

**UNIVERSITY OF SCIENCE**

Ho Chi Minh City

**INFORMATION OF TECHNOLOGY**

High quality

A

Project

Report on

# **CÀI ĐẶT HỆ ĐIỀU HÀNH NACHOS**

Under the course of

**CTT103 – Operating System**

Semester III

*Submitted by:*

*Class:19CLC9*

- |    |                       |          |
|----|-----------------------|----------|
| 1. | Ho Ngoc Minh Duc      | 19127368 |
| 2. | Nguyen Phuong Vy      | 19127088 |
| 3. | Nguyen Kim Thi To Nga | 19127219 |

**Dr. Lê Viết Long**

---

*Academic year*

*(2019-2023)*

## Table of Contents

<b>I. Mã chương trình NACHOS .....</b>	<b>3</b>
1. NACHOS là gì .....	3
2. Quy trình thực thi của một chương trình trên NACHOS .....	3
3. Các bước cập nhật thanh ghi .....	4
<b>II. Exceptions và System calls .....</b>	<b>5</b>
1. Cài đặt các Exception và cấu trúc chương trình.....	5
2. Tăng program counter .....	6
3. System2User và User2System .....	6
4. SynchConsole .....	7
5. Viết hàm con hỗ trợ SyscallException .....	7
a. Hàm int readInt() và Hàm void printInt().....	7
b. Hàm char readChar() và Hàm void printChar(char character) .....	8
c. Hàm char* readString () và Hàm void printString (): .....	8
6. Cài đặt SyscallException .....	10
7. Viết chương trình trong thư mục test.....	11
a. Chương trình string.c:.....	11
b. Chương trình char.c:.....	12
c. Chương trình int.c: .....	13
d. Chương trình sort.c: .....	16
e. Chương trình ascii.c: .....	17
f. Chương trình help:.....	18
❖ <i>Nguồn</i> .....	18

# I. Mã chương trình NACHOS

## 1. NACHOS là gì

NachOS (hay còn gọi là **Not Another Completely Heuristic Operating System**) là một hệ điều hành đơn giản chạy trên máy ảo, đây là một phần mềm có mã nguồn mở (open-source). NachOS được sử dụng với mục đích tìm hiểu cách một chương trình người dùng được tải lên và thực thi như thế nào và điều gì sẽ xảy ra khi chương trình người dùng chạy system call để yêu cầu một dịch vụ từ hệ điều hành NachOS:

- Máy ảo được giả lập có kiến trúc MIPS với hầu hết các thành phần và chức năng của một máy thật như: thanh ghi, bộ nhớ, bộ xử lý, bộ lệnh, chu kỳ thực thi lệnh, cơ chế ngắt, chu kỳ đồng hồ, ...
- Hệ điều hành NachOS chạy trên máy ảo NachOS hiện là một hệ điều hành đơn chương

Hệ thống NachOS bao gồm: chương trình ứng dụng, máy ảo MIPS, hệ điều hành NachOS và 2 chế độ User MODE và System MODE (Kernel MODE)

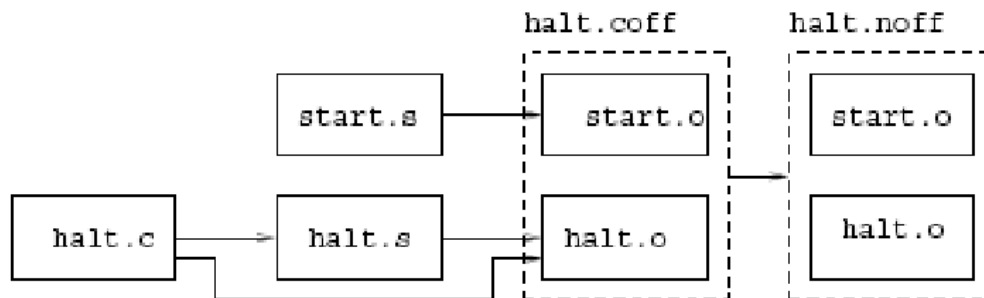
## 2. Quy trình thực thi của một chương trình trên NACHOS

Bởi vì chương trình được biên dịch dựa trên kiến trúc MIPS, do đó các chương trình sẽ không trực tiếp hoạt động trên CPU mà ta chạy NachOS. Thực tế, bởi chúng sử dụng NachOS system calls, do đó các chương trình này được xây dựng để đặc biệt để thực thi dưới nền tảng NachOS.

Ví dụ với chương trình halt.c (code/test/) sẽ được biên dịch như sau:

- Khi biên dịch NachOS, cross-compiler GCC sẽ biên dịch các file có mã nguồn .c thành file hợp ngữ .s, VD: halt.s
- Sau đó tập tin halt.s này sẽ liên kết với file start.s và tạo thành 1 file tổng hợp halt.coff (halt.o và start.o). Đây là file thực thi trên Linux với kiến trúc MIPS.
- Tập tin .coff vừa rồi sẽ được chuyển thành tập tin .noff bằng phần mềm coff2noff. Đây là file thực thi cho NachOS kiến trúc MIPS.

Quy trình trên được thể hiện như sau:



### 3. Các bước cập nhật thanh ghi

Trước khi cài đặt 1 system call, ta phải biết được cách truyền tham số và trả về giá trị của nó. Tham số sẽ được đọc từ các thanh ghi, các giá trị thanh ghi bao gồm:

- R2: Lưu mã syscall đồng thời lưu kết quả trả về của mỗi syscall nếu có.
- R4: Lưu tham số thứ nhất
- R5: Lưu tham số thứ hai
- R6: Lưu tham số thứ ba
- R7: Lưu tham số thứ tư

#### ❖ *System call là gì ?*

Một system call là cách làm cho chương trình thực hiện một tương tác với hệ điều hành. Một chương trình máy tính thực thi một system call khi nó thực hiện một yêu cầu tới nhân hệ điều hành.

## II. Exceptions và System calls

### 1. Cài đặt các Exception và cấu trúc chương trình

Tiến hành cài đặt lại các Exception được liệt kê trong file *nachos-3.4/code/machine/machine.h* bằng cách qua file */userprog/exception.cc* chuyển đoạn mã ban đầu từ cấu trúc **if-else** sang cấu trúc **switch-case** với case là các syscall và exception được gọi.

Khai báo các biến trong **switch case** trong *exception.cc* nên khai báo tách riêng phần gán. Nghĩa là khi khai báo không gán giá trị trả về từ hàm khác. Nếu không sẽ gặp lỗi crosses initialization.

Trường hợp **NoException** sẽ trả về quyền điều khiển về hệ điều hành (return). Tất cả các trường hợp exception còn lại sẽ in ra thông báo lỗi cho người dùng sau đó sẽ Halt hệ thống.

Một phần của xử lý các exception và syscall:

```
switch (which){
    case NoException:
        return;
    case PageFaultException:
        printf("PageFaultException đang diễn ra\n");
        interrupt->Halt();
        break;
    case ReadOnlyException:
        printf("ReadOnlyException đang diễn ra\n");
        interrupt->Halt();
        break;
```

## 2. Tăng program counter

Cài đặt hàm *advancePC* trong *exception.cc* nhằm tăng program counter để nạp lệnh cần thực hiện tiếp theo. Hàm tiến hành lưu giá trị của PC hiện tại (*PCReg*) vào PC trước (*PrevPCReg*), nạp giá trị của PC kế (*NextPCReg*) cho PC hiện tại (*PCReg*) và cuối cùng là nạp giá trị của PC kế tiếp nữa (*NextPCReg+4*) cho PC kế (*NextPCReg*).

Hàm được thực hiện như sau:

```
void advancePC(){
    machine->WriteRegister(PrevPCReg, machine->ReadRegister(PCReg));
    machine->WriteRegister(PCReg, machine->ReadRegister(NextPCReg));
    machine->WriteRegister(NextPCReg, machine->ReadRegister(NextPCReg) + 4);
}
```

## 3. System2User và User2System

Hai hàm được cài đặt trong *exception.cc*

### ❖ Hàm System2User:

```
int System2User(int virtAddr, int len, char* buffer)
```

Hàm sẽ chuyển dữ liệu của 1 chuỗi được lưu trong vùng nhớ hệ điều hành vào vùng nhớ của chương trình người dùng bằng cách ghi từng ký tự một vào vùng nhớ người dùng. Hàm trả về số byte đã được copy, nếu len=0 thì trả về 0 hoặc <0 thì trả về -1.

Các giá trị truyền vào gồm:

- int virtAddr: địa chỉ logic
- int len: độ dài chuỗi
- char\* buffer: chuỗi cần chuyển

### ❖ Hàm User2System

```
char* User2System(int virtAddr, int limit)
```

Hàm dùng để chuyển dữ liệu từ vùng nhớ chương trình của người dùng vào vùng nhớ hệ thống bằng cách lấy từng kí tự ở vùng nhớ chương trình vào một biến buffer sau đó trả về buffer đó.

Trong đó:

- int virtAddr: địa chỉ logic
- int limit: độ dài chuỗi

## 4. SynchConsole

- Dùng để nhập và xuất từ màn hình Console bằng hai hàm chính Read và Write

- Cài đặt:

+ Chép 2 files synchcons.h và synchcons.cc vào thư mục/code/threads.

+ Vào file Makefile.common (trong nachos-3.4/code):

Đầu dòng USERPROG\_H = thêm vào ../threads/synchcons.h

Cuối đoạn bắt đầu bằng USERPROG\_C = .. thêm vào ../threads/synchcons.cc

Và cuối đoạn bắt đầu bằng USERPROG\_O = .. thêm vào synchcons.o

+ Sau đó:

- Khai báo gSynchConsole trong file */code/threads/system.h*
- Sau đó create object SynchConsole trong ifdef USER\_PROGRAM trong *system.cc*
- Và delete gSynchConsole trong hàm clean up

## 5. Viết hàm con hỗ trợ SyscallException

### a. Hàm int readInt() và Hàm void printInt()

#### ❖ Hàm void readInt():

- Hàm để đọc số nguyên do người dùng nhập vào.
- Đọc chuỗi vào buffer từ màn hình Console qua hàm gSynchConsole->read().
- Loại bỏ khoảng trắng trước chuỗi số nếu có

- Kiểm tra từng kí tự có phải là chữ số hay không, nếu không phải thì trả về số 0 và thông báo “Invalid Integer”
- Nếu kí tự thuộc từ ‘0’ đến ‘9’ thì chuyển từng kí tự về số với hàng giá trị tương ứng của nó. Cuối cùng hàm sẽ trả về số nguyên tương ứng
- Nếu số nhập vào lớn hơn giá trị tối đa kiểu INT thì thông báo “Overflow” và trả về 0.

❖ **Hàm void printInt():**

- Hàm sẽ xuất 1 số nguyên ra màn hình.
- Lấy giá trị của số nguyên từ Register số 4
- Nếu số chỉ có 1 ký tự thuộc từ 0 đến 9, chuyển số về dạng char và gọi hàm write của gSynchConsole để in ra
- Nếu số có 2 ký tự trở lên (số âm hoặc số dương lớn hơn 9), cần chuyển số đó về chuỗi char với dấu “-” ở đầu nếu có và gọi hàm write của gSynchConsole như trên để in ra màn hình

**b. Hàm char readChar() và Hàm void printChar(char character)**

- Tương tự như hàm readInt và printInt, 2 hàm này sẽ lần lượt đọc 1 ký tự từ màn hình do người dùng nhập vào và in ra 1 ký tự qua màn hình
- Hàm xem xét có phải kí tự hợp lệ trước khi trả về
- Nếu không phải kí tự hợp lệ, hoặc không nhập gì mà nhấn enter, sẽ trả về kí tự rỗng ‘\0’ và thông báo “Invalid Char”

**c. Hàm char\* readString () và Hàm void printString ():**

❖ ***char\* readString():***

- Lấy địa chỉ và độ dài chuỗi ở register 4 và 5



- Đọc string từ màn hình console qua hàm `gSynchConsole->read()`, đồng thời lấy được số kí tự đọc được (bằng với số bytes đọc được vì 1 kí tự ascii chiếm 1 byte)

- Cuối cùng chuyển dữ liệu từ System vào User qua hàm ***System2User*** theo số lượng kí tự đọc được. Nếu số lượng byte đọc được qua lớp `gSynchConsole` bằng 0 hoặc -1 thì sẽ thông báo “Console Error”.

❖ ***void printString():***

- Lấy địa chỉ lưu chuỗi ở register 4, sau đó chuyển dữ liệu từ user space qua kernel space qua hàm ***User2System*** rồi in ra màn hình console.

## 6. Cài đặt SyscallException

-Sau khi viết hàm con thì chúng ta sẽ gọi hàm con này bên trong switch case của Syscall Exception, đồng thời cấu hình thêm 1 số file khác

- Ví dụ cách tạo 1 syscall từ hàm con readChar() như sau, các hàm khác tương tự:

- Vào **exception.cc**:
  - Viết hàm con char readChar()
  - Thêm case **SC\_ReadChar** vào trong switch case và gọi hàm con readChar() thực hiện đọc kí tự, lấy được kết quả trả về và xử lý
- Vào **syscall.h** thêm:
  - #define SC\_ReadChar 14 (mỗi hàm sẽ có số riêng biệt)
  - Khai báo hàm char ReadChar(); (đây là hàm user sẽ gọi sử dụng khi viết chương trình để test)
- Vào **start.c start.s** thêm thông tin hàm:

```
.globl ReadChar
.ent    ReadChar

ReadChar:
    addiu $2,$0, SC_ReadChar
    syscall
    j      $31
.end ReadChar
```
- Viết chương trình **char.c** trong thư mục test
- Vào **makefile** trong thư mục **test**:
  - Thêm tên file: “**char**” vào dòng all
  - Thêm vào gần cuối file các dòng sau để biên dịch chương trình:

```
char.o: char.c
    $(CC) $(CFLAGS) -c char.c

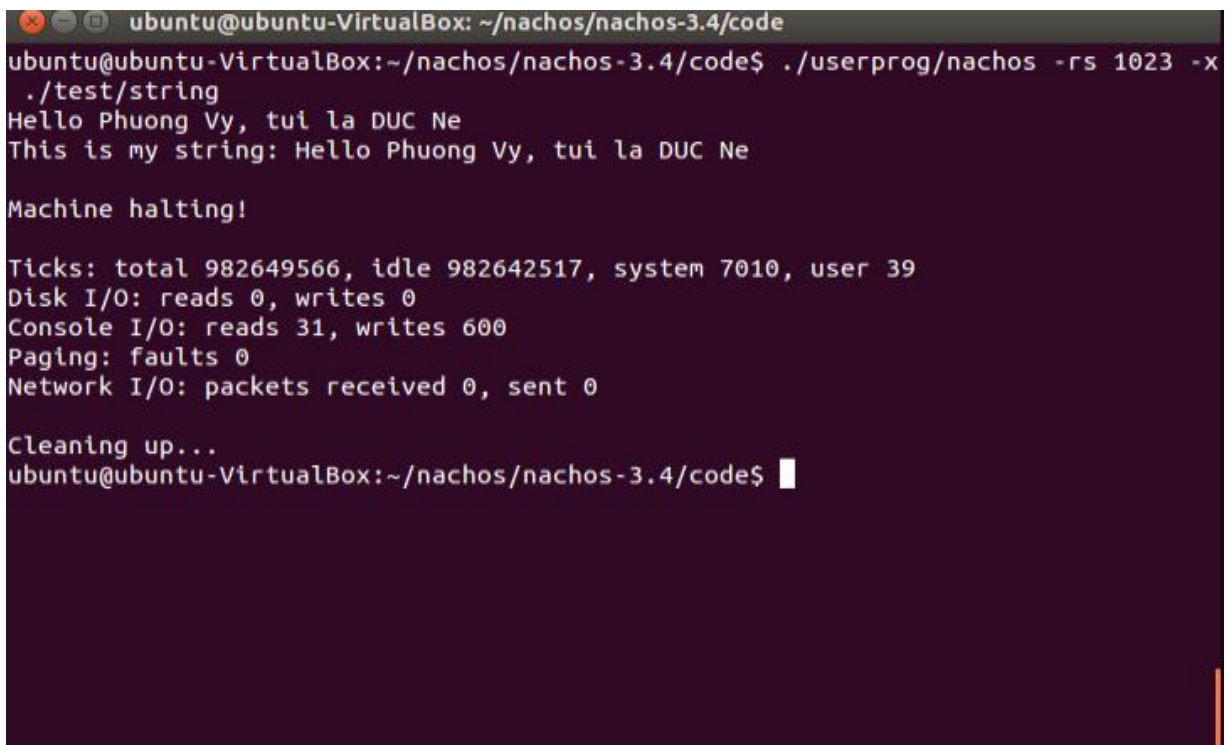
char: char.o start.o
    $(LD) $(LDFLAGS) start.o char.o -o char.coff
    ../bin/coff2noff char.coff char
```

## 7. Viết chương trình trong thư mục test

### a. Chương trình string.c:

- Sử dụng hàm ReadString gọi từ syscall.h để đọc chuỗi mà User nhập vào
- Chương trình cho phép User nhập vào chuỗi có độ dài tối đa cài đặt sẵn là 200 ký tự
- Sử dụng hàm PrintString gọi từ syscall.h in ra màn hình chuỗi vừa đọc được

#### ❖ Demo nhập và in ra chuỗi:



```
ubuntu@ubuntu-VirtualBox: ~/nachos/nachos-3.4/code
ubuntu@ubuntu-VirtualBox:~/nachos/nachos-3.4/code$ ./userprog/nachos -rs 1023 -x
./test/string
Hello Phuong Vy, tui la DUC Ne
This is my string: Hello Phuong Vy, tui la DUC Ne

Machine halting!

Ticks: total 982649566, idle 982642517, system 7010, user 39
Disk I/O: reads 0, writes 0
Console I/O: reads 31, writes 600
Paging: faults 0
Network I/O: packets received 0, sent 0

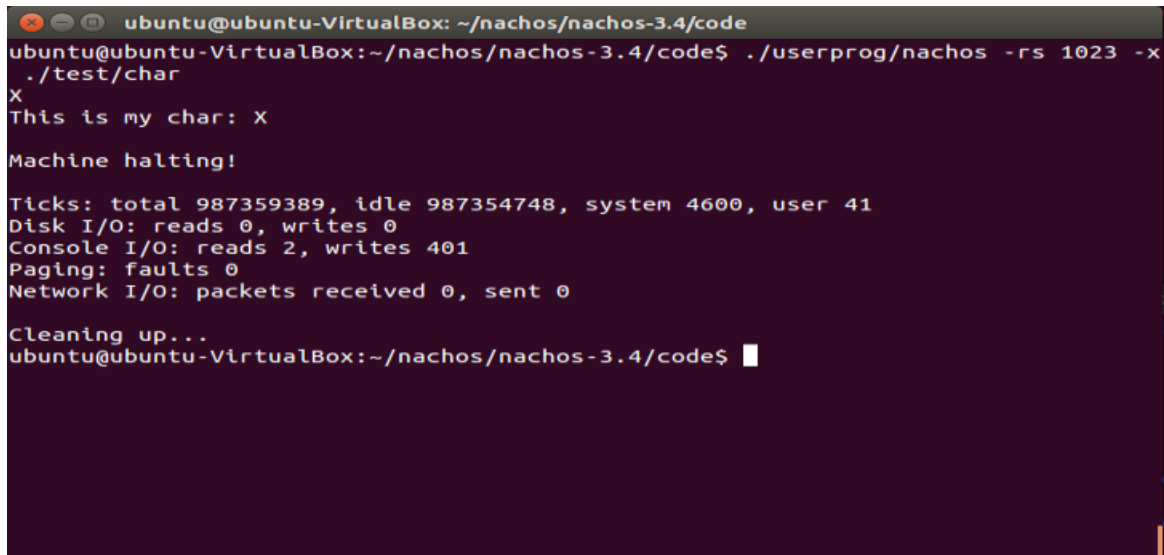
Cleaning up...
ubuntu@ubuntu-VirtualBox:~/nachos/nachos-3.4/code$
```

## b. Chương trình char.c:

- Sử dụng hàm ReadChar gọi từ syscall.h đọc 1 kí tự mà User nhập vào
- Nếu không nhập gì thì hàm ReadChar sẽ báo “Console Error”, hoặc “Invalid Char” nếu có nhiều hơn 1 kí tự

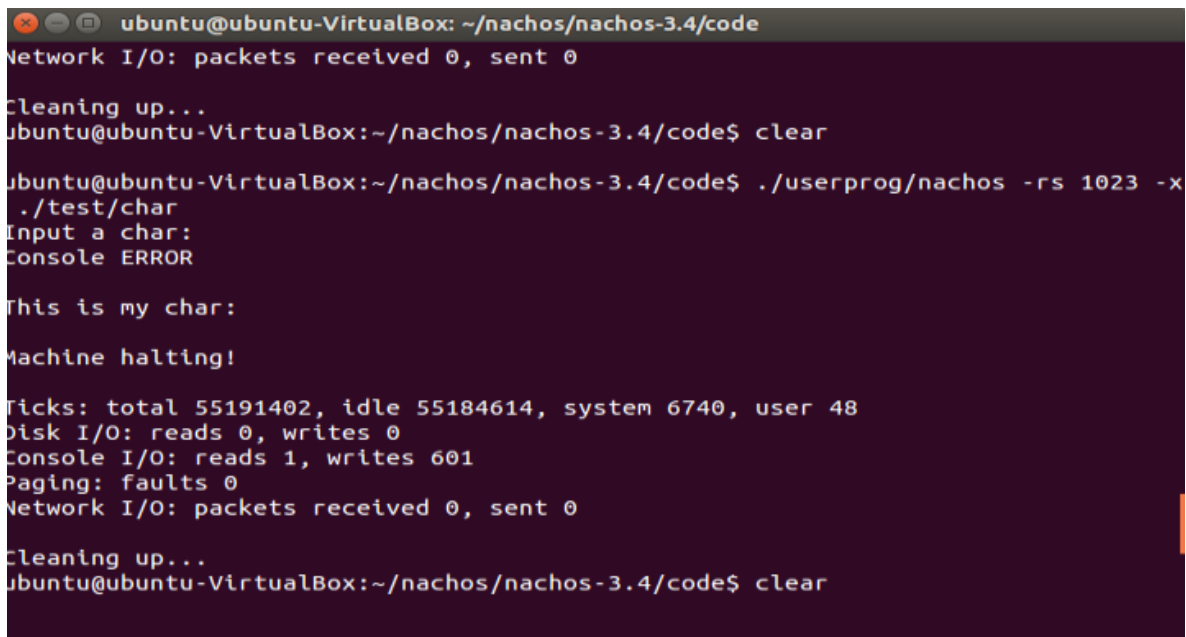
### ❖ Demo chương trình:

*Nhập và in ra 1 kí tự:*



```
ubuntu@ubuntu-VirtualBox: ~/nachos/nachos-3.4/code
ubuntu@ubuntu-VirtualBox:~/nachos/nachos-3.4/code$ ./userprog/nachos -rs 1023 -x
./test/char
X
This is my char: X
Machine halting!
Ticks: total 987359389, idle 987354748, system 4600, user 41
Disk I/O: reads 0, writes 0
Console I/O: reads 2, writes 401
Paging: faults 0
Network I/O: packets received 0, sent 0
Cleaning up...
ubuntu@ubuntu-VirtualBox:~/nachos/nachos-3.4/code$
```

*Nhấn enter và không nhập gì:*



```
ubuntu@ubuntu-VirtualBox: ~/nachos/nachos-3.4/code
Network I/O: packets received 0, sent 0
Cleaning up...
ubuntu@ubuntu-VirtualBox:~/nachos/nachos-3.4/code$ clear
ubuntu@ubuntu-VirtualBox:~/nachos/nachos-3.4/code$ ./userprog/nachos -rs 1023 -x
./test/char
Input a char:
Console ERROR
This is my char:
Machine halting!
Ticks: total 55191402, idle 55184614, system 6740, user 48
Disk I/O: reads 0, writes 0
Console I/O: reads 1, writes 601
Paging: faults 0
Network I/O: packets received 0, sent 0
Cleaning up...
ubuntu@ubuntu-VirtualBox:~/nachos/nachos-3.4/code$ clear
```

### c. Chương trình int.c:

- Sử dụng hàm ReadInt gọi từ syscall.h đọc vào giá trị User nhập
- Đồng thời hàm ReadInt cũng in ra thông báo “Invalid Integer” nếu như không phải là số hợp lệ hoặc “Integer Overflow” nếu là số lớn hơn kiểu số kiểu INT
- Sử dụng hàm PrintInt gọi từ syscall.h để in ra số vừa đọc được

#### ❖ Demo chương trình:

##### *Nhập số nguyên dương:*

```
ubuntu@ubuntu-VirtualBox: ~/nachos/nachos-3.4/code
ubuntu@ubuntu-VirtualBox:~/nachos/nachos-3.4/code$ ./userprog/nachos -rs 1023 -x
./test/int
1234567
Your Number: 1234567

Machine halting!

Ticks: total 902794678, idle 902789888, system 4750, user 40
Disk I/O: reads 0, writes 0
Console I/O: reads 8, writes 407
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
ubuntu@ubuntu-VirtualBox:~/nachos/nachos-3.4/code$
```

##### *Nhập có khoảng trắng:*

```
ubuntu@ubuntu-VirtualBox: ~/nachos/nachos-3.4/code
ubuntu@ubuntu-VirtualBox:~/nachos/nachos-3.4/code$ ./userprog/nachos -rs 1023 -x
./test/int
987654321
Your Number: 987654321

Machine halting!

Ticks: total 614816038, idle 614811288, system 4710, user 40
Disk I/O: reads 0, writes 0
Console I/O: reads 26, writes 409
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
ubuntu@ubuntu-VirtualBox:~/nachos/nachos-3.4/code$
```

*Nhập số âm:*

```
ubuntu@ubuntu-VirtualBox: ~/nachos/nachos-3.4/code
ubuntu@ubuntu-VirtualBox:~/nachos/nachos-3.4/code$ ./userprog/nachos -rs 1023 -x
./test/int
-12345567
Your Number: -12345567

Machine halting!

Ticks: total 571019558, idle 571014768, system 4750, user 40
Disk I/O: reads 0, writes 0
Console I/O: reads 10, writes 409
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
ubuntu@ubuntu-VirtualBox:~/nachos/nachos-3.4/code$
```

*Nhập số lớn hơn kiểu dữ liệu int:*

```
ubuntu@ubuntu-VirtualBox: ~/nachos/nachos-3.4/code
ubuntu@ubuntu-VirtualBox:~/nachos/nachos-3.4/code$ ./userprog/nachos -rs 1023 -
./test/int
999999999999999
Integer Overflow

Your Number: 0

Machine halting!

Ticks: total 170171988, idle 170167228, system 4720, user 40
Disk I/O: reads 0, writes 0
Console I/O: reads 14, writes 401
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
ubuntu@ubuntu-VirtualBox:~/nachos/nachos-3.4/code$
```

*Nhập không phải số nguyên:*

```
ubuntu@ubuntu-VirtualBox: ~/nachos/nachos-3.4/code
ubuntu@ubuntu-VirtualBox:~/nachos/nachos-3.4/code$ ./userprog/nachos -rs 1023 -x
./test/int
346453abcd
Invalid Integer

Your Number: 0

Machine halting!

Ticks: total 617597478, idle 617592838, system 4600, user 40
Disk I/O: reads 0, writes 0
Console I/O: reads 11, writes 401
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
ubuntu@ubuntu-VirtualBox:~/nachos/nachos-3.4/code$
```

*Nhấn enter và không nhập gì:*

```
ubuntu@ubuntu-VirtualBox: ~/nachos/nachos-3.4/code
ubuntu@ubuntu-VirtualBox:~/nachos/nachos-3.4/code$ ./userprog/nachos -rs 1023 -x
./test/int
Input your number:
Your Number: 0

Machine halting!

Ticks: total 48740802, idle 48733914, system 6840, user 48
Disk I/O: reads 0, writes 0
Console I/O: reads 1, writes 601
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
ubuntu@ubuntu-VirtualBox:~/nachos/nachos-3.4/code$
```

#### d. Chương trình sort.c:

- Yêu cầu người dùng nhập vào một mảng **n** số nguyên với **n** là số do người dùng nhập vào ( $n \leq 100$ )
- Nếu User nhập vào  $n > 100$  thì gọi hàm Halt() và kết thúc chương trình
- Chương trình sử dụng thuật toán bubble sort để sắp xếp mảng và in ra mảng đã được sắp xếp.
- Chương trình sử dụng 2 hàm ReadInt, PrintInt và PrintString từ syscall.h, kết hợp với vòng lặp for để đọc và in mảng số nguyên.

#### ❖ Demo chương trình:

*Nhập mảng có 10 chữ số:*

```
ubuntu@ubuntu-VirtualBox: ~/nachos/nachos-3.4/code
./test/sort
Input N <= 100: 10
A[0]: 9
A[1]: -3
A[2]: 13
A[3]: 10
A[4]: 29
A[5]: 99
A[6]: 27
A[7]: 18
A[8]: -72
A[9]: 1
Sorted Array: -72 -3 1 9 10 13 18 27 29 99

Machine halting!

Ticks: total 2016405213, idle 2016348750, system 53100, user 3363
Disk I/O: reads 0, writes 0
Console I/O: reads 32, writes 4639
Paging: faults 0
Network I/O: packets received 0, sent 0

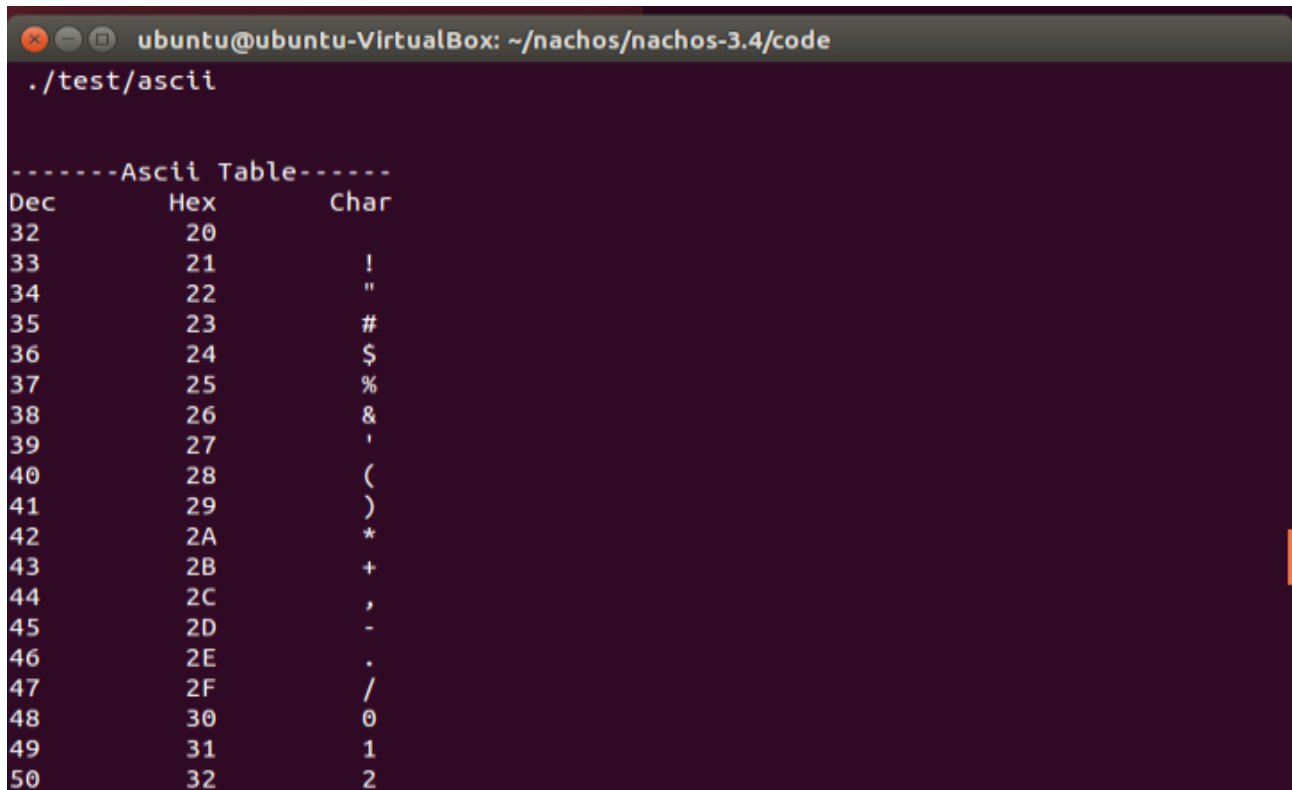
Cleaning up...
ubuntu@ubuntu-VirtualBox:~/nachos/nachos-3.4/code$
```



### e. Chương trình `ascii.c`:

- In ra màn hình bảng ascii từ ký tự 32 đến ký tự 127 với 3 cột: decimal – hexadecimal – char
- Trong chương trình có viết thêm hàm `intToHex` để in ra số hexadecimal từ kiểu `int`
- Sử dụng các hàm `PrintChar`, `PrintInt`, `PrintString` gọi từ `syscall.h` để in bảng

#### ❖ Demo chương trình:



```
ubuntu@ubuntu-VirtualBox: ~/nachos/nachos-3.4/code
./test/ascii

-----Ascii Table-----
Dec      Hex      Char
32       20
33       21      !
34       22      "
35       23      #
36       24      $
37       25      %
38       26      &
39       27      '
40       28      (
41       29      )
42       2A      *
43       2B      +
44       2C      ,
45       2D      -
46       2E      .
47       2F      /
48       30      0
49       31      1
50       32      2
```

## f. Chương trình help:

Sử dụng hàm PrintString cũng gọi từ syscall.h để in ra thông tin của nhóm cũng như mô tả tổng quan về hai hàm sort và ascii.

### ❖ Demo chương trình:

```
ubuntu@ubuntu-VirtualBox: ~/nachos/nachos-3.4/code
ubuntu@ubuntu-VirtualBox:~/nachos/nachos-3.4/code$ ./userprog/nachos -rs 1023 -x ./test/help
      DUC
      19127368
      .....VY.....
      ...19127088...
      .....NGA.....
      .....19127219.....
      |||
      |||
      |||
DO AN 2 HE DIEU HANH

SORT:
      -Cho phép người dùng nhập vào một mảng n số nguyên với n là số do người dùng nhập vào(n<=100)
      -Nếu nhập không hợp lệ thì mặc định là 0
      -In ra màn hình mảng tang dần

Ascii:
      -In ra bảng mã ascii từ kí tự 32 đến 127 theo cấu trúc:  dec - hex - char

Machine halting!

Ticks: total 66744, idle 60000, system 6700, user 44
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 600
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
ubuntu@ubuntu-VirtualBox:~/nachos/nachos-3.4/code$
```

### ❖ Nguồn

Tài liệu NachOS và video của thầy