

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN – ĐHQG HCM
KHOA CÔNG NGHỆ THÔNG TIN

MÔN HỆ ĐIỀU HÀNH
BÁO CÁO ĐỒ ÁN 02



Mục lục

THÔNG TIN THÀNH VIÊN VÀ PHÂN CÔNG	3
BÁO CÁO ĐỒ ÁN 02.....	4
PHẦN 1: HIỂU MÃ CHƯƠNG TRÌNH NACHOS	4
PHẦN 2: HIỂU THIẾT KẾ.....	8
PHẦN 3: EXCEPTIONS VÀ SYSTEM CALLS	10
PHẦN 4: DEMO	20
TÀI LIỆU THAM KHẢO.....	27

THÔNG TIN THÀNH VIÊN VÀ PHÂN CÔNG

MSSV	HỌ TÊN	PHÂN CÔNG
19120257	Phạm Anh Khoa	Thực hiện câu h, i
19120301	Võ Thành Nam	Thực hiện câu a, b, c, d, e
19120331	Phạm Lưu Mỹ Phúc	Thực hiện báo cáo
19120389	Tô Gia Thuận	Thực hiện câu f, g
19120454	Bùi Quang Bảo	Thực hiện câu j, h, i (3 câu cuối)

BÁO CÁO ĐỒ ÁN 02

PHẦN 1: HIỂU MÃ CHƯƠNG TRÌNH NACHOS

I. KHÁI QUÁT VỀ NACHOS:

- Nachos (Not another completely heuristic operating system) là phần mềm mã nguồn mở phục vụ cho việc giảng dạy và tìm hiểu về hệ điều hành. Nachos giả lập một máy tính ảo và một số thành phần của hệ điều hành như một MIPS CPU, một ổ cứng, một bộ điều khiển ngắt, bộ đếm thời gian và các thành phần khác.
- Máy ảo Nachos được giả lập theo 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,...

II. CÀI ĐẶT TỔNG QUAN:

1. Cài đặt Nachos:

Cài đặt gói Nachos trên Moodle do giáo viên cung cấp và các thành phần sau:

- Môi trường cài đặt: Ubuntu 14.4 (32 bit)
- Trình biên dịch: gcc /g++
- Trong thư mục “./nachos/cross-compiler/deystation-ultrix/bin/”. Thiết lập thuộc tính “execute” ở mọi người đối với tất cả các file trong thư mục này.
- Trong Makefile trong “./code/test” sửa “MAKE=gmake” thành “MAKE=make” và lưu lại.
- Trong thư mục code mở terminal gõ “make all” để cài đặt.
- Chạy thử chương trình trên Nachos bằng lệnh: “**nachos -rs 1234 -x ../test/halt**” để kiểm tra. Cài đặt thành công khi chương trình hiển thị như sau:

```
Assuming the program completed.
Machine halting!

Ticks: total 15, idle 5, system 10, user 0
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 0
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
```

2. Cài đặt lớp SynchConsole:

Trong file `system.h` và `system.cc` (`../code/threads/`):

- Thêm 2 tập tin: `synchcons.h` và `synchcons.cc` vào thư mục.
- Mở file `Makefile.common` (`nachos-3.4/code`). Thêm vào cuối đoạn bắt đầu bằng:
 - **USERPROG_H** nội dung `../threads/synchcons.h`
 - **USERPROG_C** nội dung `../threads/synchcons.cc`
 - **USERPROG_O** nội dung `synchcons.o`

Chức năng: Hỗ trợ nhập xuất từ màn hình console. Để nhập và xuất dữ liệu từ màn hình console ta phải sử dụng lớp **SynchConsole** được khởi tạo qua biến toàn cục `gSynchConsole`.

Gồm 2 hàm chính:

- `int Read(char *into, int numBytes)`: Cho phép nhập một chuỗi từ màn hình console và lưu biến thuộc vùng nhớ hệ điều hành Nachos.
- `int Write(char *from, int numBytes)`: Cho phép xuất một chuỗi từ biến thuộc vùng nhớ hệ điều hành Nachos ra màn hình consol.

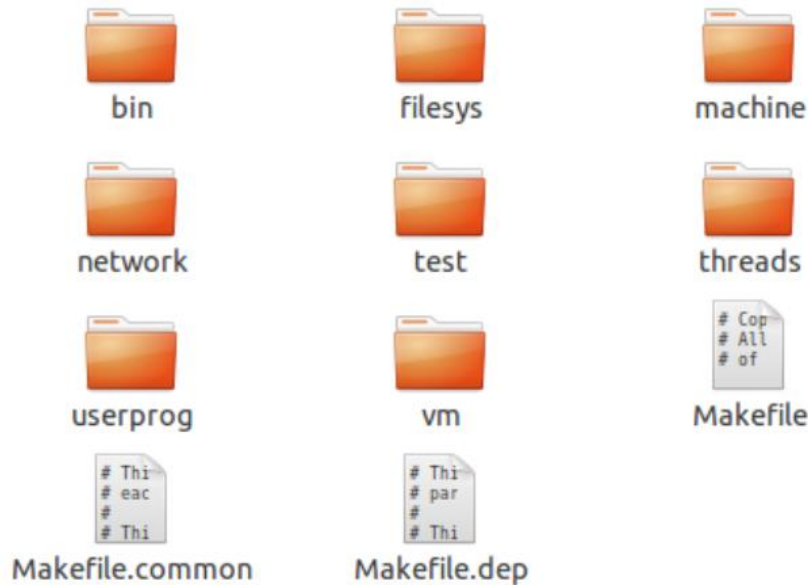
III. TÌM HIỂU THƯ MỤC TRONG FOLDER NACHOS:



Thành phần trong folder Nachos

Tìm hiểu thư mục code (thư mục chính):

Nachos có 2 thành phần chính là hệ điều hành và máy ảo (the machine simulator) được chứa trong thư mục `code`. Trong đó, máy ảo nằm trong thư mục `machine`. Trong đồ án này, không cần thực hiện thay đổi gì trên folder `machine` mà chủ yếu chỉ làm việc với folder `test`, `userprog` và `threads`. Tiếp theo, ta tìm hiểu tập tin trong các thư mục mà ta sẽ thao tác hoặc có ảnh hưởng trong quá trình thực hiện.



Thành phần trong thư mục code

1. Các tập tin chính trong thư mục **machine**:

mipssim.cc	Mô phỏng tập lệnh của MIPS R2/3000 processor
mipssim.h	Định nghĩa bảng opcodes và decoding
machine.*	Mô phỏng các thành phần của máy tính khi thực thi chương trình người dùng: bộ nhớ chính, thanh ghi,...
console.*	Mô phỏng thiết bị đầu cuối sử dụng UNIX files. Một thiết bị có các đặc tính: đơn vị dữ liệu theo byte, đọc và ghi các bytes cùng một thời điểm, các bytes đến bất đồng bộ
translate.*	Mỗi địa chỉ ảo được giả sử giống như địa chỉ vật lý, điều này giới hạn chúng ta chỉ chạy 1 chương trình tại một thời điểm.
timer.*	Định nghĩa lớp Timer và mô phỏng bộ đếm thời gian phần cứng bằng cách tạo ra CPU ngắt mỗi TimerTicks
interrupt.*	Duy trì các cấu trúc để biểu diễn ngắt, quản lý thời gian mô phỏng và xác định mức ngắt

2. Các tập tin chính trong thư mục **fileys**:

fileys.h	Định nghĩa các lớp để biểu diễn file hệ thống
openfile.h	Định nghĩa các hàm trong hệ thống file Nachos

3. Các tập tin chính trong thư mục **userprog**:

protest.cc	Kiểm tra các thủ tục để chạy chương trình người dùng
syscall.h	System call interface: các thủ tục ở kernel mà chương trình người dùng có thể gọi
exception.cc	Xử lý system call và các exception khác ở mức user, ví dụ như lỗi trang,...
bitmap.*	Các hàm xử lý cho lớp bitmap (hữu ích cho việc lưu vết các ô nhớ vật lý)

4. Tập tin chính trong thư mục **bin**:

Coff.h, noff.h, coff2noff.c: COFF là định dạng tệp đối tượng tiêu chuẩn được GCC sử dụng. NOFF là định dạng tệp đối tượng tiêu chuẩn được Nachos sử dụng. coff2noff biến các tệp coff thành định dạng noff.

5. ../test/* các chương trình C sẽ được biên dịch theo MIPS và chạy trong Nachos.

Test/start.s cài đặt mã hợp ngữ sơ khai cho tất cả các lệnh gọi hệ thống trong userprog/syscall.h và có thể thay thế cho C trong đề án này.

PHẦN 2: HIỂU THIẾT KẾ

Mỗi chương trình trong hệ thống bao gồm thông tin cục bộ của nó: program counters, registers, stack pointers, và file system handler. Mặc dù user program truy cập các thông tin cục bộ của nó nhưng HĐH điều khiển các truy cập này, HĐH đảm bảo các yêu cầu từ user program tới kernel không làm cho HĐH sụp đổ. Việc chuyển quyền điều khiển từ user mode thành system mode được thực hiện thông qua system call, software interrupt.

Trước khi gọi một lệnh trong hệ thống thì các tham số cần thiết phải được nạp vào các thanh ghi của CPU. Để chuyển một biến mang giá trị, tiến trình chỉ việc ghi các giá trị vào thanh ghi. Để chuyển một biến tham chiếu thì giá trị lưu trong thanh ghi được xem như là “user space pointer”. Ta cần chuyển nội dung từ user space vào kernel để có thể xử lý dữ liệu này. Khi trả thông tin từ system về user space các giá trị phải đặt trong các thanh ghi của CPU.

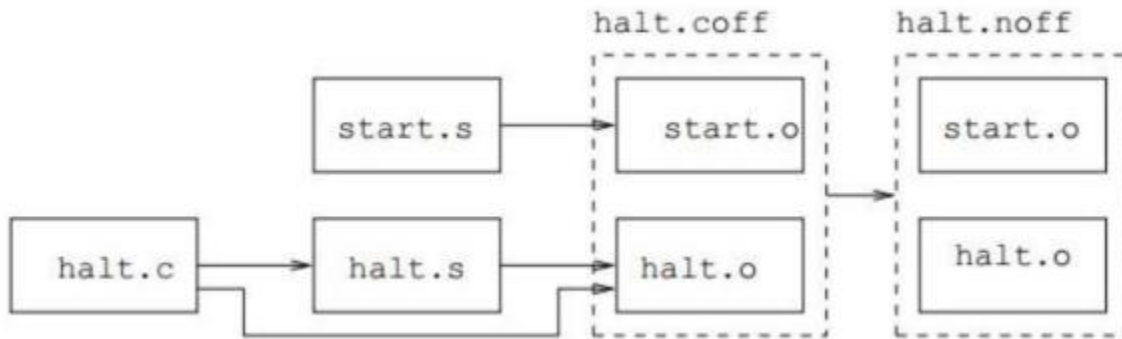
Quá trình thực thi một chương trình trên Nachos

Phân tích quá trình thực thi chương trình halt.c (/code/test/) để biết được cách thức giao tiếp giữa HĐH Nachos và user program. Chương trình này gọi một hàm duy nhất là Halt() để tắt máy ảo Nachos.

- Để thực thi một chương trình, trước tiên nó cần được biên dịch. Với Nachos, tất cả các chương trình trong thư mục /code/test/ đều được biên dịch khi biên dịch Nachos. Ta cũng có thể biên dịch chương trình mà không cần biên dịch lại Nachos như sau:


```
% ...../cross-compiler/decstation-ultrix/bin/gcc -
I ...../nachos/code/userprog-I...../nachos/code/threads -
o ...../test/halt .... /test/halt.c
```
- Quá trình biên dịch chia làm 3 giai đoạn:
 - Chương trình halt.c được biên dịch bởi cross-compiler gcc thành tập tin halt.s là mã hợp ngữ chạy trên kiến trúc MIPS
 - Tập tin halt.s này sẽ được liên kết với tập tin start.s để tạo thành tập tin halt.coff, là định dạng thực thi cho kiến trúc máy MIPS.
 - Tập tin halt.coff được chuyển thành tập tin halt.noff là định dạng thực thi trên hệ điều hành Nachos cho kiến trúc máy MIPS, sử dụng tiện ích coff2noff (trong thư mục /code/bin/)

Quá trình này được miêu tả bằng hình dưới đây:



Sau khi biên dịch thành công, ta có tập tin halt.noff là chương trình thực thi trên Nachos. Để thực hiện chương trình này, gõ lệnh:

```
% ./userprog/nachos -rs 1023 -x ./test/halt
```

Khi thực thi một chương trình trên Nachos thì start luôn được thực thi trước và quá trình thực thi của chương trình sẽ được tiến hành. Khi chương trình vào file machine.cc nó sẽ gọi ExceptionHandler để sang kernel mode (exception.cc). Tại đây, chương trình xác định which được truyền vào là loại exception nào và thực hiện các thao tác được định nghĩa trong hàm.

```

void
ExceptionHandler(ExceptionType which)
{
    int type = machine->ReadRegister(2);

    if ((which == SyscallException) && (type == SC_Halt)) {
        DEBUG('a', "Shutdown, initiated by user program.\n");
        interrupt->Halt();
    } else {
        printf("Unexpected user mode exception %d %d\n", which, type);
        ASSERT(FALSE);
    }
}
  
```

Muốn người dùng có thể thực thi được các công việc khác nhau thì người phát triển hệ điều hành phải xây dựng **bộ system call** đầy đủ để phục vụ các yêu cầu đó. System call cũng là một hàm xử lý nhưng ở kernel mode khác với một hàm của chương trình người dùng ở user mode và không thể truyền tham số trực tiếp cho system call.

PHẦN 3: EXCEPTIONS VÀ SYSTEM CALLS

a. Viết lại file `exception.cc` để xử lý tất cả các exceptions được liệt kê trong `machine/machine.h`.

- Hầu hết các exception trong này là run-time errors, khi các exception này xảy ra thì user program không thể được phục hồi. Trường hợp đặc biệt duy nhất là no exception sẽ trả quyền điều khiển về HĐH, còn syscall exceptions sẽ được xử lý bởi các hàm chúng ta viết cho user system calls. Với tất cả exceptions khác, HĐH hiển thị ra một thông báo lỗi và Halt hệ thống.

- Thực hiện sửa file `userprog/exception.cc`.

- Với NoException, thực hiện return.

```
case NoException:
    return;
```

- Với exception, in loại exception và gọi `ASSERT(FALSE)`. Hàm `ASSERT()` thực hiện in thông báo lỗi và “dump core”.

```
case IllegalInstrException:
    printf("Illegal Instr Exception.\n");
    ASSERT(FALSE);
    break;
```

- Riêng đối với loại exception là SyscallException, ta viết phần xử lý cho tất cả các system call được liệt kê trong `userprog/syscall.h`

b. Viết lại cấu trúc điều khiển của chương trình để nhận các Nachos system calls. Kiểm tra cấu trúc mới bằng cách dùng system call Halt để kiểm tra tính đúng đắn của cấu trúc mới.

- Trong tập tin `/code/userprog/syscall.h` ta định nghĩa con số cụ thể trong kernel space thành một macro

```
#define SC_Halt 0
void Halt();
```

Thêm một đoạn hợp ngữ trong tập tin `/code/test/start.c` và `/code/test/start.s` để ghi mã của syscall vừa tạo vào thanh ghi 2.

```

Halt:
    .globl Halt
    .ent    Halt
    addiu $2,$0,SC_Halt
    syscall
    j      $31
    .end Halt

```

Trong tập tin `/code/userprog/exception.cc` viết **switch case** để xử lý mỗi exception.

```

case SyscallException:{
    switch(type){
        case SC_Halt:
            DEBUG('a',"Shutdown, initiated by user program.\n");
            interrupt->Halt();
            break;
        case SC_ReadInt:
            EH_ReadInt();
            IncProgCounter();
            break;

        case SC_PrintInt:
            EH_PrintInt();
            IncProgCounter();
            break;
    }
}

```

- Khi cần thực thi chương trình có syscall mới, ta thực hiện biên dịch.

c. Viết mã để tăng giá trị biến program counter

- Tất cả các system calls (không phải Halt) sẽ yêu cầu Nachos tăng program counter (PC) trước khi system call trả kết quả về. Nếu không lập trình đúng phần này thì Nachos sẽ bị vòng lặp gọi thực hiện system call này mãi mãi. Cũng như các hệ thống khác, MIPS xử lý dựa trên giá trị của PC, vì vậy cần tìm đoạn mã này trong thư mục machine và thực hiện tăng PC khi cần.

- Đối với Nachos, ta phải tăng 4 bytes để tăng thanh ghi PC. Giá trị 3 thanh ghi: PrevPCReg, PCReg, NextPCReg sẽ thay đổi sau mỗi lần thực hiện xong syscall (trước khi trả kết quả). Hàm IncProgCounter() sẽ thực hiện công việc này.

```
void IncProgCounter(){
    int counter=machine->ReadRegister(PCReg);
    machine->WriteRegister(PprevPCReg,counter);
    counter=machine->ReadRegister(NextPCReg);
    machine->WriteRegister(PCReg,counter);
    machine->WriteRegister(NextPCReg,counter+4);
}
```

- Trong hàm IncProgCounter(), biến counter lưu địa chỉ của thanh ghi PCReg. Sau đó, hàm WriteRegister thực hiện ghi counter vào PrevPCReg, thực hiện tương tự để lưu địa chỉ thanh ghi PCReg thành địa chỉ của NextPCReg. Tăng giá trị của NextPCReg lên 4 bytes.

- Để sử dụng class Machine ta cần khai báo nó như biến toàn cục (biến toàn cục được lưu và cấp phát tại `"/code/threads/system.cc", "/code/threads/system.h"`).

- Với mỗi system call được xử lý trong ExceptionHandler, ta đều gọi hàm IncProgCounter() ở cuối các hàm xử lý system call.

d. Cài đặt system call int ReadInt().

- Sử dụng hàm read thuộc lớp SynchConsole để đọc một số nguyên do người dùng nhập vào. Nếu giá trị người dùng nhập không phải là số nguyên thì trả về zero (0)

- Trước tiên cần xác định dấu của số nguyên vừa nhập bằng cách kiểm tra ký tự đầu tiên có phải '-' hay không. Nếu số vừa nhập là số nguyên âm thì biến bool isNegative sẽ mang giá trị True và ngược lại.

- Tiếp theo là chuyển các ký tự thành số bằng công thức `re=re*10+(value[i]-'0')`, ta khởi tạo biến re = 0 để lưu kết quả.

- Kết thúc quá trình chuyển, kiểm tra nếu isNegative==True ta gán re=-re.

- Trong quá trình chuyển nếu xuất hiện ký tự không phải số hoặc có nhiều hơn 1 dấu chấm (.) (trong trường hợp là số thực) thì ghi vào thanh ghi số 2 giá trị 0, ngược lại khi kết thúc quá trình chuyển ký tự và đổi dấu ta gán cho thanh ghi 2 kết quả phép chuyển *re* bằng phương thức WriteRegister của lớp machine.

e. Cài đặt system call void PrintInt(int number).

- Vì chỉ có 1 tham số kiểu int truyền vào nên giá trị sẽ được ghi lên thanh ghi 4. Sử dụng phương thức ReadRegister() của lớp machine để ghi giá trị trong thanh ghi 4 vào biến **value**. Khởi tạo biến numberOfDigit = 0 để chứa số lượng ký tự trong value và reLength = 0 để lưu độ dài value.

- Nếu giá trị **value** == 0, dùng phương thức Write in ra console kết quả.

- Kiểm tra dấu của **value**, nếu value là số nguyên âm thì đổi dấu của value và gán isNegative=True, reLength++

- Vòng lặp while dừng khi value==0, trong vòng lặp thực hiện reverse bằng $reverse * 10 + value \% 10$, value chia lấy nguyên và tăng biến numberOfDigit. Kết thúc vòng lặp, gán $reLength += numberOfDigit$

- Khởi tạo re có độ dài bằng reLength+1, nếu isNegative=True gán $re[0] = '-'$.

- Chuyển ký tự bằng cách thực hiện phép chia dư sau đó đưa ký tự vào mảng re rồi thực hiện phép chia nguyên.

- In kết quả ra console bằng phương thức Write của SynchConsole.

f. Cài đặt system call char ReadChar().

- ReadChar system call sẽ sử dụng lớp SynchConsole để đọc một ký tự do người dùng nhập vào.

- Thực hiện gọi hàm Read của lớp SynchConsole để người dùng nhập ký tự từ bàn phím và lưu số byte của ký tự vừa nhập vào biến byte.

- System call này chỉ cần đọc một ký tự nên ta có thể gán trực tiếp giá trị của ký tự này vào thanh ghi số 2.

- Vì kích thước char là 1 byte nên ta xét các trường hợp sau đối với biến byte:

- $byte == 0$: In ra thông báo “Ban chưa nhập kí tự”
- $byte == 1$: Lưu ký tự vừa nhập vào thanh ghi số 2 bằng phương thức WriteRegister của lớp machine.

- byte >1: In ra thông báo “ban da nhap nhieu hon 1 ki tu !!! Vui long thu lai”\

- Tăng giá trị trong thanh ghi PC.

g. Cài đặt system call void PrintChar(char character)

- System call này đọc ký tự được lưu trong thanh ghi số 4 bằng hàm ReadRegister của lớp Machine và lưu vào biến kiểu char **character**.

- Sau đó, in **character** ra console bằng hàm Write của lớp SynchConsole.

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

- ReadString system call sử dụng lớp SynchConsole dùng để đọc một chuỗi ký tự vào trong buffer (chuỗi sẽ kết thúc khi người dùng nhấn enter, hoặc có chiều dài lớn hơn hoặc bằng giá trị length (\geq length)).

- buffer là vùng nhớ thuộc userspace, khi người dùng nhập chuỗi sẽ được lưu ở kernel space nên ta cần viết một hàm tương ứng để chuyển dữ liệu từ kernelspace sang userspace.

- Vị trí của chuỗi sẽ được lưu ở thanh ghi số 4, ta thực hiện lưu địa chỉ này vào biến virtAddr

- Tương tự, ta lưu độ dài chuỗi từ thanh ghi số 5 vào biến len

- Khởi tạo một chuỗi **buffer** có độ dài bằng len +1 (+1 vì chuỗi phải chứa ký tự '\0'). Sau đó, thực hiện đọc chuỗi trên màn hình bằng hàm Read của lớp SynchConsole và lưu **buffer** (cần tự thêm ký tự '\0' vào cuối chuỗi).

- Thực hiện chuyển dữ liệu từ kernelspace sang userspace, ta sử dụng phương thức WriteMem() từ lớp machine để ghi dữ liệu lên vùng nhớ của userspace.

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

- PrintString system call dùng để in chuỗi ký tự trong buffer ra màn hình. Tương tự như ReadString ta cũng cần có hàm để chuyển dữ liệu từ userspace qua kernel space.

- Đọc địa chỉ của chuỗi trong thanh ghi số 4 bằng phương thức ReadRegister() của lớp machine vào biến virtAddr.

- Vòng lặp do..while để tính độ dài chuỗi và kết thúc khi gặp ký tự '\0'. Lưu độ dài chuỗi vào biến len.

- Khởi tạo biến buffer kiểu char có độ dài bằng len.

- Thực hiện sao chép từng kí tự của vùng nhớ user space sang kernel space bằng phương thức ReadMem của lớp machine trong vòng lặp for (lưu kí tự trong biến buffer)
- Dùng phương thức Write để xuất chuỗi được lưu trong biến buffer ra màn hình.

j. Viết chương trình help, CT help dùng để in ra các dòng giới thiệu cơ bản về nhóm và mô tả vắn tắt về chương trình sort và ascii

- Thêm help vào dòng bắt đầu bằng “all” trong Makefile (nachos-3.4/code/test)

```

Makefile (~/Desktop/nachos/nachos-3.4/code/test) - gedit
Makefile x
CC = $(GCCDIR)gcc -B../gnu-decstation-ultrix/
AS = $(GCCDIR)as
LD = $(GCCDIR)ld

CPP = gcc -E
INCDIR = -I../userprog -I../threads
CFLAGS = -G 0 -c $(INCDIR)

all: halt shell matmult readint readchar readstring help ascii sort

```

- Tiếp tục chỉnh sửa Makefile, thêm đoạn code sau

```

Makefile x
readchar: readchar.o start.o
    $(LD) $(LDFLAGS) start.o readchar.o -o readchar.coff
    ../bin/coff2noff readchar.coff readchar

readstring.o: readstring.c
    $(CC) $(CFLAGS) -c readstring.c
readstring: readstring.o start.o
    $(LD) $(LDFLAGS) start.o readstring.o -o readstring.coff
    ../bin/coff2noff readstring.coff readstring

help.o: help.c
    $(CC) $(CFLAGS) -c help.c
help: help.o start.o
    $(LD) $(LDFLAGS) start.o help.o -o help.coff
    ../bin/coff2noff help.coff help

ascii.o: ascii.c
    $(CC) $(CFLAGS) -c ascii.c
ascii: ascii.o start.o
    $(LD) $(LDFLAGS) start.o ascii.o -o ascii.coff
    ../bin/coff2noff ascii.coff ascii

sort.o: sort.c
    $(CC) $(CFLAGS) -c sort.c
sort: sort.o start.o
    $(LD) $(LDFLAGS) start.o sort.o -o sort.coff
    ../bin/coff2noff sort.coff sort
  
```

Makefile Tab Width: 8 Ln 86, Col 1 INS

- Tạo 1 file tên help.c trong nachos-3.4/code/test
- Viết chương trình help sử dụng hàm PrintString đã viết trước đó để in ra console theo yêu cầu đề bài

h. Viết chương trình ascii để in ra bảng mã ascii

- Thực hiện thao tác tương tự khi viết chương trình help. Thêm ascii vào dòng “all” và thêm đoạn code

```

Makefile (~/Desktop/nachos/nachos-3.4/code/test) - gedit
Makefile x
readchar: readchar.o start.o
    $(LD) $(LDFLAGS) start.o readchar.o -o readchar.coff
    ../bin/coff2noff readchar.coff readchar

readstring.o: readstring.c
    $(CC) $(CFLAGS) -c readstring.c
readstring: readstring.o start.o
    $(LD) $(LDFLAGS) start.o readstring.o -o readstring.coff
    ../bin/coff2noff readstring.coff readstring

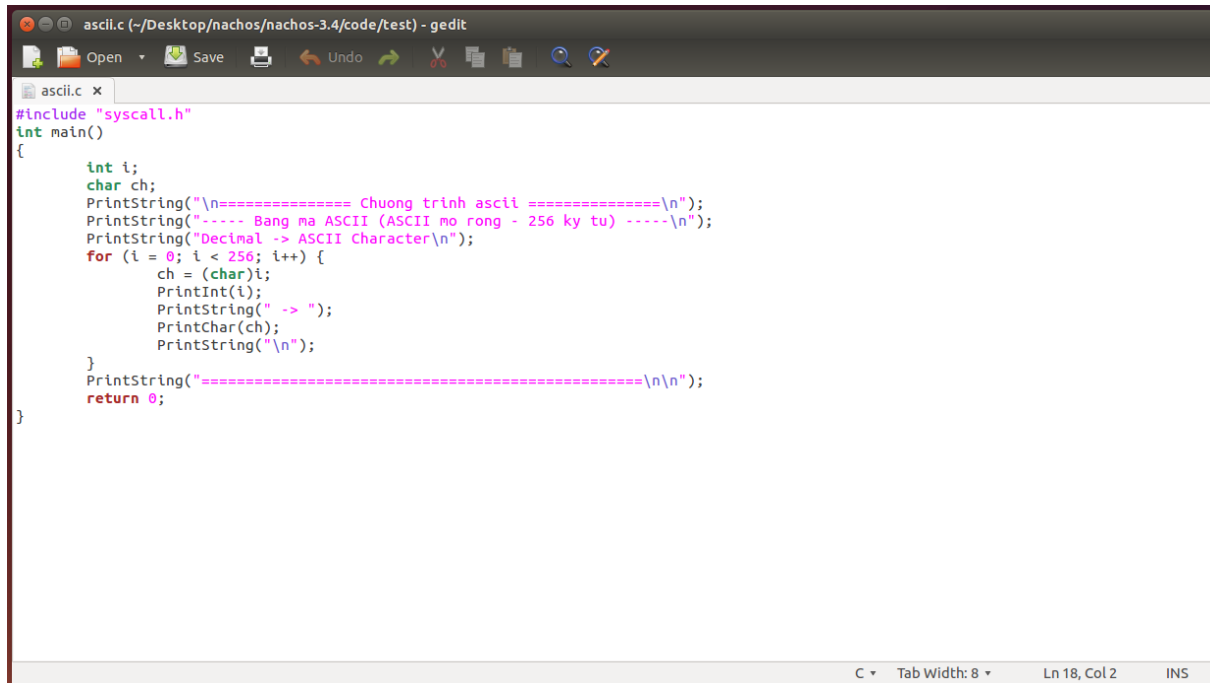
help.o: help.c
    $(CC) $(CFLAGS) -c help.c
help: help.o start.o
    $(LD) $(LDFLAGS) start.o help.o -o help.coff
    ../bin/coff2noff help.coff help

ascii.o: ascii.c
    $(CC) $(CFLAGS) -c ascii.c
ascii: ascii.o start.o
    $(LD) $(LDFLAGS) start.o ascii.o -o ascii.coff
    ../bin/coff2noff ascii.coff ascii

sort.o: sort.c
    $(CC) $(CFLAGS) -c sort.c
sort: sort.o start.o
    $(LD) $(LDFLAGS) start.o sort.o -o sort.coff
    ../bin/coff2noff sort.coff sort
  
```

Makefile ▾ Tab Width: 8 ▾ Ln 92, Col 1 INS

- Viết chương trình ascii, ta cũng dùng PrintString để in ra bảng mã ascii



```

ascii.c x
#include "syscall.h"
int main()
{
    int i;
    char ch;
    PrintString("\n===== Chương trình ascii =====\n");
    PrintString("----- Bảng mã ASCII (ASCII mở rộng - 256 ký tự) ----- \n");
    PrintString("Decimal -> ASCII Character\n");
    for (i = 0; i < 256; i++) {
        ch = (char)i;
        PrintInt(i);
        PrintString(" -> ");
        PrintChar(ch);
        PrintString("\n");
    }
    PrintString("===== \n\n");
    return 0;
}

```

i. Viết chương trình sort bằng bubble sort

- Thực hiện tương tự chương trình help, thêm sort vào dòng “all” và thêm đoạn code

```

sort.o: sort.c
$(CC) $(CFLAGS) -c sort.c
sort: sort.o start.o
$(LD) $(LDFLAGS) start.o sort.o -o sort.coff
../bin/coff2noff sort.coff sort

```

Cài đặt chương trình sort nhận số chữ số (n) và các số cần sắp xếp bằng hàm ReadInt() đã cài đặt trước đó lưu vào mảng arr. Thực hiện bubble sort để sắp xếp mảng số nguyên vừa nhập vào sau đó in ra màn hình kết quả.

```

sort.c x
#include "syscall.h"
int main()
{
    int i, j;
    int n;
    int arr[100];
    PrintString("\n===== Chuong trinh sort =====\n");
    // User input
    PrintString("Xin moi nhap so luong phan tu (n <= 100):\nn = ");
    n = ReadInt();
    for (i = 0; i < n; i++) {
        PrintString("Phan tu thu ");
        PrintInt(i);
        PrintString(": ");
        arr[i] = ReadInt();
    }
    // Bubble Sort
    for (i = 0; i < n-1; i++) {
        for (j = 0; j < n-i-1; j++) {
            if (arr[j] > arr[j+1]) {
                // Swap arr[j] and arr[j+1]
                arr[j] = arr[j] + arr[j+1];
                arr[j+1] = arr[j] - arr[j+1];
                arr[j] = arr[j] - arr[j+1];
            }
        }
    }
    // Print new order
    PrintString("\nSau khi sap xep: ");
    for (i = 0; i < n; i++) {
        if (i != 0) {PrintString(", ");}
        PrintInt(arr[i]);
    }
    PrintString("\n===== \n\n");
    return 0;
}

```

C Tab Width: 8 Ln 17, Col 23 INS

PHẦN 4: DEMO

Viết chương trình ở mức người dùng để có thể kiểm tra các syscall vừa nhập

1. Đọc một số nguyên do người dùng nhập vào

```
phucpham@PVM: ~/Downloads/Nachos_Project1/nachos/nachos-3.4/code
phucpham@PVM:~/Downloads/Nachos_Project1/nachos/nachos-3.4/code$ ./userprog/nachos -rs 1234 -x ./test/readint
123
123Machine halting!

Ticks: total 188249414, idle 188249238, system 150, user 26
Disk I/O: reads 0, writes 0
Console I/O: reads 4, writes 4
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
phucpham@PVM:~/Downloads/Nachos_Project1/nachos/nachos-3.4/code$
```

Số nguyên dương

```
phucpham@PVM: ~/Downloads/Nachos_Project1/nachos/nachos-3.4/code
phucpham@PVM:~/Downloads/Nachos_Project1/nachos/nachos-3.4/code$ ./userprog/nachos -rs 1234 -x ./test/readint
-123
-123Machine halting!

Ticks: total 554399114, idle 554398908, system 180, user 26
Disk I/O: reads 0, writes 0
Console I/O: reads 5, writes 5
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
phucpham@PVM:~/Downloads/Nachos_Project1/nachos/nachos-3.4/code$
```

Số nguyên âm

- Nhập kí tự không phải số nguyên (ct in thông báo lỗi)

```
phucpham@PVM: ~/Downloads/nachos/nachos-3.4/code
phucpham@PVM:~/Downloads/nachos/nachos-3.4/code$ ./userprog/nachos -rs 1234 -x .
/test/readint
abc
Ban nhap ki tu khong phai so nguyen
0Machine halting!

Ticks: total 716854084, idle 716853938, system 120, user 26
Disk I/O: reads 0, writes 0
Console I/O: reads 4, writes 2
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
phucpham@PVM:~/Downloads/nachos/nachos-3.4/code$
```

- Nhập số thực (ct chỉ xuất phần nguyên)

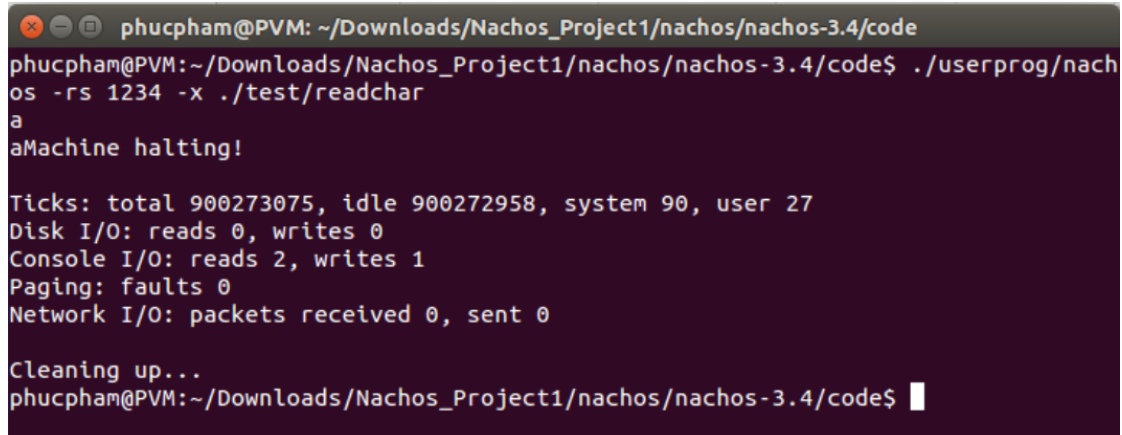
```
phucpham@PVM: ~/Downloads/nachos/nachos-3.4/code
phucpham@PVM:~/Downloads/nachos/nachos-3.4/code$ ./userprog/nachos -rs 1234 -x .
/test/readint
123.456
123Machine halting!

Ticks: total 431279604, idle 431279398, system 180, user 26
Disk I/O: reads 0, writes 0
Console I/O: reads 8, writes 4
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
phucpham@PVM:~/Downloads/nachos/nachos-3.4/code$
```

1. Đọc một kí tự do người dùng nhập vào

- Nhập đúng

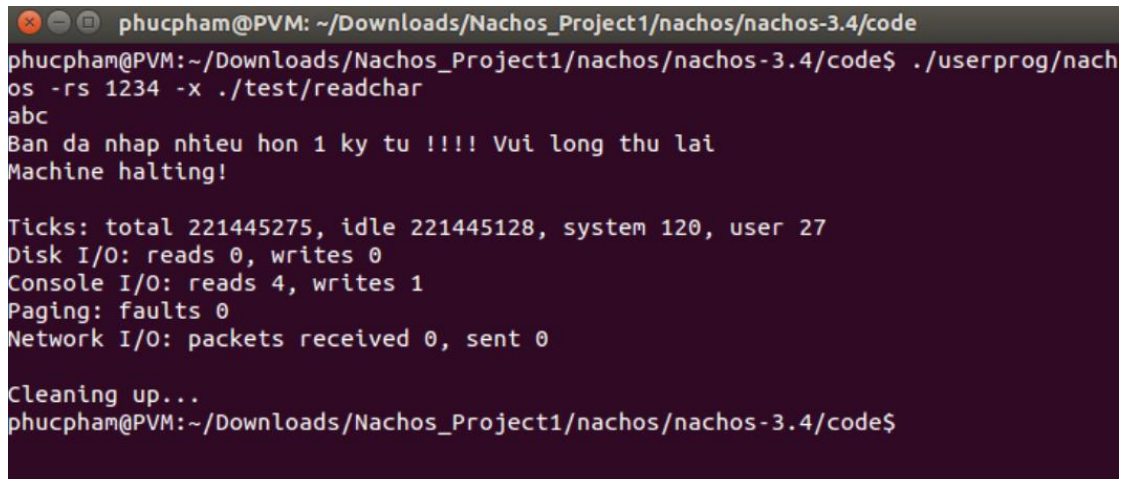


```
phucpham@PVM: ~/Downloads/Nachos_Project1/nachos/nachos-3.4/code
phucpham@PVM:~/Downloads/Nachos_Project1/nachos/nachos-3.4/code$ ./userprog/nachos -rs 1234 -x ./test/readchar
a
aMachine halting!

Ticks: total 900273075, idle 900272958, system 90, user 27
Disk I/O: reads 0, writes 0
Console I/O: reads 2, writes 1
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
phucpham@PVM:~/Downloads/Nachos_Project1/nachos/nachos-3.4/code$
```

- Nhập nhiều hơn một kí tự (chương trình thông báo lỗi)



```
phucpham@PVM: ~/Downloads/Nachos_Project1/nachos/nachos-3.4/code
phucpham@PVM:~/Downloads/Nachos_Project1/nachos/nachos-3.4/code$ ./userprog/nachos -rs 1234 -x ./test/readchar
abc
Ban da nhap nhieu hon 1 ky tu !!!! Vui long thu lai
Machine halting!

Ticks: total 221445275, idle 221445128, system 120, user 27
Disk I/O: reads 0, writes 0
Console I/O: reads 4, writes 1
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
phucpham@PVM:~/Downloads/Nachos_Project1/nachos/nachos-3.4/code$
```

- Không nhập kí tự (chương trình thông báo lỗi)

```
phucpham@PVM: ~/Downloads/Nachos_Project1/nachos/nachos-3.4/code
phucpham@PVM:~/Downloads/Nachos_Project1/nachos/nachos-3.4/code$ ./userprog/nach
os -rs 1234 -x ./test/readchar

Ban chua nhap ki tu !
Machine halting!

Ticks: total 81507975, idle 81507868, system 80, user 27
Disk I/O: reads 0, writes 0
Console I/O: reads 1, writes 1
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
phucpham@PVM:~/Downloads/Nachos_Project1/nachos/nachos-3.4/code$
```

2. Đọc một chuỗi do người dùng nhập vào

```
phucpham@PVM: ~/Downloads/Nachos_Project1/nachos/nachos-3.4/code
phucpham@PVM:~/Downloads/Nachos_Project1/nachos/nachos-3.4/code$ ./userprog/nach
os -rs 1234 -x ./test/readstring
Nhap vao chuoi:he dieu hanh
he dieu hanhMachine halting!

Ticks: total 366624537, idle 366623932, system 570, user 35
Disk I/O: reads 0, writes 0
Console I/O: reads 13, writes 27
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
phucpham@PVM:~/Downloads/Nachos_Project1/nachos/nachos-3.4/code$
```

3. Chương trình help in ra các dòng giới thiệu cơ bản về nhóm và mô tả chương trình sort, ASCII

```

hdh@ubuntu: ~/Desktop/nachos_team/nachos-3.4/code
make[1]: Entering directory `/home/hdh/Desktop/nachos_team/nachos-3.4/code/bin'
make[1]: Nothing to be done for `all'.
make[1]: Leaving directory `/home/hdh/Desktop/nachos_team/nachos-3.4/code/bin'
cd test; make all
make[1]: Entering directory `/home/hdh/Desktop/nachos_team/nachos-3.4/code/test'
make[1]: Nothing to be done for `all'.
make[1]: Leaving directory `/home/hdh/Desktop/nachos_team/nachos-3.4/code/test'
hdh@ubuntu:~/Desktop/nachos_team/nachos-3.4/code$ ./userprog/nachos -rs 1023 -x ./test/help

===== Chương trình help =====
----- Giới thiệu về nhóm -----
Nhóm do an 2 môn Hệ điều hành gồm có 5 thành viên, bao gồm:
1. 19120257 - Phạm Anh Khoa
2. 19120301 - Võ Thanh Nam
3. 19120331 - Phạm Lưu Mỹ Phúc
4. 19120389 - Tô Gia Thuận
5. 19120454 - Bùi Quang Bảo
----- Giới thiệu về chương trình -----
1. Chương trình Ascii:
   Sử dụng để in ra màn hình bảng mã Ascii.
   Cách chạy: ./userprog/nachos -rs 1023 -x ./test/ascii
2. Chương trình Sort:
   Sử dụng để sắp xếp n số nguyên theo thuật toán Bubble Sort.
   Với n <= 100, do người dùng nhập.
   Cách chạy: ./userprog/nachos -rs 1023 -x ./test/sort
=====

Machine halting!

Ticks: total 73985, idle 66200, system 7650, user 135
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 662
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
hdh@ubuntu:~/Desktop/nachos_team/nachos-3.4/code$

```


4. Chương trình ASCII in ra bảng mã ASCII

```
phucpham@PVM: ~/Downloads/nachos/nachos-3.4/code
/test/ascii

===== Chương trình ascii =====
----- Bảng mã ASCII (ASCII mở rộng - 256 ký tự) -----
Decimal -> ASCII Character
0 ->
1 -> 0
2 -> 1
3 -> 2
4 -> 3
5 -> 4
6 -> 5
7 -> 6
8 ->
9 ->
10 ->
11 ->
```

4.1

```
hdh@ubuntu: ~/Desktop/nachos_team/nachos-3.4/code
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
80 -> P
81 -> Q
82 -> R
83 -> S
84 -> T
85 -> U
86 -> V
87 -> W
88 -> X
89 -> Y
90 -> Z
91 -> [
92 -> \
```

4.2

```

235 ->
236 ->
237 ->
238 ->
239 ->
240 ->
241 ->
242 ->
243 ->
244 ->
245 ->
246 ->
247 ->
248 ->
249 ->
250 ->
251 ->
252 ->
253 ->
254 -> ♦
255 -> ♦
=====
Machine halting!

Ticks: total 327241, idle 263500, system 52930, user 10811
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 2635
Paging: faults 0
Network I/O: packets received 0, sent 0

```

4.2

5. Chương trình sort sắp xếp mảng số nguyên bằng bubble sort

```

hdh@ubuntu: ~/Desktop/nachos_team/nachos-3.4/code
254 -> ♦
255 -> ♦
=====
Machine halting!

Ticks: total 327241, idle 263500, system 52930, user 10811
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 2635
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
hdh@ubuntu:~/Desktop/nachos_team/nachos-3.4/code$ ./userprog/nachos -rs 1023 -x ./test/sort

===== Chương trình sort =====
Xin moi nhap so luong phan tu (n <= 100):
n = 5
Phan tu thu 0: 12
Phan tu thu 1: 78
Phan tu thu 2: 0
Phan tu thu 3: -9
Phan tu thu 4: 101

Sau khi sap xep: -9, 0, 12, 78, 101
=====
Machine halting!

Ticks: total 1892802707, idle 1892797137, system 4280, user 1290
Disk I/O: reads 0, writes 0
Console I/O: reads 17, writes 271
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
hdh@ubuntu:~/Desktop/nachos_team/nachos-3.4/code$ █

```

TÀI LIỆU THAM KHẢO

- Chuỗi video hướng dẫn Lập trình Nachos HCMUS - Nguyễn Thành Chung
- CS 422 Assignments: homes.cs.washington.edu/~arvind/cs422/assignments