Báo cáo đồ án 1

1. Phân công

Thành Viên	Viết mã	Kiểm tra	Viết báo cáo
19120553 - Chung Hoàng Tuấn Kiệt	☑	☑	☐
19120028 - Vũ Hữu Nghĩa	☑	☐	☑
19120159 - Trần Huy Vũ	☐	☑	☑

2. Thiết kế

2.1. Sửa đổi file.

2.1.1. File userprog/exception.cc

- Thêm các case exception
- Thêm các case cho từng loại syscall exception

2.1.2. File userprog/syscall.h

- Thêm định nghĩa system call codes cho các hàm.
- Định nghĩa tất cả các prototype cho các systemcalls.

2.1.3. File userprog/ksyscall.h

- Viết các hàm Kernel interface cho systemcalls

2.2. Tạo mới file.

2.2.1. File test/ascii.c

- Viết hàm ascii() in ra bảng mã ascii.

2.2.2. File test/sort.c

- Viết hàm bubblesort() sắp xếp n phần từ nhập từ bàn phím và tùy chọn tăng giảm.

2.2.3. File test/help.c

- Viết chương trình **help**.

3. Cài Đặt

3.1. Viết lại file **exception.cc** .

- Sử dụng biến which để nhận biết loại exception và xử lý bằng switch case.
- Tại case SyscallException sử dụng switch case biến type để xử lý các user syscalls.
- Tại mỗi case ngoại trừ no exception, còn lại xuất ra thông báo và gọi hàm SysHalt().

3.2. Viết lại cấu trúc điều khiển và kiểm tra.

- Tại switch case biến type thêm các case và gọi hàm tương ứng cho
  + SC_Exit
  + SC_Exec

+ SC_Join
+ SC_Create
+ SC_Remove
+ SC_Open
+ SC_Read
+ SC_Write
+ SC_Seek
+ SC_Close
+ SC_ThreadFork
+ SC_ThreadYield
+ SC_ExecV
+ SC_ThreadExit
+ SC_ThreadJoin

3.3. Tăng program counter.

- Ta sẽ cần thay đổi giá trị ba thanh ghi PrevPCReg, PCReg, NextPCReg để Nachos không bị vòng lặp mãi mãi.
- Ta sử dụng hàm WriteRegister trong kernel để thay đổi giá trị của các thanh ghi trên.
- Cách sử dụng hàm trên ta có thể tham khảo tại hàm SC_Add có sẵn.

3.4. Cài đặt system call **int ReadNum()**.

System call ReadNum() được cài đặt sẽ sử dụng lớp SynchConsoleIn để nhập một số nguyên do người dùng nhập vào. Tuy nhiên, lớp này chỉ hỗ trợ hàm **GetChar()** chỉ cho phép đọc một ký tự, thế nên ta cần một biến res kiểu long để lưu trữ số nguyên thu được sau mỗi lần người dùng nhập vào một kí tự

- Khi người dùng nhập kí tự đầu tiên, kiểm tra xem kí tự đó là số hay là dấu "-" (để nhận biết số âm)
- Kế tiếp đó là vòng lặp **while(True)**, lần lượt đọc từng ký tự do người dùng nhập vào, nếu kí tự đó là số thì sẽ ghép vào cuối res để cập nhật biến này
- Mỗi lần cập nhật res, giá trị tuyệt đối của nó sẽ càng lớn dần, dẫn đến khả năng tràn số (do xảy ra khả năng người dùng nhập vào số nguyên quá lớn, vượt quá phạm vi của số nguyên trong C), thế nên trước khi đọc ký tự tiếp, chúng tôi so sánh |res| với một số nguyên dương lớn là **maxInt = 65000** (gần với phạm vi biểu diễn của int trong C), nếu lớn hơn số này thì trả về số 0

Khi người dùng kết thúc việc nhập, kí tự cuối cùng ta nhận được lúc này là **"\0"** hoặc **"\n"** (kí tự xuống dòng) hoặc **" "** (ký tự khoảng trắng). Lúc này ta lưu biến res này lại, ghi vào thanh ghi số 2, tăng giá trị thanh ghi PC lên 4.

Ngoại lệ: Mỗi khi đọc một ký tự do người dùng nhập vào, nếu kí tự đó không phải là kí tự số thì trả về số 0.

3.5. Cài đặt system call **void PrintNum(int number)**.

Đầu tiên, lấy giá trị lưu trữ trong thanh ghi số 4 ra và lưu vào number

Vì lớp SynchConsoleOut hỗ trợ hàm **PutChar(char ch)** chỉ cho phép in một kí tự ra màn hình nên ta sẽ in lần lượt các chữ số của số number đó:

- Kiểm tra số đó có phải là số âm không. Nếu có, in kí tự "-"
- Sau đó in lần lượt các chữ số theo thứ tự từ trái sang phải.

Cuối cùng, tăng giá trị thanh ghi PC lên 4.

3.6. Cài đặt system call **char ReadChar()**.

- Sử dụng hàm **GetChar()** trong lớp SynchConsoleIn để đọc kí tự do người dùng nhập vào.
- Chuyển kí tự thu được dưới dạng số nguyên kiểu int và lưu vào thanh ghi số 2
- Tăng giá trị thanh ghi PC lên 4.

3.7. Cài đặt system call **void PrintChar(char character)**.

- Đầu tiên, lấy giá trị số nguyên lưu trong thanh ghi số 4 và ép kiểu (char) biến thành ký tự
- Sử dụng hàm **PutChar(char ch)** thuộc lớp SynchConsoleOut để in ký tự đó ra màn hình
- Tăng giá trị thanh ghi PC lên 4

3.8. Cài đặt system call **int RandomNum()**.

- System call này sẽ dùng hàm **srand(time(NULL))** để trả về một số nguyên dương ngẫu nhiên. Tuy nhiên ta muốn giá trị này nằm trong phạm vi biểu diễn của kiểu số nguyên int (từ 0 đến 65535) nên ta thực hiện phép chia lấy phần dư cho 65536.
- Kết quả thu được lưu vào thanh ghi số 2, và tăng giá trị thanh ghi PC lên 4

3.9. Cài đặt system call **void ReadString (char[] buffer, int length)**.

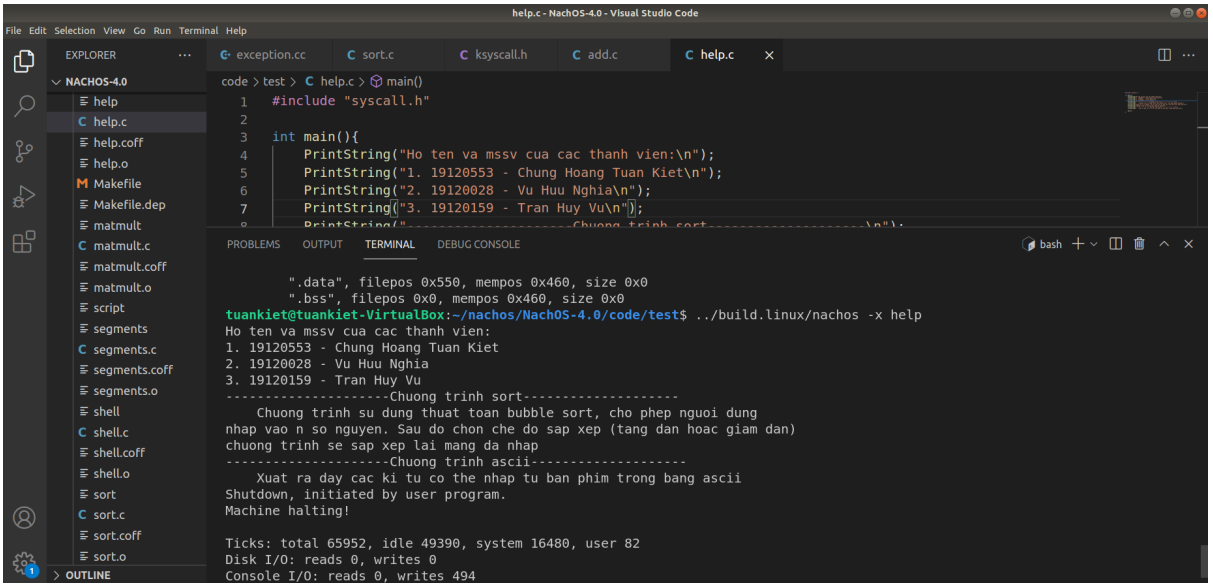
- Đầu tiên, ta tạo ra một mảng tạm kernelBuff kiểu **char*** (cấp phát động) để lưu trữ chuỗi ký tự do người dùng nhập vào. Sau đó dùng hàm **GetChar()** thuộc lớp SynchConsoleIn để đọc từng ký tự người dùng nhập vào và lưu vào kernelBuff. Khi người dùng kết thúc việc nhập, đặt ký tự kết thúc chuỗi **kernelBuff[length] = "\0"**
- Tuy nhiên, kernelBuff lại nằm ở trong MainMemory của chương trình giả lập, trong khi buffer là vùng nhớ thuộc về userspace. Do đó ta viết hàm **int System2User (buffer, length, kernelBuff)** để chuyển dữ liệu từ kernelBuff ở trong kernelspace sang buffer ở userspace.

3.10. Cài đặt system call **void PrintString (char[] buffer)**.

- System call này để in chuỗi kí tự trong buffer ra màn hình. Ở đây ta sẽ dùng hàm **PutChar(char ch)** thuộc lớp SynchConsoleOut để in từng ký tự của chuỗi ra màn hình.
- Tuy nhiên muốn in được thì dữ liệu phải nằm ở kernelspace, trong khi chuỗi ký tự buffer ta đang có đang ở userspace. Do đó ta sẽ viết hàm **char* User2System(int buffer, int limit)** để chuyển dữ liệu từ buffer sang vùng nhớ thuộc kernelspace và trả về địa chỉ vùng nhớ đó

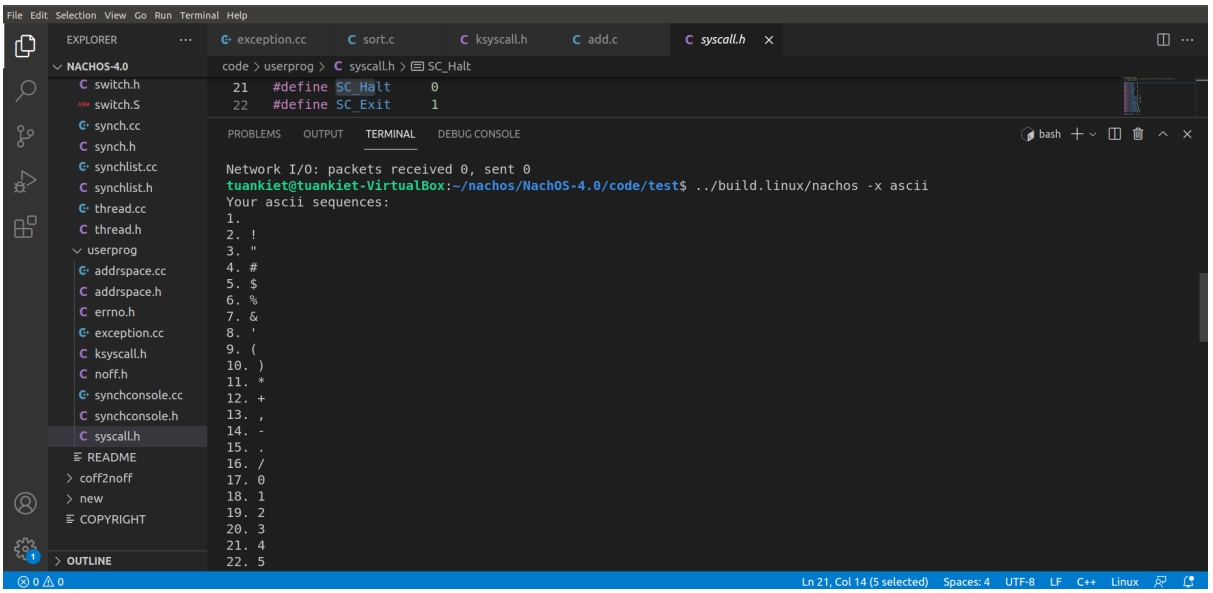
3.11. Viết chương trình **help**.

Chương trình **help** dùng để in ra các dòng giới thiệu cơ bản về nhóm và mô tả vắn tắt về chương trình **sort** và **ascii**. Thực chất việc này chỉ là xuất chuỗi ra màn hình, ta chỉ cần gọi liên tục system call **void PrintString (char[] buffer)**



3.12. Viết chương trình **ascii**

Chương trình này in ra các bảng mã ascii gồm các kí tự đọc được (các kí tự có mã ascii từ 32 đến 127) nên ta thực hiện một vòng lặp for, mỗi lần lặp ta in ra ký tự đó bằng system call **void PrintChar(char ch)**.

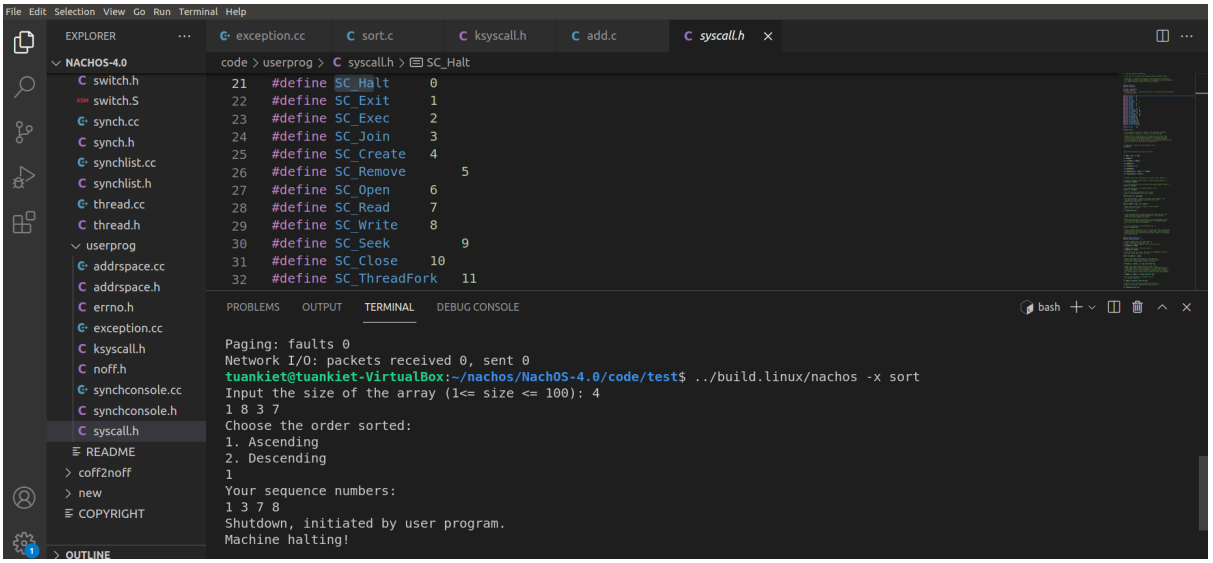


3.13. Viết chương trình **sort**

Chương trình sort nhằm sắp xếp mảng các số nguyên do người dùng nhập vào theo thứ tự tăng dần hoặc giảm dần, sử dụng giải thuật Bubble Sort.

Quá trình thiết kế:

+ Yêu cầu người dùng nhập vào số phần tử của mảng, cũng như các phần tử của mảng: Sử dụng system call **int ReadNum()**.
+ Xuất thông báo lựa chọn thứ tự sắp xếp mảng (tăng dần hay giảm dần): Sử dụng system call **void PrintString (char[] buffer)**
+ Nhận lựa chọn của người dùng
+ Sắp xếp mảng theo thứ tự người dùng yêu cầu
+ Xuất mảng đã được sắp xếp: Sử dụng system call **void PrintNum(int number)**.


File Edit Selection View Go Run Terminal Help
EXPLORER
NACHOS-4.0
switch.h
switch.S
synch.cc
synch.h
synchlist.cc
synchlist.h
thread.cc
thread.h
userprog
addrspace.cc
addrspace.h
errno.h
exception.cc
ksyscall.h
noff.h
synchconsole.cc
synchconsole.h
syscall.h
README
coff2noff
new
COPYRIGHT
OUTLINE
exception.cc
sort.c
ksyscall.h
add.c
syscall.h
code > userprog > syscall.h > SC_Halt
21 #define SC_Halt 0
22 #define SC_Exit 1
23 #define SC_Exec 2
24 #define SC_Join 3
25 #define SC_Create 4
26 #define SC_Remove 5
27 #define SC_Open 6
28 #define SC_Read 7
29 #define SC_Write 8
30 #define SC_Seek 9
31 #define SC_Close 10
32 #define SC_ThreadFork 11
PROBLEMS
OUTPUT
TERMINAL
DEBUG CONSOLE
bash
Paging: faults 0
Network I/O: packets received 0, sent 0
tuankiet@tuankiet-VirtualBox:~/nachos/NachOS-4.0/code/test$ ../build.linux/nachos -x sort
Input the size of the array (1<= size <= 100): 4
1 8 3 7
Choose the order sorted:
1. Ascending
2. Descending
1
Your sequence numbers:
1 3 7 8
Shutdown, initiated by user program.
Machine halting!