ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

20127643 - TRƯƠNG GIA TIẾN

20127664 - NGUYỄN VĂN VIỆT

20127669 - NGÔ ANH VŨ

ĐỒ ÁN NACHOS 1

| GIẢNG VIÊN HƯỚNG DẪN |

Ths. Nguyễn Văn Giang

Ths. Lê Quốc Hòa

Môn:Hệ điều hành

MỤC LỤC

THÀNH VIÊN NHÓM

20127643 – Trương Gia Tiến

20127664 – Nguyễn Văn Việt

20127669 – Ngô Anh Vũ

THÔNG TIN CHUNG VỀ ĐỒ ÁN

Hệ điều hành và công cụ dùng

Hệ điều hành sử dụng: CentOS (bản đã built sẵn do Giảng viên cung cấp)

Công cụ máy ảo: VM Ware Workstation 16 Player.

Hệ điều hành NachOS:

Bao gồm 2 file tải từ moodle: nachos.rar và synchcons.rar (đã thêm vào nachos)

Mô tả thiết kế. Các system call được define ở syscall.h thuộc folder userprog, với cấu trúc #defined SC_<Tên syscall> số tương ứng. Và khai báo các hàm tương ứng ở đó. Phần thực hiện chính chủ yếu được cài đặt trong file exception.cc . File có mục đích là bắt cách exception và xử lý nó. Nếu không có exception nào thì chương trình sẽ tự động Halt(). Trong file chủ yếu xử lý các exception liên quan đến system call, còn các exception khác hệ thống sẽ chỉ in ra lỗi trên console. Trong switch exception luên quan đến system call thì có các system call là : SC_Halt, SC_ReadNum, SC_PrintNum, SC_ReadChar, SC_PrintChar, SC_RanNum, SC_ReadString, SC_PrintString. Mỗi exception trên sẽ được xử lý riêng, cụ thể như sau:

Ý TƯỞNG VÀ GIẢI THÍCH Ý TƯỞNG CỦA CÁC SYSTEM CALL

System call ReadNum

// Tham số: không

// Output: trả về một số nguyên do người dùng nhập theo kiểu **int**

Ý tưởng: người dùng nhập vào 1 chuỗi số (**char *numString**) thông qua hàm **SynchConsole.Read** . Chuỗi số này sẽ có độ dài max là 32 bytes, tương đương với 31 ký

tự + 1 ký tự '\0' (độ dài này do người viết chương trình quyết định). Việc cho độ dài chuỗi có max = 32 bytes ở trên là để trừ trường hợp người dùng nhập một chuỗi dài --> có thể làm sụp hệ thống.

Do kết quả trả về sẽ là kiểu int -> max = 2147483648, min = -2147483648 --> độ dài chuỗi hợp lệ sẽ là 11 bytes đối với số dương và 12 bytes đối với số âm (do số âm có thêm '-'). Ta sẽ để **maxLength = 12** và sẽ check các điều kiện để điều chỉnh lại **maxLength** sau.

Kết quả trả về của hàm Read thuộc lớp SynchConsole là số byte đã đọc (hay nói cách khác là số byte mà người dùng đã nhập). Tạo một biến để lưu số byte này lại (**readBytes)**. Sau đó ta sẽ check xem số byte này có hợp lệ hay không. Có 2 trường hợp:

- Trường hợp 1 - số vừa nhập là số dương (**numString[0] != '-'**) --> **readBytes < 11**
- Trường hợp 2 - số vừa nhập là số âm (**numString[0] = '-'**) --> **readBytes < 12**
- Nếu **readBytes** ở cả 2 trường hợp đều không thỏa ==> kết quả = 0 ==> **machine->WriteRegister(2, 0);**

Nếu **readBytes** thỏa:

- Trường hợp 1: **--maxLength //** lúc đầu **maxLength = 12** (ở trên) và do đây là số dương nên **maxLength = 11**)
- Trường hợp 2: **start = 1** // lúc đầu **start = 0** (bắt đầu chuỗi tại vị trí 0) nhưng do đây là số âm nên có thêm ký tự '-' nên chuỗi số của ta sẽ bắt đầu tại vị trí 1

Tiếp đến, ta sẽ duyệt chuỗi số vừa nhập và đổi từ chuỗi ra số kiểu int với vòng for có:

- **i = start** // bắt đầu tại vị trí **start** đã xử lý ở trên
- Điều kiện dừng: **i < maxLength && numString[i] != '\0'** // lý do xét ký tự '\0' vì user có thể nhập chuỗi không cần đến vị trí **maxLength – 1.** vd: 123, 12353153,… vì vậy cần xét '\0' để biết chuỗi kết thúc tại đâu.

Ở trong vòng lặp **for** này, ta sẽ có **if(numString[i] < '0' || numstring[i] > '9')** để check xem chuỗi đã nhập có chứa ký tự nào khác số (ở dạng char) hay không. Nếu có thì chuỗi này không phải là số ==> kết quả xuất ra sẽ là 0 ==> **machine->WriteRegister(2, 0);** và **return**, nếu không thì tính **result = result * 10 + numString[i] - '0'** // chuyển từ kí tự char sang số.

Sau khi đã tính xong kết quả **result**, ta xét các trường hợp đặc biệt đó là min và max của kiểu int. Do ta chưa gán dấu '-' (trong trường hợp là số âm) cho **result** nên ta phải xét trường hợp **result = -2147483648** nếu thỏa trường hợp này thì ghi kết quả này vào thanh ghi 2 **machine->WriteRegister(2, -result);**

Tiếp đến, xét trường hợp **result > 2147483647 ==>** dù là số âm hay dương đều đã vượt quá khoảng của int (do ta đã xét TH –2147483648) ==> **result = 0.**

Cuối cùng xét có phải số âm không ? (**numString[0] == '-')** nếu phải thì **result = -result.**

Ghi kết quả cuối cùng vào thanh ghi số 2.

System call PrintNum

// Tham số: **int**

// Output: In ra số nguyên **int**

Là system call có chức năng in ra console một số nguyên (kiểu int). Số nguyên này là tham số mà người dùng truyền vào hàm.

Ý tưởng:

Đầu tiên lấy tham số mà user truyền vào hàm ở thanh ghi số 4:

int Num = **machine->ReadRegister(4);**

Sau đó ta sẽ dùng **SynchConsole.Write** để in số Num ra console. Do hàm **Write** này chỉ nhận tham số là dạng chuỗi (char*) nên ta phải chuyển số nguyên này về dạng chuỗi. Do đối với 1 số thì ta chỉ có thể lấy từng chữ số của số đó theo chiều từ dưới lên, nên có:

- biến **end = 0/1 (1-s**ố âm**, 0-s**ố dương**)** chỉ vị trí kết thúc của vòng lặp đổi từ số sang chuỗi (do xét và thêm ký tự vào chuỗi ngược từ dưới lên)
- **totalDigit** để đếm số lượng chữ số của số Num
- **char*numString** là chuỗi ta sẽ xuất ra console

Ta chia ra 3 trường hợp:

- Trường hợp 1 – Num bằng 0: in ra console số 0 và kết thúc
- Trường hợp 2 – Num là số âm: lúc này thì độ dài chuỗi sẽ là 12 và **numString[0] = '-', ++totalDigit** (do có thêm '-') và **end = 1**;
- Trường hợp 3 – Num là số dương: độ dài max của chuỗi là 11

Ta có 1 vòng while để đếm số lượng các chữ số có trong **Num**

Vòng For để thêm các chữ số ở dạng char vào trong chuỗi với:

(int i= totalDigit – 1; I >= end; --i, num /=10) và

numString[i] = (char)(num % 10 + '0');

Cuối cùng dùng **SynchConsole.Write(numString, totalDigit)** để in chuỗi ra console.

System call ReadChar

// Tham số: không

// output: trả về 1 ký tự char trong bảng mã ASCII

Syscall này cho phép user nhập vào một ký tự char trong bảng mã ASCII.

Ý tưởng: con trỏ **char *result = new char[Bytes]** (**Bytes = 128** – do ta tự quy định) sẽ được dùng để lưu ký tự mà user nhập vào. Lý do sử dụng char * là do người dùng có thể sẽ nhập không phải chỉ một ký tự mà có thể nhiều ký tự, nhưng mục đích của syscall này chỉ cho phép nhập một. Vì vậy dùng char * để check xem người dùng đã nhập bao nhiêu ký tự để có thể báo lỗi.

Ta có biến **int readBytes = SynchConsole.Read(result, 128)**; để đọc số bytes (số ký tự mà người dùng đã nhập).

Ta có 3 trường hợp:

- Trường hợp 1 – **readBytes > 1**: tức là user nhập nhiều hơn một ký tự, chương trình sẽ báo lỗi và dừng lại.
- Trường hợp 2 – **readBytes = 1**: Ghi ký tự vừa nhập vào thanh ghi 2 thông qua lệnh **machine->WriteRegister(2, (int)result[0])**;
- Trường hợp 3 – **readBytes = 0**: tức là user không nhập gì cả, báo cho người dùng biết và dừng hệ thống.

Symtem call PrintChar

// Tham số: 1 ký tự char

//output: In ra một kí tự char đã truyền vào

Là system call có chức năng in ra một kí tự bất kì trong bảng mã ascii. Sử dụng thông qua hàm PrintChar(); được khai báo trong file syscall.h : **void PrintChar()**. System call này được define là **#defined SC_PrintChar 18**.

Khi hệ thống rơi vào trường hợp này case SC_PrintChar sẽ được gọi, hàm void **Exception_sysCall_PrintChar()**; được cài đặt để xử lí exception của system call này. Trong đó hàm sẽ lấy giá trị nguyên của char (trong barngg mã ascii) từ thanh ghi số 4 thông qua câu lệnh: **machine->ReadRegister(4).** Sau đó tiến hành ép lại kiểu char cho giá trị vừa lấy được. Rồi hệ thống sử dụng giải pháp của lớp **SynchConsole** để tiến hành ghi ra console kí tự cần in ra.

System call random

//input: Không

//output: Một số nguyên dương ngẫu nhiên do hệ thống trả về.

Là hàm trả về ngẫu nhiên một số nguyên dương trong khoảng int, trong đó. Ở file randomNum.c người dùng thực hiện gọi hàm **RandomNum();** được khai báo trong file syscall.h của folder userprog.

Trong file syscall.h hàm **RandomNum** sẽ được khai báo, đồng thời sẽ được define một exception **#defined SC_RanNum 21**. Để xử lý exception random number trong file exception.cc. Trong file này hàm random sẽ được cài đặt lại với tên gọi là int **randIntNum()**; Trong này hàm random sẽ được định nghĩa theo 2 hàm random number có sẵn của hệ điều hành đó là **rand()** tương đương ran2 và **srand()** tương đương ran1 trong file sysdep.cc trong folder machine. Cả 2 hàm trên đều trả về một con số ngẫu nhiên trong khoảng giá trị int. Và một yếu tố nữa đó là thanh ghi PC. Để tránh sự trùng lặp hàm tiến hành kết hợp với giá trị của thanh ghi PC. Với mỗi giá trị thanh ghi PC khác nhau mà hàm sẽ trả về một giá trị khác nhau, chia theo 4 case. Nếu giá trị của thanh ghi PC chia hết cho 4 hàm sẽ trả về kết quả là ran1 – ran2. Nếu chia cho 4 dư 1 thì trả về kết quả là ran2+ran1. Nếu dư 2 thì ran1 % ran2. Còn nếu không hàm sẽ trả về kết quả ran1/ran2. Trước khi trả về thì hàm sẽ kiểm tra số đó có âm hay không. Nếu có hàm tiến hành * -1 để đổi về số dương.

System call ReadString

// Tham số: Một chuỗi

//output: Không

Cũng như các syscall khác, system call readstring sẽ được gọi thông qua hàm **ReadString** được khai báo trong file syscall.h trong folder userprog. Trong này system call readString được define là **#define SC_ReadString 19**. Và hàm **void ReadString(char* str);** với tham số truyền vào là một con trỏ kiểu char.

Ý tưởng. Đầu tiên là hàm sẽ lấy địa chỉ ảo của con trỏ thông qua thanh ghi số 4 bằng câu lệnh: **machine->ReadRegister(4)**. Sau đó cài đặt cho độ dài chuỗi tối đa là 128 bytes. Rồi khai báo một con trỏ char tạm là buffer. Sau đó sử dụng hàm **User2System** để lấy thông tin từ người dùng nhập vào. Copy từ không gian người dùng vào không gian vùng nhớ của hệ thống, đồng thời lấy về giá trị buffer (char*). Sau đó sử dụng lớp **SynchConsole** ioCons; Để tiến hành đọc số byte của chuỗi buffer. Tiếp theo hàm khai báo một biến i = 0. Rồi cho chạy vòng lặp tới khi **buffer[i] == '\0'**. Để tính toán độ dài

chuỗi. Sau đó hàm tiếp tục gọi hàm **System2User** truyền vào địa chỉ ảo vừa lấy ở trên, chiều dài chuỗi và chuỗi. Rồi ghi kết quả vào thanh ghi số 2, kết thúc việc đọc chuỗi.

System call print string
// Tham số: Không

//output: Một chuỗi kí tự

Là system call có chức năng in ra chuỗi kí tự, sử dụng thông qua việc gọi hàm **PrintString(char* str)**, được khai báo trong file syscall.h và defined là **#defined SC_PrintString 20**.

Xử lý trong exception. Trong file exception, hàm PrintString được chuyển thành exception system call : SC_PrintString. Trong này, một hàm mới được cài đặt đó là hàm void **Exception_syscall_PrintString().** Ở hàm này, hệ thống nhận một địa chỉ ảo (kiểu int) thông qua việc đọc thanh ghi số 4 bằng câu lệnh: **machine->ReadRegister(4).** Đồng thời chọn số bytes tối đa mà hệ thống đọc được: const int limit = 128. Rồi khai báo một biến buffer (char*) để lấy chuỗi kí tự từ hệ thống thông qua hàm **User2System(địa chỉ ảo, limit);** Sau khi lấy được xong hệ thống tiến hành chạy vòng lặp từ i = 0, cho đến khi **buffer[i] != '\0'**. Mỗi lần lặp hệ thống sẽ sử dụng giải pháp của lớp **SynchConsole** để in từng kí tự ra console cho đến khi chuỗi đạt đủ kí tự, kết thúc chương trình.

COMMAND GỌI CÁC FILE TEST

Read và print string: **"**./userprog/nachos -rs 1023 -x ./test/ReadPrintString"

Read và print char: **"**./userprog/nachos -rs 1023 -x ./test/ReadPrintChar"

Read và print number (kiểu int):

"./userprog/nachos -rs 1023 -x ./test/ReadPrintNum"

Random number: **"**./userprog/nachos -rs 1023 -x ./test/randomNum"

Sort: **"**./userprog/nachos -rs 1023 –x ./test/sortArray"

Print ASCII: **"**./userprog/nachos -rs 1023 –x ./test/PrintASCII"

Help: **"**./userprog/nachos -rs 1023 -x ./test/help"