**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN – ĐHQG TPHCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

**BÁO CÁO ĐỒ ÁN**

**Môn: HỆ ĐIỀU HÀNH**

**PROJECT 1 : EXCEPTIONS VÀ CÁC SYSTEMCALLS**

*Giảng viên: Lê Giang Thanh*

*Nguyễn Thanh Quân*

Thành viên:

20127064 – Nguyễn Trần Mai Phương

20127090 – Lê Thanh Tú

20127344 – Võ Hiền Hải Thuận

*NK 2021 - 2022*

Mục lục

1. Mức độ hoàn thiện 2
2. Cài đặt trước 3
3. Cài đặt các system calls 4
4. Tài liệu tham khảo 10

**MỨC ĐỘ HOÀN THIỆN**

|  |  |
| --- | --- |
| **Nội dung** | **Hoàn thành** |
| Hiểu mã NachOS | ✓ |
| Hiểu thiết kế | ✓ |
| Xử lý exceptions | ✓ |
| IncreasePC | ✓ |
| ReadNum | ✓ |
| PrintNum | ✓ |
| ReadChar | ✓ |
| PrintChar | ✓ |
| ReadChar | ✓ |
| RandomNum | ✓ |
| ReadString | ✓ |
| PrintString | ✓ |
| Create,Remove | ✓ |
| Open,Close | ✓ |
| Read | ✓ |
| Seek | ✓ |
| Help, ASCII, Sort | ✓ |
| Createfile, cat, copy, delete | ✓ |
| Không để user làm sụp hệ điều hành | ✓ |
| Báo cáo | ✓ |

**CÀI ĐẶT TRƯỚC**

Vào thư mục userprog mở file exception.cc cài đặt hai hàm

* char\* **User2System**(int virtAddr, int limit) để sao chép vùng nhớ từ user => system
* int **System2User**(int virtAddr, int len, char\* buffer) để sao chép vùng nhớ từ system => user.

*Các giá trị thanh ghi:*

R2: Lưu mã syscall và lưu kết quả trả về của syscall.

R4: Lưu tham số thứ nhất.

R5: Lưu tham số thứ hai.

R6: Lưu tham số thứ ba.

R7: Lưu tham số thứ tư.

*Khai báo biến toàn cục SynchConsole:*

Mục đích: Sử dụng các hàm mặc định của SynchConsole để đọc và ghi, trả về đúng giá trị cho user. Trong trường hợp đọc hay ghi vào console bị lỗi thì trả về -1. Nếu đang đọc từ console mà đã hết file thì trả về -2

Cách cài đặt: Vào thư mục threads trong file system.h và system.cc thực hiện khai báo, cấp phát, và xóa vùng nhớ cấp phát một biến toàn cục thuộc lớp “SynchConsole” để hỗ trợ việc nhập xuất với màn hình console.

**CÀI ĐẶT CÁC SYSTEM CALL**

**Câu 1: Viết lại file exception.cc để xử lý tất cả các exceptions được liệt kê trong machine/machine.h**

Xử lý tất cả các exceptions được liệt kê trong machine/machine.h. Với tất cả các exceptions khác, HĐH hiển thị ra một thông báo lỗi và Halt hệ thống.

**Câu 2: Viết mã để tăng gia trị biến program counter**

Cài đặt hàm increasePC(): Tăng program counter lên vì các system call sẽ yêu cầu Nachos tăng program counter trước khi trả về kết quả, cũng là để nạp lệnh tiếp theo được thực hiện. Hàm increasePC() được dùng sau khi xử lí xong exceptions.

Lưu lại giá trị PC hiện tại vào PC trước rồi tăng giá trị của PC hiện tại lên 4 và lưu lại giá trị của PC kế tiếp.

**Câu 3: Cài đặt hàm system call int ReadNum()**

Mô tả: Input: không có

Output: giá trị số nguyên nhập vào từ console

Mục đích: Trả về giá trị người dùng đã nhập vào.

Cách cài đặt:

Sử dụng lớp SynchConsoleIn

1. Lấy kí tự vào mảng buffer
2. Kiểm tra số âm/ dương
3. Kiểm tra số nguyên, nếu không phải là số nguyên (có các kí tự khác, số thập phân) thì return 0
4. Chuyển char => int
5. Lưu vào R2 giá trị kết quả và trả về số nguyên.

**Câu 4: Cài đặt system call void PrintNum(int number)**

Mô tả: Input: số nguyên

Output: Không có

Mục đích: In ra số nguyên trên màn hình console

Cách cài đặt:

Sử dụng lớp SynchConsoleOut

Nếu tham số = 0 thì in ra màn hình số 0

1. Kiểm tra âm/dương
2. Đếm số chữ số
3. Chuyển số => char
4. In ra màn hình

**Câu 5: Cài đặt system call char ReadChar()**

Mô tả: Input: Không có

Output: char

Mục đích: trả về 1 kí tự do người dùng nhập vào

Cách cài đặt:

Sử dụng GetChar trong thư viện synchconsole.h kernel->synchConsoleIn->GetChar(); để nhận input cho đến khi người dùng nhập enter, đếm số kí tự đã nhập. Nếu số kí tự là 1 thì lưu giá trị vào R2 và trả về giá trị kí tự, nếu không thì báo lỗi và yêu cầu nhập lại.

**Câu 6: Cài đặt system call void PrintChar(char character)**

Mô tả: Input: char c

Output: Không có

Mục đích: In ra màn hình kí tự c

Cách cài đặt:

Sử dụng PutChar trong thư viện synchconsole.h kernel->synchConsoleOut->PutChar(c) để in ra màn hình console

**Câu 7: Cài đặt system call int RandomNum()**

Mô tả: Input: Không có

Output: số nguyên

Mục đích: Trả về một giá trị số nguyên ngẫu nhiên.

Cách cài đặt:

Dùng hàm rand() để tạo ngẫu nhiên một số nguyên và lưu vào biến random và trả về giá trị đó.

**Câu 8: Cài đặt system call void ReadString (char\* buffer, int length)**

Mô tả: Input: char\* buffer, độ dài chuỗi

Output: Không có

Mục đích: đọc chuỗi và lưu vào buffer

Cách cài đặt:

1. Set buffer là ‘\0‘ : memset(buff, 0, limit + 1)
2. Đọc kí tự và lưu vào buffer
3. Dùng GetChar trong thư viện synchconsole.h: kernel->synchConsoleIn->GetChar() để đọc từng kí tự
4. Copy giá trị từ User memory space => System memory space: System2User(address, limit + 1, buff)

**Câu 9: Cài đặt system call void PrintString (char\* buffer)**

Mô tả: Input: char\* buffer

Output: không có

Mục đích: In ra màn hình chuỗi string

Cách cài đặt:

1. Lấy chuỗi từ địa chỉ của buffer: User2System(address, 255)
2. Tạo vòng lặp để in từng kí tự trong chuỗi string: kernel->synchConsoleOut->PutChar(buffer[i])

**Câu 10: Cài đặt system call int Create(char \*name)**

Mô tả: Input: địa chỉ của tên file

Output: kết quả thành công/thất bại

Mục đích: tạo ra 1 file

Cách cài đặt:

1. Lấy tên file từ địa chỉ: User2System(virtAddr, MaxFileLength + 1)
2. Kiểm tra tên file có tồn tại không
3. Dùng Create trong filesys.h để tạo file: kernel->fileSystem->Create(filename)

**Câu 11: Cài đặt system call OpenFileID Open(char \*name) và int Close(OpenFileID id)**

1. *OpenFileID Open(char\* name)*

Mô tả: Input: địa chỉ tên file

Output: trả về OpenFileID nếu thành công, -1 nếu thất bại

Mục đích: Mở file và trả về giá trị OpenFileID

Cách cài đặt:

1. Kiểm tra số lượng file đã mở, chương trình báo lỗi khi số lượng file vượt quá 10.
2. Lấy địa chỉ file: User2System(bufAddr, MaxFileLength + 1)
3. Xác định mode của file là stdin hay stdout
4. Mở file: kernel->fileSystem->openf[kernel->fileSystem->index] = kernel->fileSystem->Open(buf))
5. *Int Close(OpenFileID id)*

Mô tả: Input: ID của file

Output: 0 nếu thành công, -1 nếu thất bại

Mục đích: Đóng file

Cách cài đặt:

1. Lấy tên file từ địa chỉ: User2System(id, MaxFileLength + 1)
2. Kiểm tra file có tồn tại không và đóng file:

delete kernel->fileSystem->openf[id];

kernel->fileSystem->openf[id] == NULL;

**Câu 12: Cài đặt system call int Read(char \*buffer, int size, OpenFileID id) và int Write(char**

**\*buffer, int size, OpenFileID id)**

1. *Int Read(char\* buffer, int size, OpenFileID id)*

Mô tả: Input: địa chỉ của biến để lưu giá trị trả về, số lượng kí tự muốn read, ID của file.

Output: số bytes đã đọc được hoặc -1 nếu thất bại

Mục đích: đọc file, lưu vào biến và trả về số bytes đã đọc được.

Cách cài đặt:

1. Kiểm tra file có tồn tại không:
2. Xác định loại file (stdin/stdout)
3. Dùng void ReadString(char\* buffer, int length) để đọc dữ liệu
4. Lưu buff vào địa chỉ của biến, trả về độ dài file đã đọc được
5. *Int Write(char\*buffer, int size, OpenFileID id)*

Mô tả: Input: địa chỉ của biến để lưu giá trị trả về, số lượng kí tự muốn write, ID của file

Output: Số bytes đã viết được hoặc -1 nếu thất bại

Mục đích: viết vào file, lưu dữ liệu đã viết được vào buffer và trả về số bytes đã viết

Cách cài đặt

1. Kiểm tra file có tồn tại không:
2. Xác định loại file (stdin/stdout)
3. Gọi hàm void PrintString(char\* buffer) để in dữ liệu viết được ra màn hình console.
4. Lưu buff vào địa chỉ của biến, trả về độ dài file đã đọc được

**Câu 13: Cài đặt system call int Seek(int position, OpenFileID id)**

Mô tả: Input: số nguyên position chỉ vị trí cần tìm, ID của file

Output: vị trí cần tìm hoặc -1 nếu không tìm thấy

Mục đích: di chuyển con trỏ đến vị trí, biến position lưu vị trí cần chuyển tới

Cách cài đặt:

1. Kiểm tra file có tồn tại không
2. So sánh position với độ dài của file, nếu pos lớn hơn thì lưu pos = -1 và chuyển con trỏ đến vị trí cuối file, lưu giá trị pos bằng độ dài của file: kernel->machine->WriteRegister(2, độ dài)
3. Di chuyển con trỏ đến vị trí đó và lưu giá trị của pos: kernel->fileSystem->openf[id]->Seek(pos)

**Câu 14: Cài đặt system call int Remove(char \*name)**

Mô tả: Input: địa chỉ của tên file

Output: 0 nếu thành công, -1 nếu thất bại

Mục đích: Remove một file

Cách cài đặt:

1. Lấy filename từ địa chỉ được truyền vào: User2System(virtAddr1, MaxFileLength + 1)
2. Kiểm tra file có tồn tại không
3. Kiểm tra file có tồn tại trong folder không: check(openning, kernel->fileSystem->index)
4. Dùng hàm Remove trong filesys.h để remove file: kernel->fileSystem->Remove(filename)

**Câu 15: Cài đặt chương trình help**

Mô tả : Input: Không có

Output: Hướng dẫn chương trình

Mục đích : Cung cấp thông tin

Cách cài đặt: Dùng PrintString(char[]) để in chuỗi ra màn hình

Text

Description automatically generated

**Câu 16: Cài đặt chương trình Ascii**

Mô tả : Input: Không có

Output: Kí tự bảng mã ASCII

Mục đích: In ra màn hình các kí tự của bảng mã ASCII

Cách cài đặt: Dùng PrintChar() để in ra các kí tự và PrintNum() để in ra các số

Text

Description automatically generated

**Câu 17: Cài đặt chương trình Sort**

Mô tả : Input: Số nguyên

Output: dãy số nguyên đã được sắp xếp

Mục đích: Sắp xếp các số

Cách cài đặt:

1. Dùng ReadNum để lấy số lượng, phần tử mảng, loại sort (tăng/giảm)
2. Sort và in ra màn hình kết quả sắp xếp.

**Câu 18: Viết chương trình createfile:**

Mô tả : Input: Tên file

Output: In ra dòng chữ thành công hay thất bại

Mục đích: Kiểm tra SC\_Create

Cách cài đặt :

1. Nhập vô tên file
2. Dùng Create để tạo file
3. Dùng PrintString in ra kết quả

A screenshot of a computer

Description automatically generated with medium confidence

**Câu 19: Viết chương trình cat**

Mô tả : Input: filename

Output: Nội dung của file

Mục đích: Hiển thị nội dung của file

Cách cài đặt

1. Đọc tên file (ReadString), mở file (Open) và lưu giá trị ID của file
2. Kiểm tra tính hợp lệ của file
3. Dùng hàm Read() để đọc file và lưu vào biến tạm
4. In nội dung đã đọc ra màn hình console (Print)
5. Dùng Close() để đóng file
6. Graphical user interface

   Description automatically generated

**Câu 20: Viết chương trình copy**

Mô tả : Input: filename nguồn và đích

Output: Kết quả chương trình có thành công hay không

Mục đích: Copy nội dung từ file nguồn => file đích

Cách cái đặt :

1. Đọc tên và mở 2 file
2. Dùng hàm Read() để đọc file nguồn và lưu vào biến tạm
3. Dùng hàm Write() để viết nội dung của biến tạm vào file đích
4. In ra màn hình kết quả copy thành công hay thất bại
5. Dùng Close() để đóng file
6. A screenshot of a computer

   Description automatically generated with medium confidence

**Câu 21:Viết chương trình Remove**

Mô tả : Input: filename

Output: Kết quả thành công hay không

Mục đích : Kiểm tra SC\_Remove

Cách cái đặt :

1. Dùng Remove() để xóa file
2. In ra màn hình kết quả

A screenshot of a computer

Description automatically generated with medium confidence

**TÀI LIỆU THAM KHẢO**

**Github:**

<https://github.com/nguyenthanhchungfit/Nachos-Programing-HCMUS?fbclid=IwAR1n7veuHvC5cNj1IR5D6eoTkdjgoSj0lSYxl7OqUDYiZ1wRXNcVyspd4fA>

https://github.com/king1358/nachos

https://github.com/hieulm2k/HCMUS-PROJECT\_Nachos-OS-HCMUS