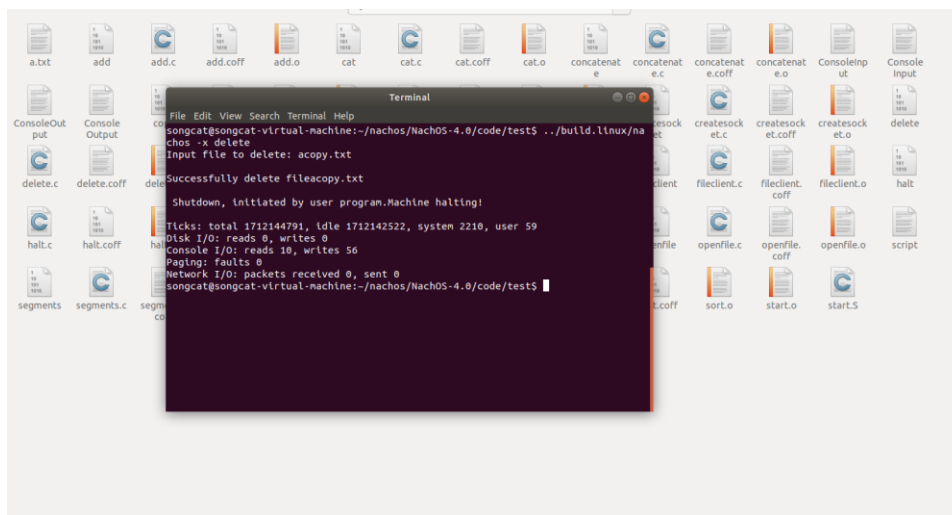
- Yêu cầu: Viết chương trình **delete** để kiểm tra SystemCall Remove, yêu cầu nhập tên file muốn xóa.
- Cách thực hiện:
 - Bước 1: Người dùng nhập tên file muốn xóa.
 - Bước 2: Nếu file không tồn tại thì thông báo lỗi, ngược lại chuyển sang bước 3.
 - Bước 2: Xóa file và kết thúc chương trình.
- Hình ảnh demo chương trình: Xóa file ‘acopy.txt’
 - Ban đầu, thư mục test có chứa file ‘acopy.txt’.



- Chạy chương trình delete và nhập tên file cần xóa là ‘acopy.txt’.



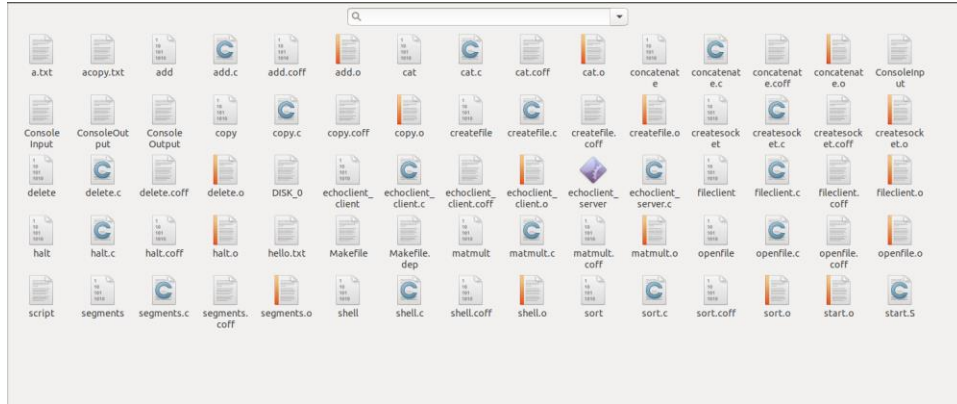
- Chương trình kết thúc và file ‘acopy.txt’ đã được xóa.



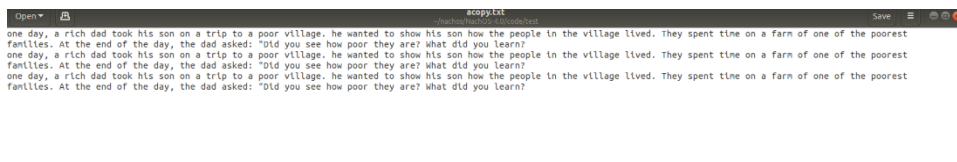
5. Chương trình concatenate:

- Yêu cầu: Viết chương trình **concatenate** để nối nội dung của 2 file, yêu cầu nhập tên file nguồn 1 và file nguồn 2.
- Cách thực hiện:
 - Bước 1: Người dùng nhập tên file nguồn 1 và 2 (src1 và src2).
 - Bước 2: Mở src1, nếu mở không được thì báo lỗi và kết thúc chương trình, nếu được thì đọc nội dung src1.
 - Bước 3: Mở src2, nếu mở không được thì báo lỗi và kết thúc chương trình, nếu được thì đọc nội dung src2.
 - Bước 4: Tạo file lưu nội dung sau khi nối 2 file, nếu thất bại thì báo lỗi và kết thúc chương trình, ngược lại thì viết nội dung src1 và src2 vào file và kết thúc chương trình.
- Hình ảnh demo chương trình: Nối nội dung hai file ‘a.txt’ và file ‘b.txt’ và ghi vào file ‘resConcatenate.txt’
 - Ban đầu, thư mục test không có file ‘resConcatenate.txt’ và file ‘a.txt’, ‘b.txt’ có nội dung như sau:

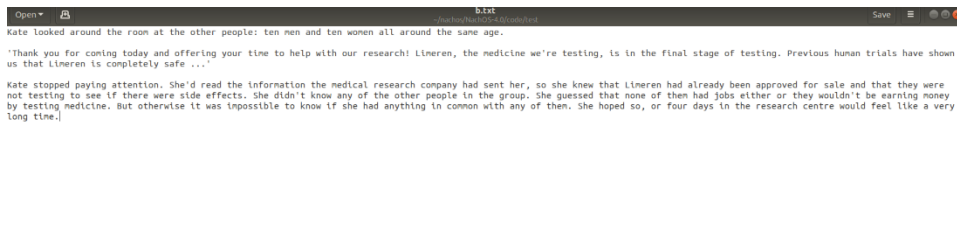
Thư mục test không có file ‘resConcatenate.txt’



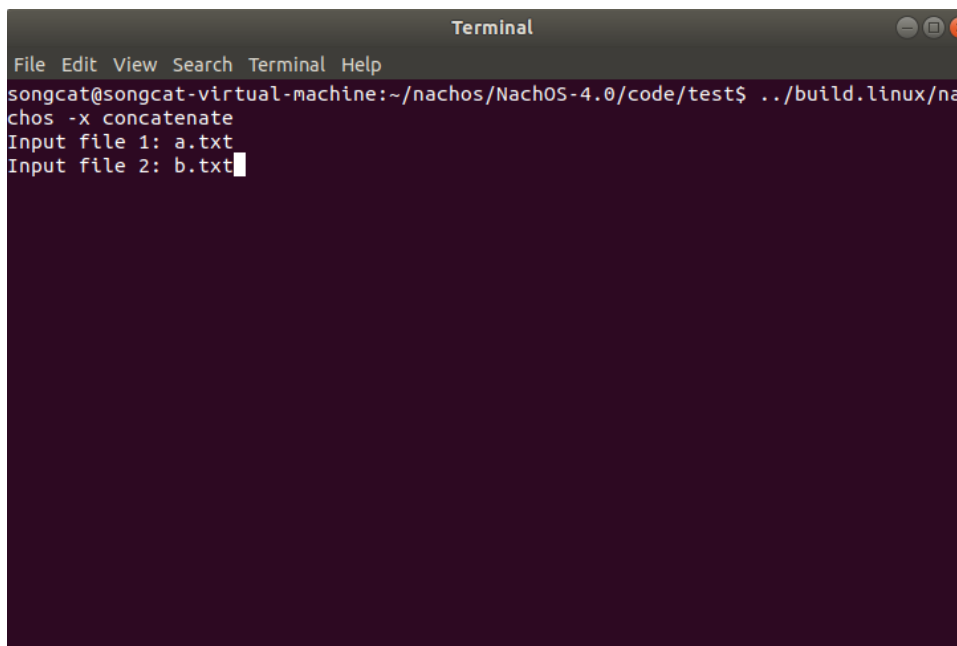
Nội dung file ‘a.txt’



Nội dung file ‘b.txt’



- Chạy chương trình và nhập tên hai file cần nối là ‘a.txt’ và ‘b.txt’.



- Chương trình kết thúc và file ‘resConcatenate.txt’ đã được tạo ra với nội dung như bên dưới

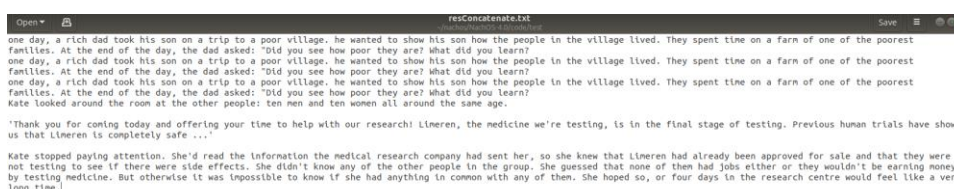
Màn hình console sau khi chương trình thực hiện xong thao tác:

```
Concatenate file successfully !

Shutdown, initiated by user program.Machine halting!

Ticks: total 1591925251, idle 1591832374, system 66770, user 26107
Disk I/O: reads 0, writes 0
Console I/O: reads 12, writes 1904
Paging: faults 0
Network I/O: packets received 0, sent 0
songcat@songcat-virtual-machine:~/nachos/NachOS-4.0/code/test$
```

Nội dung file ‘resConcatenate.txt’:



```
one day, a rich dad took his son on a trip to a poor village. he wanted to show his son how the people in the village lived. They spent time on a farm of one of the poorest families. At the end of the day, the dad asked: 'Did you see how poor they are? What did you learn?'
one day, a rich dad took his son on a trip to a poor village. he wanted to show his son how the people in the village lived. They spent time on a farm of one of the poorest families. At the end of the day, the dad asked: 'Did you see how poor they are? What did you learn?'
one day, a rich dad took his son on a trip to a poor village. he wanted to show his son how the people in the village lived. They spent time on a farm of one of the poorest families. At the end of the day, the dad asked: 'Did you see how poor they are? What did you learn?'
Kate looked around the room at the other people: ten men and ten women all around the same age.

'Thank you for coming today and offering your time to help with our research! Lineren, the medicine we're testing, is in the final stage of testing. Previous human trials have shown us that Lineren is completely safe ...'

Kate stopped paying attention. She'd read the information the medical research company had sent her, so she knew that Lineren had already been approved for sale and that they were not testing to see if there were side effects. She didn't know any of the other people in the group. She guessed that none of them had jobs either or they wouldn't be earning money by testing medicine. But otherwise it was impossible to know if she had anything in common with any of them. She hoped so, or four days in the research centre would feel like a very long time.
```

6. Chương trình echoclient_client:

- Yêu cầu: Viết chương trình trao đổi thông tin giữa client và server thông qua socket. Chương trình tạo ra 4 sockets để gửi tin nhắn cho client và sau đó nhận lại tin nhắn với nội dung đã được viết in hoa rồi xuất ra màn hình console.
- Cách thực hiện:
 - Bước 1: Chạy chương trình server và client
 - Bước 2: Chương trình client tạo ra 4 socket để kết nối với server. Nếu kết nối không thành công thì màn hình hiển thị không thể kết nối.
 - Bước 3: Cho từng client gửi tin nhắn đến server và nhận về chuỗi tin nhắn được in hoa.
 - Bước 4: Lần lượt đóng socket của các client, nếu đóng không thành công thì báo lỗi, ngược lại, chương trình hiển thị đóng thành công.
- Hình ảnh demo chương trình:
 - Chạy chương trình server

```
songcat@songcat-virtual-machine:~/nuchos/NachOS-4.0/code/test$ gcc echoclient_server.c -o echoclient_server
songcat@songcat-virtual-machine:~/nuchos/NachOS-4.0/code/test$ ./echoclient_server
waiting for clients
```

- Chạy chương trình **echoclient**

```
Terminal
File Edit View Search Terminal Help
songcat@songcat-virtual-machine:~$ cd nachos/NachOS-4.0/code/test
songcat@songcat-virtual-machine:~/nuchos/NachOS-4.0/code/test$ make
make: Nothing to be done for 'all'.
songcat@songcat-virtual-machine:~/nuchos/NachOS-4.0/code/test$ ../build.linux/nuchos -x echoclient_client
-----Client 1-----
Input message to send:
```

- Gửi chuỗi cho từ các clients và server trả về chuỗi đã được in hoa

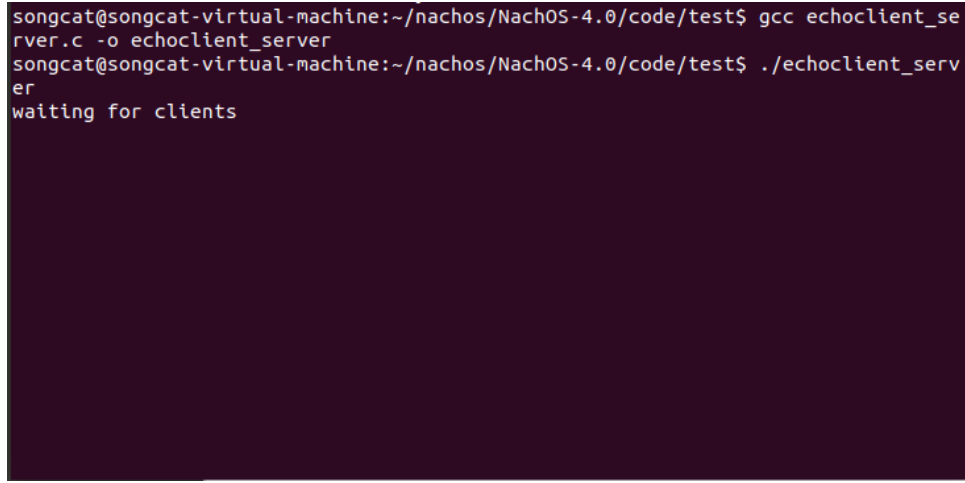
```
Terminal
File Edit View Search Terminal Help
-----Client 1-----
Input message to send:
hello world
Received from server:
HELLO WORLD
-----Client 2-----
Input message to send:
abcxyz
Received from server:
ABCXYZ
-----Client 3-----
Input message to send:
djsjd
Received from server:
DJSJD
-----Client 4-----
Input message to send:
djetlowie
Received from server:
DJETLOWIE

Terminal
File Edit View Search Terminal Help
songcat@songcat-virtual-machine:~$ cd nachos/NachOS-4.0/code/test
songcat@songcat-virtual-machine:~/nuchos/NachOS-4.0/code/test$ gcc echoclient_server.c -o echoclient_server
songcat@songcat-virtual-machine:~/nuchos/NachOS-4.0/code/test$ ./echoclient_server
waiting for clients
Accepted new connection from a client 127.0.0.1:44656
received hello world from client
waiting for clients
Accepted new connection from a client 127.0.0.1:44662
received abcxyz from client
waiting for clients
Accepted new connection from a client 127.0.0.1:44664
received djsjd from client
waiting for clients
Accepted new connection from a client 127.0.0.1:44672
received djetlowie from client
waiting for clients
```

7. Chương trình fileclient:

- Yêu cầu: Viết chương trình trao đổi thông tin giữa client và server thông qua socket. Chương trình tạo ra một client đọc thông tin của một file .txt rồi gửi cho server thông qua socket. Sau đó, server gửi lại thông tin file đó cho client dưới dạng in hoa để client viết vào file đích.
- Cách thực hiện:
 - Bước 1: Chạy chương trình server và client.
 - Bước 2: Chương trình client tạo ra 1 client kết nối với server thông qua socket.
 - Bước 3: Nhập tên file nguồn cần đọc và file đích để ghi thông tin nhận được.
 - Bước 4: Cho client đọc thông tin file nguồn rồi gửi đến server và nhận về chuỗi tin nhắn được in hoa.

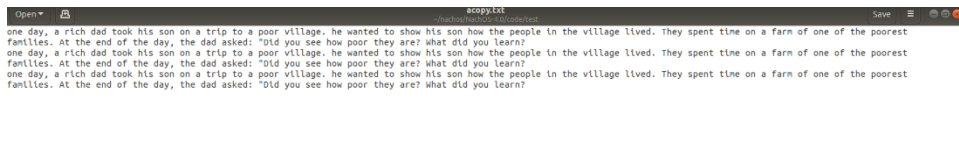
- Bước 5: Viết thông tin nhận được vào file đích. Nếu file đích đã tồn tại thì viết vào, ngược lại tạo file rỗng để ghi thông tin nhận được.
 - Bước 6: Đóng socket.
- Hình ảnh demo chương trình: Đọc file ‘a.txt’ và gửi lên server rồi viết thông tin nhận được vào file ‘b.txt’
- Chạy chương trình server



```

songcat@songcat-virtual-machine:~/nachos/NachOS-4.0/code/test$ gcc echoclient_server.c -o echoclient_server
songcat@songcat-virtual-machine:~/nachos/NachOS-4.0/code/test$ ./echoclient_server
waiting for clients
    
```

- Nội dung file ‘a.txt’

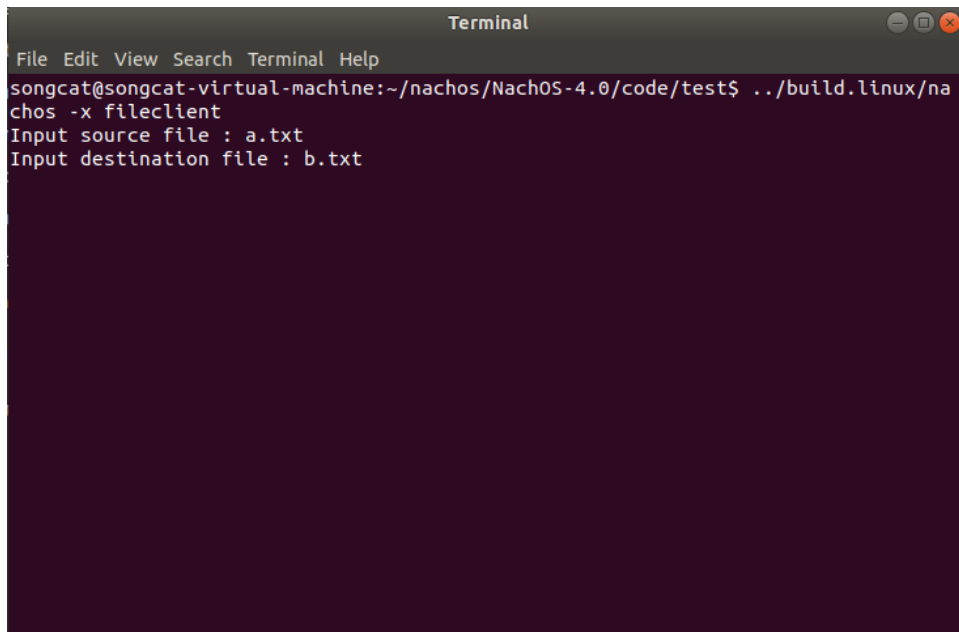


one day, a rich dad took his son on a trip to a poor village. he wanted to show his son how the people in the village lived. They spent time on a farm of one of the poorest families. At the end of the day, the dad asked: "Did you see how poor they are? what did you learn?"

one day, a rich dad took his son on a trip to a poor village. he wanted to show his son how the people in the village lived. They spent time on a farm of one of the poorest families. At the end of the day, the dad asked: "Did you see how poor they are? what did you learn?"

one day, a rich dad took his son on a trip to a poor village. he wanted to show his son how the people in the village lived. They spent time on a farm of one of the poorest families. At the end of the day, the dad asked: "Did you see how poor they are? what did you learn?"

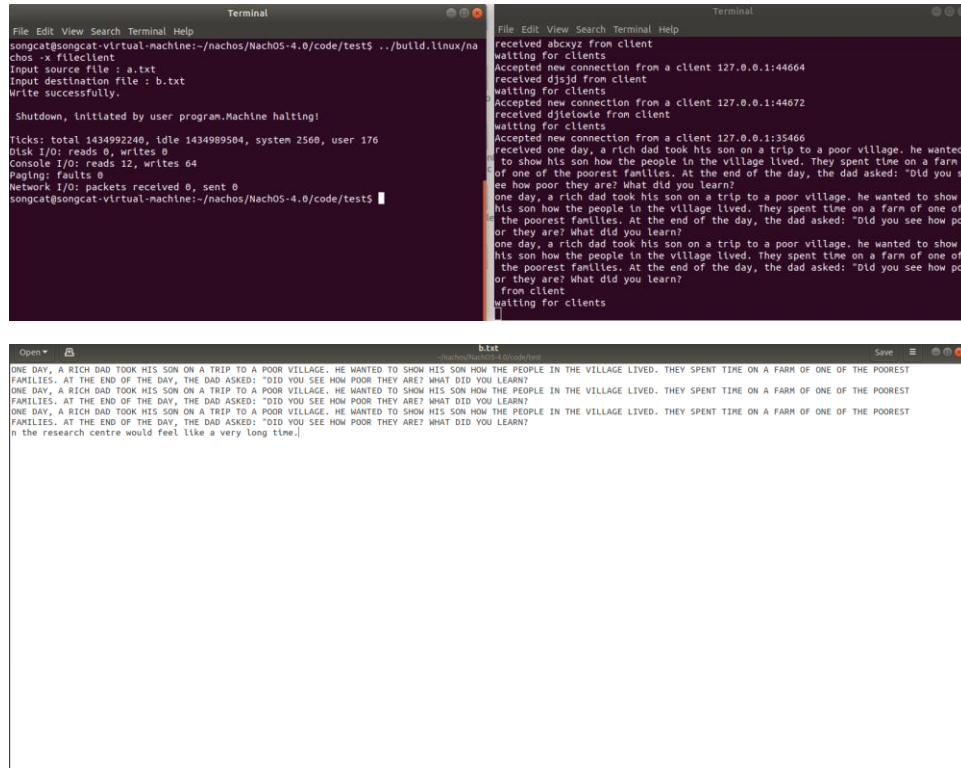
- Chạy chương trình fileclient và nhập tên file nguồn và file đích



```

Terminal
File Edit View Search Terminal Help
songcat@songcat-virtual-machine:~/nachos/NachOS-4.0/code/test$ ../build.linux/nachos -x fileclient
Input source file : a.txt
Input destination file : b.txt
    
```

- Chương trình kết thúc thông tin nằm trong file ‘a.txt’ đã được in hoa và viết vào file ‘b.txt’.



VI. TÀI LIỆU THAM KHẢO

- [1] <https://github.com/haile01/nachos>
- [2] <https://mohsensy.github.io/programming/2019/09/25/echo-server-and-client-using-sockets-in-c.html>
- [3] <https://pubs.opengroup.org/onlinepubs/7908799/xns/syssocket.h.html>
- [4] <https://github.com/nguyenthanhchungfit/Nachos-Programing-HCMUS>
- [5] https://youtube.com/playlist?list=PLRgTVtca98hUgCN2_2vzsAAXPiTFbvHpO