TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN
TÌM HIỂU SYSTEM CALLS, CÁC THAO TÁC VỚI FILE VÀ NETWORK

GV hướng dẫn: thầy Nguyễn Thanh Quân
Nhóm thực hiện:
21120575 – Nguyễn Thành Trí
21120574 – Trần Đình Nhật Trí
21120590 – Nguyễn Thủy Uyên

Thành phố Hồ Chí Minh, ngày 23 tháng 10 năm 2023

MỤC LỤC

I. THÔNG TIN ĐỒ ÁN

A. Bảng phân công công việc

MSSV	Họ tên	Nhiệm vụ
21120575	Nguyễn Thành Trí	Syscall socketTCP, Connect, Send, Receive, Close Chương trình test echo, truyền file
2112076	Trần Đình Nhật Trí	Syscall Create, syscall Open, syscall Close Chương trình test create, cat, concatenate, delete
21120590	Nguyễn Thủy Uyên	Syscall Read, syscall Write, syscall Seek, syscall Remove Report

II. CẤU TRÚC CHUNG

A. Cấu trúc file exception.cc

- Hàm increaseProgramCounter() tăng PC sau mỗi lần thực hiện lệnh theo MIPS.
- Hàm User2System và System2User chuyển đổi giữa user space và kernel space bằng cách gọi ReadMem và WriteMem đã có trong Machine class.
- Hàm readCharacters() đọc char từ console và lưu vào mảng characterBuffer, hàm kết thúc ghi gặp 1 trong các empty space: \n, \r, \t, ‘ ‘, \x0b, EOF.
- Hàm ReadNumFromConsole() đọc số nguyên từ console sử dụng hàm readCharacters(), hàm kết thúc khi gặp điều kiện: đọc được chữ cái hoặc tràn số nguyên.
- Hàm PrintNumToConsole() in số nguyên ra console, sử dụng PutChar từ class SynchConsoleOut để in từng ký tự.
- Hàm ReadCharFromConsole() đọc char từ console bằng GetChar từ class SynchoConsoleIn.
- Hàm PrintCharToConsole() ghi char lên console bằng PutChar từ class SynchoConsoleOut.
- Hàm ReadStringFromConsole() đọc một chuỗi từ console bằng GetChar từ class SynchoConsoleIn nếu kí tự ‘\n’ thì thay bằng ‘\0’. Hàm dừng lại khi gặp \0 hoặc đã đọc đủ độ dài truyền vào.
- Hàm PrintStringToConsole() ghi một chuỗi lên console bằng PutChar từ class SynchoConsoleOut. Hàm dừng lại khi gặp \0 hoặc đã ghi đủ độ dài truyền vào.
- ExceptionHandler đọc giá trị từ thanh ghi 2 gọi các system call tương ứng.

B. Cấu trúc File Descriptor Table

- Các loại type mở file (phân biệt file thông thường và socket)
  - 0: read and write
  - 1: read only
  - 2: socket
- OpenFileId là một số nguyên lưu ID file.

- Struct Socket quản lí các socket client
  - ID của socket
  - Hàm close socket
- Struct NormalFile quản lí các file đọc ghi thông thường
  - Tên file
  - Con trỏ OpenFile
  - Vị trí hiện tại của con trỏ file
- Class FileDescriptor là riêng biệt giữa các file
  - Gồm các đối tượng Socket, normalFile, loại file và OpenFileId đã được định nghĩa ở trên. Khi thao tác trên một fil hoặc socket chúng sẽ có 1 ID, để phân biệt ID nào của file và ID nào của Socket ta sẽ có công thức:
    ID = type*100 + ID thực sự được tạo bởi Nachos.
    Trả về 1 số có 3 chữ số với hàng trăm là type, 2 chữ số còn lại là ID được tạo bởi Nachos. VD: một file mở với chế độ read only và FD được Nachos tạo là 5 thì ID sẽ là 105.
  - Bao gồm các hàm openFile, openSocket, connectSocket, readFile, writeFile, seekFile, closeFile.
- Class Table lưu 20 file descriptors
  - Gồm khai báo maxsize = 20 theo yêu cầu đề bài.
  - Biến lưu size hiện tại thuận tiện cho việc kiểm tra thêm file, xóa file,...
  - Con trỏ FileDescriptor
  - Các phương thức được cài đặt: isOpened (kiểm tra file có mở hay chưa, với ID hoặc với filename và type), closeFile (với ID), open (với filename và type), getFile trả về con trỏ FileDescriptor để truy cập vào các phần tử trong bảng và thực hiện các hàm.

III. CÀI ĐẶT CÁC SYSTEM CALL

A. System call thao tác với File

1. System call Create

- Mục đích: int Create(char *name) tạo 1 file rỗng, trả về 0 nếu thành công hoặc -1 nếu có lỗi.
- Cách hoạt động:
  - Một khi SC_Create thực thi, đầu tiên sẽ đọc vào giá trị thanh ghi 4 vào một địa chỉ ảo, và tiến hành chuyển vùng nhớ từ User space sang Kernel space để lấy filename.
  - Kiểm tra file có Null hay không, nếu có trả về thanh ghi 2 giá trị -1.
  - Kiểm tra có tạo được file hay không (sử dụng Nachos FileSystem Object, hàm Create()), nếu không trả về thanh ghi 2 giá trị -1.
  - Nếu các trường hợp trên không lỗi, file tạo thành công trả về thanh ghi 2 giá trị 0 và tăng PC.

2. System call Open

- Mục đích: OpenFileID Open(char *name, int type) mở file read only hoặc read and write. Hàm trả về file descriptor id hoặc -1 nếu lỗi.
- Cách hoạt động:
  - Đọc giá trị thanh ghi 4 vào một địa chỉ ảo, và tiến hành chuyển vùng nhớ từ User space sang Kernel space để lấy filename. Đọc giá trị thanh ghi 5 vào type để xác định mở file hay socket.
  - Nếu type là 0 (đọc) hoặc 1(đọc ghi) thì thực thi, nếu không trả về -1 vào thanh ghi 2.

- Nếu đúng, ta tiếp tục kiểm tra trong Table có còn trống không (currentSize có bằng 20), nếu bảng đã đầy không thể mở file thì trả về -1.
- Nếu Table vẫn trống, duyệt qua 20 vị trí và tìm được vị trí trống để mở file (gọi Open của fileSystem Object) và tiến hành định danh ID (bằng công thức trong phần file_descriptors.h).
- Ghi ID vào thanh ghi 2 và tăng PC.

3. System call Close

- Mục đích: int Close (OpenFileID id) đóng file, trả về -1 nếu lỗi hoặc 0 nếu thành công.
- Cách hoạt động:
  - Đọc giá trị thanh ghi 4 vào ID.
  - Kiểm tra file đã mở hay chưa bằng cách duyệt qua 20 vị trí trong Table. Nếu chưa mở trả về -1 và tăng PC. Nếu đã mở thì giải phóng vùng nhớ của OpenFile*, gán vị trí trong danh sách file là NULL và set các giá trị về default.
  - Ghi 0 vào thanh ghi 2 và tăng PC.

4. System call Read

- Mục đích: int Read(char *buffer, int size, OpenFile id) đọc file và trả đúng số ký tự đọc được hoặc -1 nếu lỗi.
- Cách hoạt động:
  - Đọc vào thanh ghi 4 một địa chỉ ảo, đọc vào thanh ghi 5 độ dài byte cần đọc và đọc vào thanh ghi 6 OpenFileID của file cần đọc.
  - Chuyển vùng nhớ từ User space sang Kernel space với số byte và địa chỉ ảo.
  - Kiểm tra ID là số âm, trả về -1 vào thanh ghi 2.
  - Kiểm tra ID = 0 hay đọc từ console input thì ta đọc với lớp synchConsoleIn, hệ thống sẽ chờ để nhận được ký tự thật sự và trả về số byte đọc được vào thanh ghi 2, còn nếu không trả về -1 vào thanh ghi 2.
  - Kiểm tra ID không nằm trong Table trả về -1, nếu có tiến hành đọc file sử dụng hàm Read trong OpenFile để đọc số byte thực sự đọc được từ file. Kiểm tra số byte đọc được < 0 thì trả về -1, nếu không trả về số byte thật sự đọc được vào thanh ghi 2. Sau đó truyền buffer đọc được từ Kernel space về User space và giải phóng vùng nhớ buff.
  - Tăng PC.

5. System call Write

- Mục đích: int Write(char *buffer, int size, OpenFile id) đọc file và trả đúng số ký tự ghi được hoặc -1 nếu lỗi.
- Cách hoạt động:
  - Đọc vào thanh ghi 4 một địa chỉ ảo, đọc vào thanh ghi 5 độ dài byte cần ghi và đọc vào thanh ghi 6 OpenFileID của file.
  - Chuyển vùng nhớ từ User space sang Kernel space với số byte và địa chỉ ảo vào buffer.
  - Kiểm tra ID là số âm, trả về -1 vào thanh ghi 2.
  - Kiểm tra ID = 1 hay ghi vào console output thì ta ghi với lớp synchConsoleOut, và trả về số byte ghi được vào thanh ghi 2 và tăng PC.
  - Kiểm tra ID không nằm trong Table trả về -1 và tăng PC.
  - Nếu có tiến hành kiểm tra type bằng cách xem chữ số đầu của ID nếu là read only trả về-1
  - Nếu không kiểm tra file có tồn tại trong Table hay không và trả về -1 nếu không.

- Ghi file sử dụng hàm trong OpenFile để ghi số byte thực sự. Kiểm tra số byte đọc được < 0 thì trả về -1, nếu không trả về số byte thật sự ghi được vào thanh ghi 2. Sau đó truyền buffer từ Kernel space về User space và giải phóng vùng nhớ và tăng PC.
- Các trường hợp còn lại nếu không thỏa điều kiện, trả về -1 vào thanh ghi 2 và tăng PC.

6. System call Seek

- Mục đích: int Seek(int position, OpenFileID id) chuyển con trỏ tới vị trí pos (-1 là cuối file), trả về vị trí thực sự trong file hoặc -1 nếu lỗi.
- Cách hoạt động:
  - Đọc giá trị thanh ghi 4 vào pos là vị trí cần dời, đọc giá trị thanh ghi 5 vào ID.
  - Nếu ID = 0 hoặc 1 (console input hoặc console output) thì trả về giá trị -1 vào thanh ghi 2 và tăng PC.
  - Kiểm tra ID có nằm trong Table hay chưa để xác định đã mở file, nếu chưa trả về -1 vào thanh ghi 2 và tăng PC.
  - Nếu đã mở file, kiểm tra vị trí pos = -1 thì trả về độ dài file bằng Length() trong OpenFile class, vị trí pos vượt ngoài độ dài file hoặc pos là số âm hoặc thấy file được mở dưới mode socket thì trả về -1 vào thanh ghi 2. Vị trí pos hợp lệ thì gọi hàm Seek trong class OpenFile và trả về vị trí hiện tại đã được dời đến.
  - Tăng PC.

7. System call Remove

- Mục đích: int Remove(char *name) xóa file thành công trả về 0 hoặc -1 nếu thất bại.
- Cách hoạt động:
  - Đầu tiên sẽ đọc vào giá trị thanh ghi 4 vào một địa chỉ ảo, và tiến hành chuyển vùng nhớ từ User space sang Kernel space để lấy filename.
  - Kiểm tra file có nằm trong Table hay không bằng cách gọi hàm isOpened đã định nghĩa trong lớp Table (file_descriptors.h) với type 0 và 1 (read only và read and write), ở đây ta duyệt qua 20 vị trí trong table và so sánh filename được truyền với filename của từng file descriptor để lấy ra OpenFileId, nếu không có file nào có filename giống thì trả về -1 vào thanh ghi 2, nghĩa là file không nằm trong Table.
  - Kiểm tra file có đang được mở không với hàm isOpened truyền vào ID ở trên, nếu không mở trả về -1.
  - Sau khi mở được, gọi closeFile() để đóng. closeFile sẽ nhận vào ID đã được lấy ở trên và kiểm tra đây là ID của normal file hay socket để tiến hành đóng file. Đóng file bằng hàm Close() của class OpenFile.
  - Sau khi giải phóng vùng nhớ, set các giá trị tại vị trí file đã đóng về default, tăng PC.

B. System call thao tác với Network TCP

1. System call SocketTCP

- Mục đích: int SocketTCP() trả về file descriptor id hoặc -1 nếu lỗi.
- Cách hoạt động:
  - Tạo 1 biến fileid để lưu trữ OpenFileId của file socket sắp tạo.
  - Sử dụng phương thức open trong class table (đã được cài đặt trong file_descriptors.h) với 2 tham số đầu vào của phương thức open là char* name = NULL (vì file socket tạo ra không cần tên, vì ta sẽ tham chiếu đến file đấy bằng IP và port) và int type = 2 (tương ứng với tạo 1 file socket).

- Ở phương thức open, ta sẽ kiểm tra mảng file description có đầy chưa, nếu đầy thì trả về -1, nghĩa là fileid được khai báo ban đầu sẽ có giá trị -1.
- Nếu mảng file description chưa đầy, ta tiến hành tạo socket và trả về giá trị id trong mảng file description, nghĩa là fileid được khai báo ở trên sẽ có giá trị là id của file socket vừa tạo trong mảng file description.
- Cuối cùng, ghi giá trị của fileid trong thanh ghi số 2 và tăng PC.

2. System call Connect

- Mục đích: int Connect(int socketid, char *ip, int port) connect đến server theo thông tin ip và port, trả về 0 nếu thành công hoặc -1 nếu lỗi.
- Cách hoạt động:
  - Nhận tham số từ thanh ghi:
    - Đọc giá trị của thanh ghi số 4 để lấy socketid - đây là file descriptor của socket cần kết nối.
    - Đọc giá trị của thanh ghi số 5 để lấy địa chỉ IP của máy chủ từ xa được truyền từ không gian người dùng.
    - Đọc giá trị của thanh ghi số 6 để lấy cổng của máy chủ từ xa.
  - Chuyển đổi địa chỉ IP từ không gian người dùng sang không gian kernel:
    - Sử dụng hàm copyStringFromMachine để sao chép địa chỉ IP từ không gian người dùng sang không gian kernel.
  - Kết nối socket đến máy chủ từ xa:
    - Tạo một struct sockaddr_in (remote_addr) để đại diện cho máy chủ từ xa với các thông tin như địa chỉ IP và cổng.
    - Gọi hàm connect với socket ID và địa chỉ của máy chủ để thực hiện việc kết nối.
  - Kiểm tra kết nối và xử lý kết quả:
    - Kiểm tra giá trị trả về từ hàm connect. Nếu kết nối thành công, trả về file descriptor của socket, ngược lại thông báo lỗi và đóng socket.
    - Ghi giá trị của socketid vào thanh ghi số 2 để trả về kết quả cho chương trình người dùng.
  - Tăng Program Counter và kết thúc System call:
    - Tăng giá trị của Program Counter để chuyển đến lệnh tiếp theo.
    - Kết thúc system call.

3. System call Send

- Mục đích: int Send(int socketid, char *buffer, int len) gửi dữ liệu từ socket, trả về số lượng bytes gửi đi, trả về 0 nếu kết nối đóng hoặc -1 nếu thất bại.
- Cách hoạt động:
  - Nhận tham số từ thanh ghi:
    - Đọc giá trị của thanh ghi số 4 để lấy addrSocketid - đây là địa chỉ của socketid trong không gian người dùng.
    - Đọc giá trị của thanh ghi số 5 để lấy idbuffer - đây là địa chỉ của bộ đệm trong không gian người dùng chứa dữ liệu cần gửi.
    - Đọc giá trị của thanh ghi số 6 để lấy len - đây là độ dài của dữ liệu cần gửi.
  - Lấy thông tin socket và chuẩn bị bộ đệm:
    - Lấy socketid từ địa chỉ socketid trong không gian người dùng thông qua table.getFile(addrSocketid)->getSocketID().
    - Cấp phát bộ đệm buffer với độ dài là len.

    - Sao chép dữ liệu từ không gian người dùng tới không gian kernel thông qua hàm User2System và lưu vào buffer.
  - Gửi dữ liệu thông qua hàm SocketSend:
    - Gọi hàm SocketSend với tham số là buffer, len, và socketid.
    - Sử dụng setsockopt để đặt timeout cho việc gửi dữ liệu.
    - Sử dụng hàm send để gửi dữ liệu thông qua socket.
    - Kiểm tra giá trị trả về. Nếu có lỗi, in thông báo lỗi và trả về -1.
    - Nếu gửi thành công, trả về độ dài thực sự của dữ liệu đã gửi.
    - Nếu có lỗi, trả về -1.
  - Kết thúc System call:
    - Tăng Program Counter và kết thúc.

4. System call Receive

- Mục đích: int Receive(int socketid, char *buffer, int len) nhận dữ liệu từ socket, trả về số lượng bytes nhận được, trả về 0 nếu kết nối đóng hoặc -1 nếu thất bại.
- Cách hoạt động:
  - Nhận tham số từ thanh ghi:
    - Đọc giá trị của thanh ghi số 4 để lấy addrSocketid - đây là địa chỉ của socketid trong không gian người dùng.
    - Đọc giá trị của thanh ghi số 5 để lấy idbuffer - đây là địa chỉ của bộ đệm trong không gian người dùng để lưu dữ liệu nhận được.
    - Đọc giá trị của thanh ghi số 6 để lấy len - đây là độ dài của dữ liệu cần nhận.
  - Lấy thông tin socket và chuẩn bị bộ đệm:
    - Lấy socketid từ địa chỉ socketid trong không gian người dùng thông qua table.getFile(addrSocketid)->getSocketID().
    - Cấp phát bộ đệm buffer với độ dài là len.
  - Nhận dữ liệu thông qua hàm SocketReceive:
    - Gọi hàm SocketReceive với tham số là buffer, len, và socketid.
    - Sử dụng setsockopt để đặt timeout cho việc nhận dữ liệu.
    - Sử dụng hàm recv để nhận dữ liệu từ socket.
    - Kiểm tra giá trị trả về. Nếu có lỗi, in thông báo lỗi và trả về -1.
    - Nếu nhận thành công, trả về độ dài thực sự của dữ liệu đã nhận.
    - Nếu có lỗi, trả về -1.
  - Chuyển dữ liệu từ không gian kernel tới không gian người dùng:
    - Sao chép dữ liệu từ không gian kernel tới không gian người dùng thông qua hàm System2User và lưu vào idbuffer.
  - In thông báo và kết thúc System call:
    - In dữ liệu nhận được và thông báo thành công.
    - Tăng Program Counter và kết thúc system call.

5. System call Close

- Mục đích: int Close(int socketid) đóng socket với 0 nếu thành công hoặc -1 nếu lỗi.
- Cách hoạt động:
  - Đọc giá trị thanh ghi 4 vào ID.
  - Kiểm tra file đã mở hay chưa bằng cách duyệt qua 20 vị trí trong Table. Nếu chưa mở trả về -1 và tăng PC. Nếu đã mở thì giải phóng vùng nhớ của OpenFile*, gán vị trí trong danh sách file là NULL và set các giá trị về default.

- Ghi 0 vào thanh ghi 2 và tăng PC.

IV. CHƯƠNG TRÌNH KIỂM THỬ

A. Chương trình createfile

- Mục đích: chương trình tạo file.
- Hướng dẫn sử dụng tính năng: sử dụng cmd ../build.linux/nachos -x createfile
  - Với tên file được cho cố định, gọi Create để tạo file với tên.
  - Kiểm tra tạo file có lỗi hay không và xuất thông báo lỗi.
  - Gọi Halt để tắt máy.

B. Chương trình cat

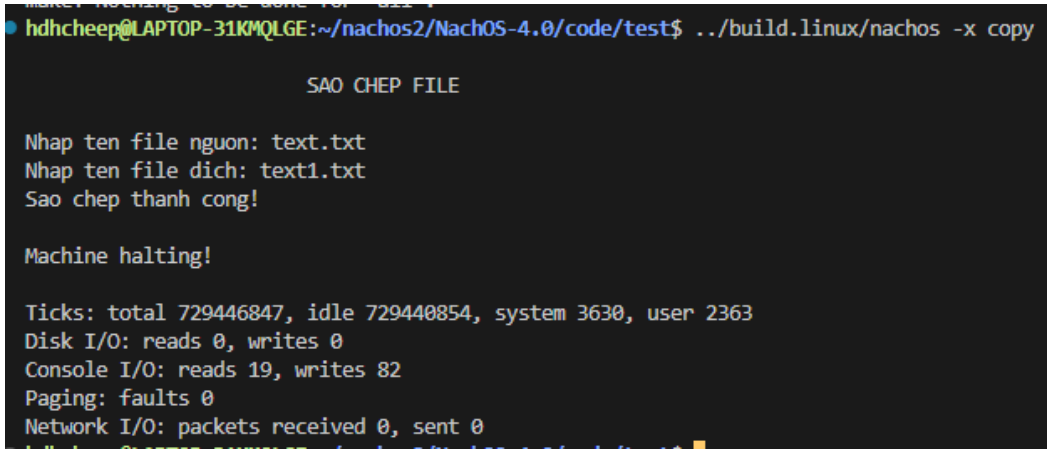
- Mục đích: hiển thị nội dung của file.
- Hướng dẫn sử dụng tính năng: ../build.linux/nachos -x cat
  - Chương trình sẽ đọc tên file người dùng nhập từ console.
  - Mở file Open và kiểm tra có lỗi hay không (-1 là lỗi), nếu không gọi Read để đọc với charCount cấp sẵn là 256 và hiển thị nội dung lên console với PrintString.
  - Đóng file Close.
  - Halt để tắt máy.

```
hdhcheep@LAPTOP-31KMQLGE:~/nachos2/NachOS-4.0/code/test$ ../build.linux/nachos -x cat

                        HIEN THI NOI DUNG FILE

Nhap vao ten file can doc: text.txt
Noi dung file:
huhuhuhu
Machine halting!

Ticks: total 305024677, idle 305021652, system 2940, user 85
Disk I/O: reads 0, writes 0
Console I/O: reads 9, writes 79
Paging: faults 0
Network I/O: packets received 0, sent 0
```

Hình 1. Chương trình cat

C. Chương trình copy

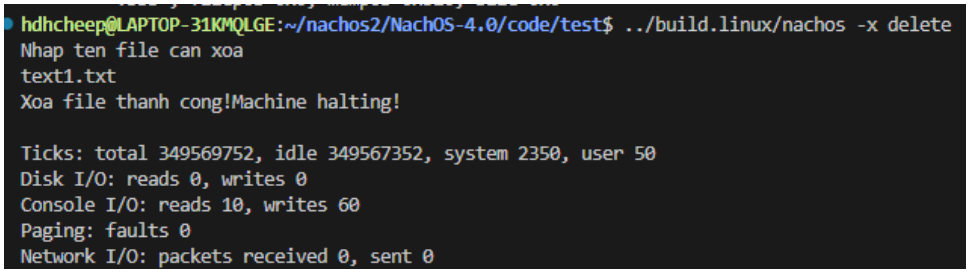
- Mục đích: copy nội dung từ file nguồn đến file đích.
- Hướng dẫn sử dụng tính năng: ../build.linux/nachos -x copy
  - Nhập tên file nguồn và file đích.
  - Chương trình đọc ID file scr bằng Open và kiểm tra có nếu không có lỗi thì tạo file dest bằng Create, đóng file Close và mở lại Open với type = 0 (read and write).
  - Kiểm tra mở file không lỗi thì ta gọi Read để đọc từ file scr vào buffer để lấy số byte thật sự hệ thống đọc được. Sử dụng Seek để dời con trỏ file scr và dest lên đầu file bắt đầu copy.
  - Duyệt qua chiều dài của buffer và đọc từng char của file scr Read đồng thời ghi từng char đó vào file dest bằng hàm Write.
  - Sau đó đóng 2 file với Close
  - Gọi Halt để tắt máy.



Hình 2. Chương trình copy

D. Chương trình delete

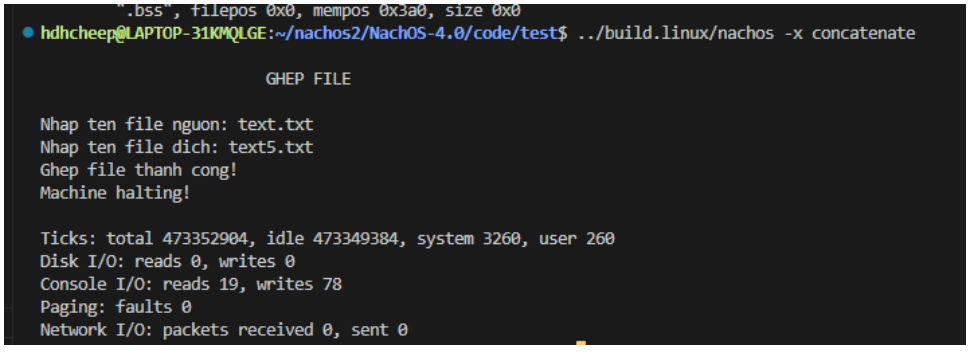
- Mục đích: xóa file
- Hướng dẫn sử dụng tính năng: ../build.linux/nachos -x delete
  - Tên file được nhập từ người dùng. Gọi Remove để xóa file
  - Nếu xóa thành công, thông báo ra console "Xoa file thanh cong!", ngược lại thông báo "Khong the xoa!".
  - Gọi Halt để tắt máy.



Hình 3. Chương trình delete

E. Chương trình concatenate

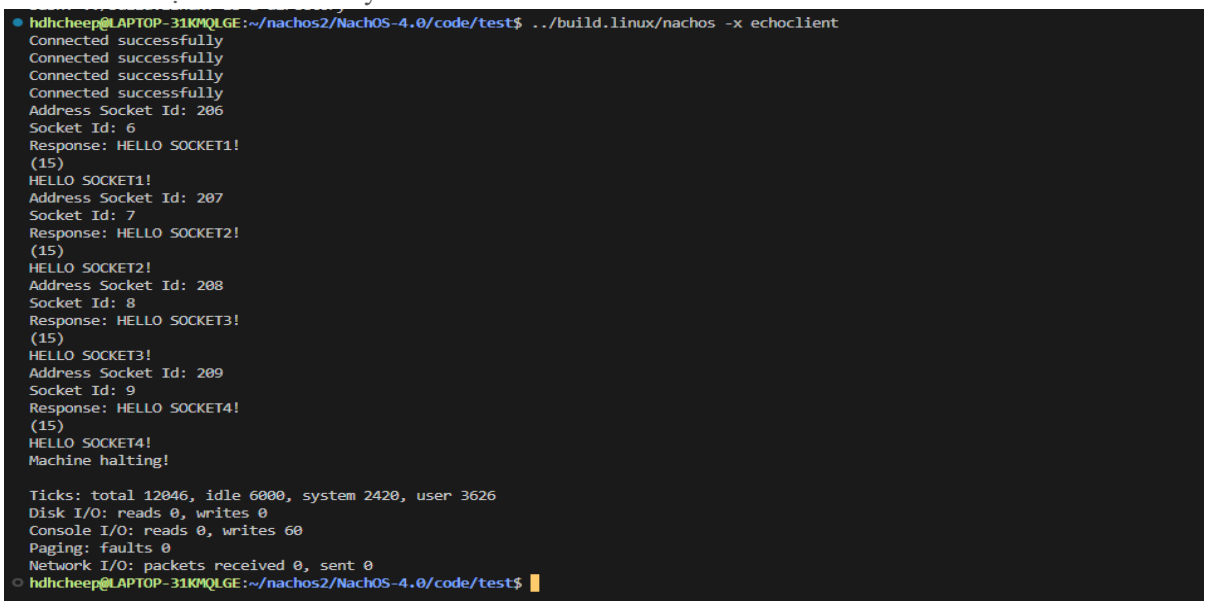
- Mục đích: nối nội dung file nguồn 1 và file nguồn 2 với nhau. Ghi vào file đích.
- Hướng dẫn sử dụng tính năng: ../build.linux/nachos -x concatenate
  - Gọi Open để mở 3 file với chế độ read and write.
  - Tiến hành đọc file scr1 vào buffer với Read, sau đó gọi Write ghi buffer vào file dest với chiều dài là chiều dài của buffer.
  - Đưa con trỏ file dest tới cuối file bằng Seek, tiếp tục thực hiện đọc file scr2 và ghi vào dest.
  - Đóng cả ba file với Close.
  - Gọi Halt để tắt máy.



Hình 4. Chương trình concatenate

F. Chương trình echo (socket network)

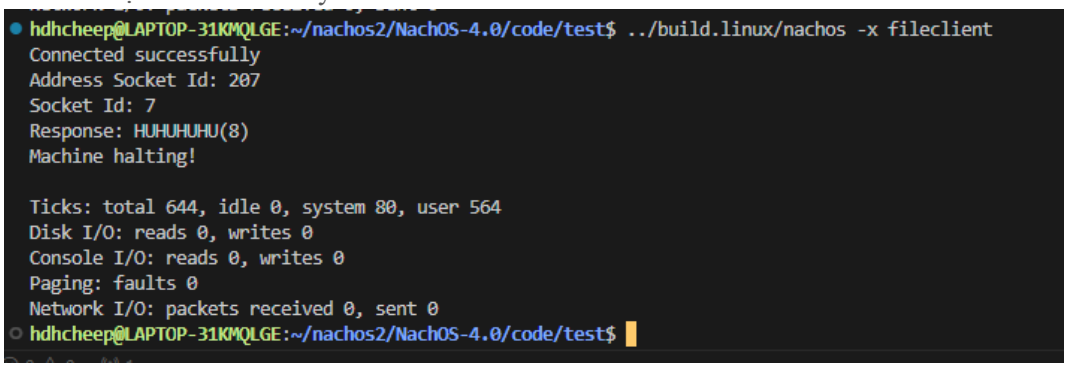
- Mục đích: Kết nối socket từ client đến server, client sẽ gửi tin đến server, sau đó server nhận và gửi ngược lại chuỗi in hoa đã nhận đến client.
- Hướng dẫn sử dụng tính năng:
  - Tạo 4 socket, với 4 fileid khác nhau và lần lượt connect đến server.
  - Lần lượt gửi và nhận các tin đến server theo thứ tự.
  - Server nhận và gửi lại chuỗi in hoa cho client.
  - Hoàn thành và close các socket đã mở.
  - Gọi Halt để tắt máy.



Hình 5. Chương trình echoclient

G. Chương trình truyền file (socket network)

- Mục đích: Đọc 1 file và ghi dữ liệu in hoa vào file khác thông qua socket
- Hướng dẫn sử dụng tính năng:
  - Đọc dữ liệu từ 1 file cho trước.
  - Sau đó tạo một socket để gửi dữ liệu lên server. Server nhận dữ liệu và gửi cho client dữ liệu in hoa.
  - Client nhận dữ liệu, ghi vào 1 file mới.
  - Đóng socket và các file đã mở.
  - Gọi Halt để tắt máy.



Hình 6. Chương trình fileclient

V. ĐÁNH GIÁ MỨC ĐỘ HOÀN THÀNH ĐỒ ÁN

Câu	Chức năng	Hoàn thành

1.1	Syscall Create	☒
1.2	Syscall Open và Close	☒
1.3	Syscall Read và Write	☒
1.4	Syscall Seek	☒
1.5	Syscall Remove	☒
2.1	Syscall socketTCP	☒
2.2	Syscall Connect	☒
2.3	Syscall Send và Receive	☒
3.1	Nâng cao	☒
4.1	Chương trình test create	☒
4.2	Chương trình test copy	☒
4.3	Chương trình test cat	☒
4.4	Chương trình test delete	☒
4.5	Chương trình test concatenate	☒
4.6	Chương trình test echo	☒
4.7	Chương trình test truyền file	☒
5	Báo cáo	☒

VI. TÀI LIỆU THAM KHẢO

Tài liệu tham khảo môn học Hệ điều hành-21_21 (2023-2024)
OperatingSystem_Project2_NACHOS_MultiProgramming-huukhanh0653
HuongdanNachos-NguyenThanhChung