

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 MÔN HỌC
HỆ ĐIỀU HÀNH

ĐỀ TÀI: Exceptions và các system calls đơn giản

Giảng viên lý thuyết: TS Trần Trung Dũng

Giảng viên hướng dẫn thực hành:

- Thầy Lê Giang Thanh
- Thầy Nguyễn Thanh Quân

Lớp: 20TN

Thành viên thực hiện:

- 20120131 – Nguyễn Văn Lộc
- 20120536 – Võ Trọng Nghĩa
- 20120572 – Nguyễn Kiều Minh Tâm

THÀNH PHỐ HỒ CHÍ MINH, THÁNG 9 10 NĂM 2022

Mục lục

1	Thông tin của nhóm – Phân chia công việc	2
2	Môi trường làm việc	3
3	Hiểu mã chương trình NachOS	3
4	Hiểu thiết kế NachOS	3
5	Tổng quan	3
5.1	Giá trị của các thanh ghi	3
5.2	Các bước viết một system call	3
6	Exceptions và system calls	4
6.1	Viết lại file <i>exception.cc</i>	4
6.1.1	Cài đặt lại các exceptions	4
6.1.2	Cài đặt các syscalls	4
6.2	Tăng program counter	4
6.3	Cài đặt syscall <i>int ReadNum()</i>	4
6.4	Cài đặt syscall <i>void PrintNum(int number)</i>	5
6.5	Cài đặt syscall <i>char ReadChar()</i>	5
6.6	Cài đặt syscall <i>void PrintChar(char character)</i>	5
6.7	Cài đặt syscall <i>void ReadString(char buffer[], int length)</i>	5
6.8	Cài đặt syscall <i>void PrintString(char buffer[])</i>	6
7	Các chương trình test	6
7.1	Chương trình <i>help</i>	6
7.2	Chương trình <i>ascii</i>	6
7.3	Chương trình <i>sort</i>	10

Danh sách hình vẽ

1	Chương trình <i>help</i>	6
2	Chương trình <i>ascii</i>	7
3	Chương trình <i>ascii</i>	8
4	Chương trình <i>ascii</i>	9
5	Chương trình <i>ascii</i>	10
6	Trường hợp n không hợp lệ	10
7	Trường hợp sắp xếp tăng dần	11
8	Trường hợp sắp xếp giảm dần	12

Danh sách bảng

1	Bảng thông tin thành viên	2
---	-------------------------------------	---

1 Thông tin của nhóm – Phân chia công việc

MSSV	Họ và tên	Công việc	Đánh giá mức độ hoàn thành
20120131	Nguyễn Văn Lộc	<ul style="list-style-type: none"> • Xử lý exceptions • Tăng giá trị thanh ghi PC • Viết chương trình ascii • Viết chương trình sort • Viết báo cáo chính cho đồ án 	100%
20120536	Võ Trọng Nghĩa	<ul style="list-style-type: none"> • System call ReadChar • System call PrintChar • System call ReadString • System call PrintString • Tester chính của đồ án 	100%
20120572	Nguyễn Kiều Minh Tâm	<ul style="list-style-type: none"> • System call ReadNum • System call PrintNum • System call RandomNum • Viết chương trình help 	100%

Bảng 1: Bảng thông tin thành viên

2 Môi trường làm việc

NachOS được biên dịch và cài đặt trên Ubuntu 18.04 Image của Docker.

3 Hiểu mã chương trình NachOS

Mã NachOS hiện nay ở mức 1 người dùng bằng chương trình C tại một thời điểm. Chương trình **halt**, yêu cầu hệ điều hành tắt máy, được sử dụng để thử nghiệm NachOS. Dò tìm khi chương trình người dùng nạp, chạy, và gọi một system call.

4 Hiểu thiết kế NachOS

NachOS cung cấp một CPU giả lập, thực tế CPU giả lập này giống hệt CPU thật (MIPS-32 bit chip), nhưng chúng ta không thể chỉ thực thi chương trình như một tiến trình bình thường của UNIX, bởi vì chúng ta muốn kiểm soát có bao nhiêu lệnh được thực hiện trong một đơn vị thời gian, không gian địa chỉ làm việc như thế nào, các interrupt và exception (system calls) được xử lý ra sao.

NachOS cung cấp môi trường giả lập để chạy các chương trình C, chương trình biên dịch thông qua gcc/g++ tạo file object, sau đó chuyển sang định dạng đặc biệt của NachOS nhờ "coff2noff".

5 Tổng quan

5.1 Giá trị của các thanh ghi

- Mã system call được đưa vào thanh ghi r2.
- Tham số thứ 1 được đưa vào thanh ghi r4.
- Tham số thứ 2 được đưa vào thanh ghi r5.
- Tham số thứ 3 được đưa vào thanh ghi r6.
- Tham số thứ 4 được đưa vào thanh ghi r7.
- Kết quả thực hiện của system call được đưa vào thanh ghi r2.

5.2 Các bước viết một system call

Muốn phục vụ người dùng có thể thực thi được các công việc khác nhau, người lập trình hệ điều hành phải xây dựng một bộ các system call đủ để phục vụ các yêu cầu này. System call cũng là hàm xử lý nhưng ở *kernel mode*, khác với một hàm của chương trình người dùng ở *user mode*. Sau đây là các bước viết một system call.

Bước 1: Trong tập tin `/code/userprog/syscall.h`, thêm dòng khai báo một syscall mới.

Bước 2: Thêm các dòng định nghĩa vào tập tin `/code/test/start.c` và `/code/test/start.s`.

Bước 3: Sửa điều kiện **if...** thành **switch... case** trong tập tin *code/userprog/exception.cc*. Trong phần xử lý cho các syscall, tạo tập tin có sử dụng hàm **System2User()** và **User2System()** được hướng dẫn.

Bước 4: Viết chương trình ở mức người dùng để kiểm tra.

Bước 5: Thêm đoạn code vào */code/test/Makefile*.

Bước 6: Biên dịch lại NachOS.

Bước 7: Thực thi chương trình test, nếu chương trình không báo lỗi thì xem như thành công.

6 Exceptions và system calls

6.1 Viết lại file *exception.cc*

6.1.1 Cài đặt lại các exceptions

Danh sách các exceptions nằm ở tập tin *machine.h* trong thư mục */code/machine*.

Trong tập tin */code/userprog/exception.cc*, dùng cấu trúc **switch...case** để cài đặt các exception. Với mỗi exception, sau khi đưa thông báo về exception, ta **Halt** chương trình.

6.1.2 Cài đặt các syscalls

Cấu trúc **switch...case** được sử dụng để tổ chức cài đặt các syscalls theo yêu cầu của đồ án.

6.2 Tăng program counter

Chức năng: Tất cả các syscalls (không phải Halt) sẽ yêu cầu NachOS tăng program counter trước khi syscall trả kết quả về. Nếu không lập trình phần này thì NachOS sẽ rơi vào vòng lặp vô tận, gọi thực hiện syscall này mãi mãi.

Cách thức thực hiện:

- Lấy địa chỉ đang lưu trong thanh ghi PC, ghi vào thanh ghi PrevPC.
- Lấy địa chỉ kế tiếp (tăng lên 4 bytes) lưu vào thanh ghi PC.
- Lấy địa chỉ trong thanh ghi kế tiếp của thanh ghi NextPC lưu vào thanh ghi NextPC.

6.3 Cài đặt syscall *int ReadNum()*

Chức năng: sử dụng lớp *SynchConsoleInput* để đọc một số nguyên do người dùng nhập vào.

Cách thức thực hiện:

- Chương trình chỉ xử lý trường hợp số nhập vào ở hệ thập phân.
- Đọc chuỗi ký tự do người dùng nhập vào.
- Kiểm tra dấu của số được nhập.

- Nếu có ký tự khác (không phải chữ số) thì trả về 0.
- Kiểm tra tràn số, nếu tràn số thì trả ra 0 và dừng.

6.4 Cài đặt syscall *void PrintNum(int number)*

Chức năng: sử dụng lớp `SynchConsoleOutput` để xuất một số nguyên ra màn hình.

Cách thức thực hiện:

- Đọc giá trị *number* từ thanh ghi r4.
- Kiểm tra giá trị *number* so sánh với `INT_MIN`, so sánh với 0 rồi xử lý các trường hợp này.
- Kiểm tra số âm.
- Tìm số lượng chữ số của *number*.
- Lần lượt chèn các ký tự tương ứng với các chữ số của *number* vào string *output*.
- Xuất string ra màn hình bằng cách gọi system call *PrintString*.

6.5 Cài đặt syscall *char ReadChar()*

Chức năng: sử dụng lớp `SynchConsoleInput` để đọc một ký tự do người dùng nhập vào.

Cách thức thực hiện:

- Sử dụng hàm *GetChar()* của lớp *SynchConsoleInput*.
- Chuyển ký tự sang giá trị nguyên 32 bits.
- Lưu giá trị đó vào thanh ghi r2.

6.6 Cài đặt syscall *void PrintChar(char character)*

Chức năng: sử dụng lớp `SynchConsoleOutput` để xuất một ký tự ra màn hình.

Cách thức thực hiện:

- Đọc giá trị *character* từ thanh ghi r4.
- Sử dụng hàm *PutChar()* của lớp *SynchConsoleOutput* để in ký tự ra màn hình.

6.7 Cài đặt syscall *void ReadString(char buffer[], int length)*

Chức năng: đọc một chuỗi ký tự do người dùng nhập vào.

Cách thức thực hiện:

- Đọc địa chỉ của *buffer* từ thanh ghi r4.
- Đọc giá trị *length* từ thanh ghi r5.

- Kiểm tra tính hợp lệ của giá trị *length*.
- Nếu *length* hợp lệ, tạo một chuỗi ký tự có độ dài *length* trên kernel space. Lần lượt đọc các ký tự bằng syscall *ReadChar*, nếu đọc thất bại thì dừng.
- Sử dụng hàm *System2User* để copy vùng nhớ có độ dài *length* từ kernel space sang user space.

6.8 Cài đặt syscall *void PrintString(char buffer[])*

Chức năng: in một chuỗi ký tự ra màn hình.

Cách thức thực hiện:

- Đọc địa chỉ của *buffer* từ thanh ghi r4.
- Bằng hàm *User2System*, copy vùng nhớ từ user space sang kernel space với độ dài tối đa 1024 mỗi lần, rồi gọi syscall *PrintChar* cho từng ký tự.

7 Các chương trình test

7.1 Chương trình *help*

Chương trình *help* gọi syscall *PrintString* để in thông tin về nhóm cũng như giới thiệu vắn tắt về chương trình *ascii* và chương trình *sort*.

```
# ./build.linux/nachos -x ./test/help
Thanh vien trong nhom:
1. 20120131 - Nguyen Van Loc
2. 20120536 - Vo Trong Nghia
3. 20120572 - Nguyen Kieu Minh Tam
-----
Chuong trinh ascii
Chuong trinh in ra cac ky tu ascii voi dinh dang:
Ma ASCII - ky tu
-----
Chuong trinh sort
Nhap vao mot mang so nguyen co n phan tu (1 <= n <= 100),
Nhap 0 de sap xep tang dan, 1 de sap xep giam dan
Machine halting!

Ticks: total 54506, idle 40790, system 13620, user 96
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 408
Paging: faults 0
Network I/O: packets received 0, sent 0
# □
```

Hình 1: Chương trình *help*

7.2 Chương trình *ascii*

Chương trình *ascii* chủ yếu sử dụng *PrintChar*, *PrintString*, và *PrintNum* để in mã ASCII và ký tự tương ứng.

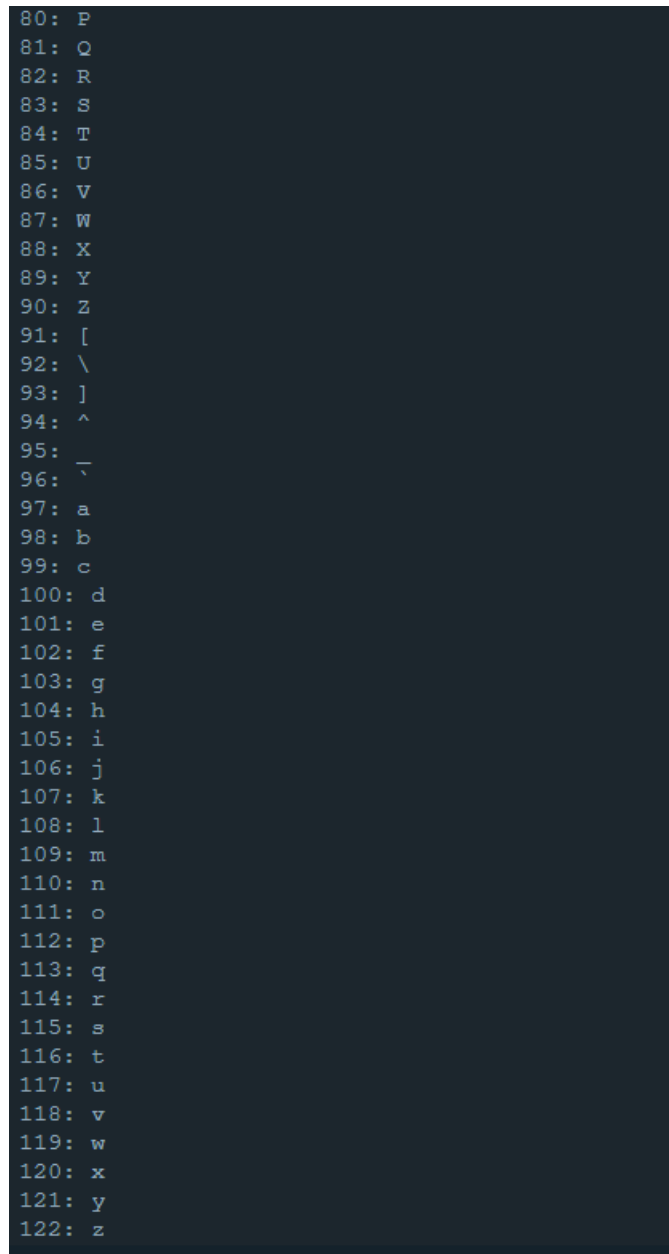
Lưu ý: Chỉ những ký tự in được mới có thể hiện lên màn hình.
Hình ảnh chương trình *ascii*:

```
# ./build.linux/nachos -x ./test/ascii
0:
1:
2:
3:
4:
5:
6:
7:
8:
9:
10:
11:
12:
13:
14:
15:
16:
17:
18:
19:
20:
21:
22:
23:
24:
25:
26:
27:
8:
29:
30:
31:
32:
33: !
34: "
35: #
```

Hình 2: Chương trình *ascii*


```
36: $  
37: %  
38: &  
39: '  
40: (  
41: )  
42: *  
43: +  
44: ,  
45: -  
46: .  
47: /  
48: 0  
49: 1  
50: 2  
51: 3  
52: 4  
53: 5  
54: 6  
55: 7  
56: 8  
57: 9  
58: :  
59: ;  
60: <  
61: =  
62: >  
63: ?  
64: @  
65: A  
66: B  
67: C  
68: D  
69: E  
70: F  
71: G  
72: H  
73: I  
74: J  
75: K  
76: L  
77: M  
78: N  
79: O
```

Hình 3: Chương trình *ascii*



```
80: P
81: Q
82: R
83: S
84: T
85: U
86: V
87: W
88: X
89: Y
90: Z
91: [
92: \
93: ]
94: ^
95: _
96: `
97: a
98: b
99: c
100: d
101: e
102: f
103: g
104: h
105: i
106: j
107: k
108: l
109: m
110: n
111: o
112: p
113: q
114: r
115: s
116: t
117: u
118: v
119: w
120: x
121: y
122: z
```

Hình 4: Chương trình *ascii*

```
123: {
124: |
125: }
126: ~
127:
128:
129:
130:
131:
132:
133:
134:
135:
136:
137:
138:
139:
140:
141:
142:
143:
144: ~
145:
146:
147:
148:
149:
150:
151:
152:
153:
154:
155:
156:
```

Hình 5: Chương trình *ascii*

7.3 Chương trình *sort*

Chương trình sắp xếp tối đa 100 số nguyên tăng dần/giảm dần bằng thuật toán bubble sort.

Chương trình sử dụng các syscall *ReadNum*, *PrintNum*, *PrintString*.

Hình ảnh chương trình *sort*:

```
# ./build.linux/nachos -x ./test/sort
Enter a positive integer n
-1
Please enter n in [1, 100]
105
Please enter n in [1, 100]
□
```

Hình 6: Trường hợp n không hợp lệ

```

# ./build.linux/nachos -x ./test/sort
Enter a positive integer n
9
Enter a[0]
18
Enter a[1]
8
Enter a[2]
19
Enter a[3]
11
Enter a[4]
6
Enter a[5]
1
Enter a[6]
4
Enter a[7]
10
Enter a[8]
2022
Enter 0 for ascending order, 1 for descending order
0
After sorting in ascending order
1 4 6 8 10 11 18 19 2022
Machine halting!

Ticks: total 2101081699, idle 2101069314, system 9210, user 3175
Disk I/O: reads 0, writes 0
Console I/O: reads 29, writes 237
Paging: faults 0
Network I/O: packets received 0, sent 0
```

Hình 7: Trường hợp sắp xếp tăng dần

```
# ./build.linux/nachos -x ./test/sort
Enter a positive integer n
9
Enter a[0]
8
Enter a[1]
18
Enter a[2]
11
Enter a[3]
19
Enter a[4]
27
Enter a[5]
36
Enter a[6]
6
Enter a[7]
1
Enter a[8]
22
Enter 0 for ascending order, 1 for descending order
1
After sorting in descending order
36 27 22 19 18 11 8 6 1
Machine halting!

Ticks: total 1047513085, idle 1047500724, system 9190, user 3171
Disk I/O: reads 0, writes 0
Console I/O: reads 28, writes 237
Paging: faults 0
Network I/O: packets received 0, sent 0
#
```

Hình 8: Trường hợp sắp xếp giảm dần

Tài liệu tham khảo chủ yếu

- [1] *Các tài liệu hướng dẫn thực hành đồ án.*
- [2] Nguyễn Thành Chung. *Các video hướng dẫn lập trình NachOS.* URL: youtube.com/watch?v=t0jtY1C129s&list=PLRgTVtca98hUgCN2_2vzsAAXPiTFbvHp0.