

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

-----*****-----



BÁO CÁO ĐỒ ÁN: NACHOS

Giáo viên hướng dẫn: TS. Lê Viết Long

Sinh viên thực hiện:

20120138

20120187

20120555

20120146

Lê Thành Nam

Nguyễn Viết Thái

Nguyễn Xuân Quân

Nguyễn Thị Châu Ngọc

MỤC LỤC

I.	Thông tin thành viên	2
II.	Bảng phân công công việc	2
III.	Nội dung chính của đề án	2
1.	Hiểu mã chương trình nachos:	2
2.	Hiểu thiết kế:	3
3.	Exceptions và system calls	4
a)	Viết lại file exception.cc để xử lý tất cả exceptions trong machine/machine.h	4
b)	Viết lại cấu trúc điều khiển để nhận các Nachos system calls	5
c)	Tìm đoạn mã giúp tăng giá trị program counter	5
d)	Cài đặt System Call int ReadInt()	6
e)	Cài đặt System Call void PrintInt(int s)	7
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.	8
g)	Cài đặt system call void PrintChar(char character) . PrintChar system call sẽ sử dụng lớp SynchConsole để xuất một ký tự ra màn hình	8
h)	Cài đặt đặt system call void ReadString (char[] buffer, int length)	9
i)	Cài đặt system call void PrintString (char[] buffer)	10
j)	Chương trình help:	10
j.1)	Chương trình ascii:	11
j.2)	Chương trình bubble sort:	11
IV.	Tham khảo	12

I. Thông tin thành viên

MSSV	Họ tên
20120138	Lê Thành Nam
20120187	Nguyễn Viết Thái
20120555	Nguyễn Xuân Quân
20120146	Nguyễn Thị Châu Ngọc

II. Bảng phân công công việc

Họ tên	Công việc	Đánh giá
Lê Thành Nam	Câu a,b,c	100%
Nguyễn Viết Thái	Câu d,e,f	100%
Nguyễn Xuân Quân	Câu g,h	100%
Nguyễn Thị Châu Ngọc	Câu i,j	100%

III. Nội dung chính của đề án

1. Hiểu mã chương trình nachos:

Một trong những bước quan trọng đầu tiên đó chính là việc đọc và hiểu rõ ràng về nachos. Nachos là một phần mềm mã nguồn mở (open-source) giả lập một máy tính ảo và một số thành phần cơ bản của hệ điều hành chạy trên máy tính ảo này nhằm giúp cho việc tìm hiểu và xây dựng các thành phần phức tạp hơn của hệ điều hành. Máy ảo được giả lập có kiến trúc MIPS với hầu hết các thành phần và chức năng của một máy thật. Hệ điều hành Nachos chạy trên máy ảo Nachos hiện là một hệ điều hành đơn chương.

Để biên dịch Nachos trên Linux, sử dụng trình biên dịch gcc. Các chương trình C trên hệ điều hành Nachos cho kiến trúc máy MIPS được biên dịch thành các chương trình thực thi dựa trên một trình biên dịch gọi là Cross-compiler. Một chương trình Halt đơn giản đã được cung cấp để thử nghiệm Nachos, vai trò của chương trình này là bắt hệ điều hành tắt máy. Chạy chương trình “nachos -rs 1234 -x ../test/halt”. Dò tìm khi chương trình người dùng nạp, chạy, và gọi một system call.

Các tập tin trong đề án này:

- **Progtest.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, trong phần mã chúng tôi cung cấp, chỉ có 'halt' system call được viết.
- **bitmap.***: các hàm xử lý cho lớp bitmap (hữu ích cho việc lưu vết các ô nhớ vật lý) **fileys.h**.
- **Openfile.h**: định nghĩa các hàm trong hệ thống file nachos.
- **translate.***: Ở phiên bản Nachos hiện tại, giả sử mỗi địa chỉ ảo là cũng giống hệt như địa chỉ vật lý, điều này giới hạn chúng ta chỉ chạy 1 chương trình tại một thời điểm. Ta có thể viết lại phần này để cho phép nhiều chương trình chạy cùng lúc.
- **machine.***: mô phỏng các thành phần của máy tính khi thực thi chương trình người dùng: bộ nhớ chính, thanh ghi, v.v.
- **Mipssim.cc**: mô phỏng tập lệnh của MIPS R2/3000 processor.
- **console.***: mô phỏng thiết bị đầu cuối sử dụng UNIX files. Một thiết bị có đặc tính (i) đơn vị dữ liệu theo byte, (ii) đọc và ghi các bytes cùng một thời điểm, (iii) các bytes đến bất đồng bộ.
- **synchconsole.***: nhóm hàm cho việc quản lý nhập xuất I/O theo dòng trong Nachos.
- **../test/***: Các chương trình C sẽ được biên dịch theo MIPS và chạy trong Nachos.

2. Hiểu thiết kế:

Để hiểu HĐH làm việc như thế nào, chúng ta cần phải hiểu và phân biệt được kernel (system space) và user space. Mỗi chương trình trong hệ thống phải có các thông tin cục bộ của nó, bao gồm 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 calls, software interrupt/trap. Trước khi gọi một lệnh trong hệ thống thì các tham số truyền vào cần thiết phải được nạp vào các thanh ghi của CPU. Để chuyển một biến mang giá trị, tiến trình chỉ việc ghi giá trị vào thanh ghi. Để chuyển một biến tham chiếu, thì giá trị lưu trong thanh ghi đc xem như là "user space pointer". Bởi vì user space pointer không có ý nghĩa đối với kernel, mà chúng ta cần là chuyển nội dung từ user space vào kernel sao cho ta có thể xử lý dữ liệu này. Khi trả thông tin từ system về user space, thì các giá trị phải đặt trong các thanh ghi của CPU.

Nachos cung cấp một CPU giả lập, thực tế CPU giả lập này giống hệt CPU thật (MIPS 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 như thế nào.

Nachos cung cấp môi trường giả lập để chạy các chương trình bằng C, xem Makefile trong thư mục test, chương trình biên dịch phải link với vài flag, sao đó chuyển sang định dạng đặc biệt của Nachos, dùng công cụ “coff2noff”(đây là công cụ được viết sẵn trong phần mềm Nachos và được dùng để chuyển đổi định dạng COFF thành định dạng NOFF, định dạng chạy trên hệ điều hành Nachos cho kiến trúc máy MIPS.)

3. Exceptions và system calls

a) Viết lại file exception.cc để xử lý tất cả exceptions trong machine/machine.h

- Danh sách các exception:

```
enum ExceptionType { NoException,           // Everything ok!
                    SyscallException,       // A program executed a system call.
                    PageFaultException,     // No valid translation found
                    ReadOnlyException,      // Write attempted to page marked
                                           // "read-only"
                    BusErrorException,      // Translation resulted in an
                                           // invalid physical address
                    AddressErrorException,   // Unaligned reference or one that
                                           // was beyond the end of the
                                           // address space
                    OverflowException,      // Integer overflow in add or sub.
                    IllegalInstrException,  // Unimplemented or reserved instr.

                    NumExceptionTypes
};
```

- Viết lại các exception trong file exception.cc:

```
case NoException:
    return;
case PageFaultException:
    DEBUG('a', "\n\t No valid translation found");
    printf("\n\n\t No valid translation found");
    interrupt->Halt();
    break;
case ReadOnlyException:
    DEBUG('a', "\n\t Write attempted to page marked read-only");
    printf("\n\n\t Write attempted to page marked read-only");
    interrupt->Halt();
    break;
case BusErrorException:
    DEBUG('a', "\n\t Write attempted to page marked read-only");
    printf("\n\n\t Write attempted to page marked read-only");
    interrupt->Halt();
    break;
case AddressErrorException:
    DEBUG('a', "\n\t Unaligned reference or one that was beyond the end of the address space");
    printf("\n\n\t Unaligned reference or one that was beyond the end of the address space");
    interrupt->Halt();
    break;
case OverflowException:
    DEBUG('a', "\n\t Integer overflow in add or sub.");
    printf("\n\n\t Integer overflow in add or sub.");
    interrupt->Halt();
    break;
case IllegalInstrException:
    DEBUG('a', "\n\t Unimplemented or reserved instr.");
    printf("\n\n\t Unimplemented or reserved instr.");
    interrupt->Halt();
    break;
case NumExceptionTypes:
    DEBUG('a', "\n\t Number exception types");
    printf("\n\n\t Number exception types");
    interrupt->Halt();
    break;
```

b) Viết lại cấu trúc điều khiển để nhận các Nachos system calls

- Sửa lại cấu trúc If Else thành Switch Case trong file exception.cc

```
case SyscallException:
{
    switch (type)
    {
    case SC_Halt:
    {
        DEBUG('a', "\n Shutdown, initiated by user program.");
        printf("\n\n Shutdown, initiated by user program.");
        interrupt->Halt();
        break;
    }
    case SC_Create:
    {
        int virtAddr;
        char *filename;
        DEBUG('a', "\n SC_Create call ...");
        DEBUG('a', "\n Reading virtual address of filename");

        virtAddr = machine->ReadRegister(4);
        DEBUG('a', "\n Reading filename.");

        filename = User2System(virtAddr, MaxFileLength + 1);
        if (filename == NULL)
        {
            printf("\n Not enough memory in system");
            DEBUG('a', "\n Not enough memory in system");
            machine->WriteRegister(2, -1);
            delete filename;
            return;
        }
        DEBUG('a', "\n Finish reading filename.");

        if (!fileSystem->Create(filename, 0))
        {
            printf("\n Error create file '%s'", filename);
            machine->WriteRegister(2, -1);
            delete filename;
            return;
        }
        machine->WriteRegister(2, 0);
        delete[] filename;
        break;
    }
}
```

- Kết quả chạy chương trình Halt:

```
make[1]: Leaving directory `/home/nam/Downloads/nachos/nachos-3.4/code/test'
nam@ubuntu:~/Downloads/nachos/nachos-3.4/code$ cd threads
nam@ubuntu:~/Downloads/nachos/nachos-3.4/code/threads$ ./nachos
No threads ready or runnable, and no pending interrupts.
Assuming the program completed.
Machine halting!

Ticks: total 10, idle 0, 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...
nam@ubuntu:~/Downloads/nachos/nachos-3.4/code/threads$
```

Ta sẽ tiến hành cài đặt các System Calls tương tự theo các của System Call Halt.

c) Tìm đoạn mã giúp tăng giá trị program counter

- Ta tiến hành tạo hàm tăng giá trị của program counter trong exception.cc như hình dưới:

```
// Increase programming counter
void IncreaseProgramCounter()
{
    int counter = machine->ReadRegister(PCReg); //lay gia tri thanh ghi hien tai
    machine->WriteRegister(PrevPCReg, counter); //gan gia tri thanh ghi truoc = hien tai
    counter = machine->ReadRegister(NextPCReg); //lay gia tri thanh ghi ke tiep
    machine->WriteRegister(PCReg, counter); //gan gia tri thanh ghi hien tai = thanh ghi ke tiep
    machine->WriteRegister(NextPCReg, counter + 4); //gan thanh ghi ke tiep ve sau 4 byte de tiep tuc nap lenh
}
```

Khi ta tiến hành xử lý một System Call bất kỳ thì sẽ sử dụng hàm tăng giá trị program counter này.

d) Cài đặt System Call **int ReadInt()**

Trước khi xử lý các System Call, ta tiến hành cài đặt file Syncons để thao tác với màn hình console như sau:

- Copy file synchcons.h và synchcons.cc vào thư mục code/threads
- Mở file code/Makefile.common, thêm ../threads/synchcons.* vào USERPROG_* (* ở đây là h, cc, o)

```
USERPROG_H = ../userprog/addrspace.h\
    ../userprog/bitmap.h\
    ../filesys/filesys.h\
    ../filesys/openfile.h\
    ../machine/console.h\
    ../machine/machine.h\
    ../machine/mipssim.h\
    ../threads/synchcons.h\
    ../machine/translate.h
```

```
USERPROG_C = ../userprog/addrspace.cc\
    ../userprog/bitmap.cc\
    ../userprog/exception.cc\
    ../userprog/progtest.cc\
    ../machine/console.cc\
    ../threads/synchcons.cc\
    ../machine/machine.cc\
    ../machine/mipssim.cc\
    ../machine/translate.cc
```

```
USERPROG_O = addrspace.o bitmap.o exception.o progtest.o console.o machine.o \
    mipssim.o translate.o synchcons.o
```

- Khai báo biến toàn cục gSynchConsole trong threads/system.h

Cách xử lý system call **SC_ReadInt:**

- Lưu ký tự người dùng nhập vào char* buff
- Xác định số âm hay dương bằng cách đọc ký tự đầu tiên buff[0], nếu âm thì bool Negative = true và ngược lại
- Xử lý trường hợp số có dấu chấm (vd: 12.000 = 12), nếu sau dấu chấm là số khác 0 thì xuất ra thông báo lỗi.

- Trường hợp ký tự đọc được ngoài khoảng 0-9, xuất ra thông báo lỗi.
- Sau khi đã hợp lệ, ta chuyển chuỗi ký tự người dùng nhập vào (char* buff) thành kiểu int bằng công thức: $num = num * 10 + (int)(buff[i] - 48)$
- Xét dấu, nếu là số âm thì nhân kết quả num với -1, số dương thì giữ nguyên

e) Cài đặt System Call **void PrintInt(int s)**

Hàm xử lý system call **SC_PrintInt**:

- Xét int s là âm hay dương, nếu là âm thì bool Negative = true và ngược lại char* buff lưu ký tự chuỗi để xuất ra màn hình
- Chuyển int s thành kiểu char để lưu vào char* buff qua vòng lặp:

```
buff[i] = (char)(s % 10 + 48)
```

```
s /= 10
```

- Trường hợp s là số âm, buff[0] = '-'

Kiểm thử bằng chương trình mức người dùng ReadInt.c (gồm cả 2 hàm ReadInt() và PrintInt())

- Kiểm thử 1:

```
Cleaning up...
nam@ubuntu:~/Downloads/20120138_20120555_20120187_20120146/Source/nachos/nachos-3.4/code$ ./userprog/nachos -rs 1023 -x ./test/ReadInt
Nhập số: 83123
Số nhập: 83123
Shutdown, initiated by user program. Machine halting!

Ticks: total 686776184, idle 686775664, system 480, user 40
Disk I/O: reads 0, writes 0
Console I/O: reads 6, writes 25
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
nam@ubuntu:~/Downloads/20120138_20120555_20120187_20120146/Source/nachos/nachos-3.4/code$
```

Input: 83123

Output: 83123

- Kiểm thử 2:

```
Cleaning up...
nam@ubuntu:~/Downloads/20120138_20120555_20120187_20120146/Source/nachos/nachos-3.4/code$ ./userprog/nachos -rs 1023 -x ./test/ReadInt
Nhập số: 123.4

Số nhập: 0 Không hợp lệ
Shutdown, initiated by user program. Machine halting!

Ticks: total 569858834, idle 569858374, system 420, user 40
Disk I/O: reads 0, writes 0
Console I/O: reads 6, writes 21
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
nam@ubuntu:~/Downloads/20120138_20120555_20120187_20120146/Source/nachos/nachos-3.4/code$
```

Input: 123.4

Output: 0 Không hợp lệ

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.

- Code SC_ReadChar trong file exception.cc

```

case SC_ReadChar:
{
    int maxBytes = 255;
    char* buffer = new char[255];
    int numBytes = gSynchConsole->Read(buffer, maxBytes); //numBytes = số
byte của buffer = số kí tự
    if(numBytes > 1) //Nếu nhập nhiều hơn 1 ký tự thì không hợp lệ
    {
        printf("Chỉ được nhập duy nhất 1 ký tự!");
        DEBUG('a', "\nERROR: Chỉ được nhập duy nhất 1 ký tự!");
        machine->WriteRegister(2, 0);
    }
    else if(numBytes == 0) //Ký tự rỗng
    {
        printf("Ký tự rỗng!");
        DEBUG('a', "\nERROR: Ký tự rỗng!");
        machine->WriteRegister(2, 0);
    }
    else
    {
        //Chuỗi vừa lấy có đúng 1 ký tự, lấy ký tự ở index = 0, return vào
thành ghi R2
        char c = buffer[0];
        machine->WriteRegister(2, c);
    }
    delete buffer;
    break;
}

```

- Cách xử lý: sử dụng biến numBytes để lấy số lượng byte tại buffer khi dùng lệnh Read, vì 1 ký tự kiểu Char sẽ có giá trị là 1 byte nên lúc này biến numBytes chính là số ký tự trong buffer. Nếu numBytes > 1 hoặc = 0 hay số lượng ký tự vượt quá 1 hoặc không có ký tự nào thì trả về thông báo lỗi. Ở trong đoạn code này có biến maxBytes là số lượng phần tử của con trỏ buffer và số Bytes được Read, và biến này có thể được khai báo với số khác 255 chỉ cần số này > 0 và <= 255.

g) Cài đặt system call **void PrintChar(char character)**. **PrintChar** system call sẽ sử dụng lớp SynchConsole để xuất một ký tự ra màn hình

- Code SC_PrintChar trong file exception.cc

```

case SC_PrintChar:
{
    char c = (char)machine->ReadRegister(4); // Đọc ký tự từ thanh ghi r4
    gSynchConsole->Write(&c, 1); // In ký tự từ biến c, 1 byte
    break;
}

```

- Cách xử lý: đọc ký tự đã được nhập từ Register 4 và in ra màn hình

* Hình ảnh test hai SC trên:

- Trường hợp nhập đúng:

```
Cleaning up...
nam@ubuntu:~/Downloads/20120138_20120555_20120187_20120146/Source/nachos/nachos-3.4/code$ ./userprog/nachos -rs 1023 -x ./test/ReadChar
Nhap ki tu: a
Ki tu nhap: a
Shutdown, initiated by user program.Machine halting!

Ticks: total 320321005, idle 320320474, system 490, user 41
Disk I/O: reads 0, writes 0
Console I/O: reads 2, writes 27
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
nam@ubuntu:~/Downloads/20120138_20120555_20120187_20120146/Source/nachos/nachos-3.4/code$
```

- Trường hợp nhập nhiều hơn 1 ký tự:

```
Cleaning up...
nam@ubuntu:~/Downloads/20120138_20120555_20120187_20120146/Source/nachos/nachos-3.4/code$ ./userprog/nachos -rs 1023 -x ./test/ReadChar
Nhap ki tu: a
Ki tu nhap: a
Shutdown, initiated by user program.Machine halting!

Ticks: total 320321005, idle 320320474, system 490, user 41
Disk I/O: reads 0, writes 0
Console I/O: reads 2, writes 27
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
nam@ubuntu:~/Downloads/20120138_20120555_20120187_20120146/Source/nachos/nachos-3.4/code$
```

= Trường hợp không nhập

```
Cleaning up...
nam@ubuntu:~/Downloads/20120138_20120555_20120187_20120146/Source/nachos/nachos-3.4/code$ ./userprog/nachos -rs 1023 -x ./test/ReadChar
Nhap ki tu:
Ki tu nhap: Ky tu rong!
Shutdown, initiated by user program.Machine halting!

Ticks: total 150989755, idle 150989284, system 430, user 41
Disk I/O: reads 0, writes 0
Console I/O: reads 1, writes 27
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
nam@ubuntu:~/Downloads/20120138_20120555_20120187_20120146/Source/nachos/nachos-3.4/code$
```

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

- Code SC_ReadString trong file exception.cc

```
case SC_ReadString:
{
    int virtAddr, length;
    char* buffer;
    virtAddr = machine->ReadRegister(4); // Lay dia chi tham so buffer
    // truyen vao tu thanh ghi so 4
    length = machine->ReadRegister(5); // Lay do dai toi da cua chuoai
    // nhap vao tu thanh ghi so 5
    buffer = User2System(virtAddr, length); // Copy chuoai tu vung nho
    // User Space sang System Space
    gSynchConsole->Read(buffer, length); // Goi ham Read cua SynchConsole
    // de doc chuoai
    System2User(virtAddr, length, buffer); // Copy chuoai tu vung nho
    // System Space sang vung nho User Space
    delete buffer;
    IncreaseProgramCounter(); // Tang Program Counter
    break;
}
```

- Cách xử lý: từ một mảng ký tự rỗng ở User Space chuyển vào System Space, sau đó dùng lệnh Read để đọc chuỗi ký tự vào trong buffer rỗng này, và chuyển lại ra User Space để hiển thị

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

- Code SC_PrintString trong file exception.cc

```
case SC_PrintString:
{
    int virtAddr;
    char* buffer;
    virtAddr = machine->ReadRegister(4); // Lay dia chi cua tham so
    buffer tu thanh ghi so 4
    buffer = User2System(virtAddr, 255); // Copy chuỗi từ vùng nhớ User
    Space sang System Space với bộ đệm buffer dài 255 kí tự
    int length = 0;
    while (buffer[length] != 0) length++; // Đếm độ dài thật của chuỗi
    gSynchConsole->Write(buffer, length + 1); // Gọi hàm Write của
    SynchConsole để in chuỗi
    delete buffer;
    break;
}
```

- Cách xử lý: chuyển mảng ký tự buffer từ User Space vào System Space, sau đó dùng lệnh Write để viết từng ký tự lên màn hình.

* Hình ảnh test hai SC trên:

```
Cleaning up...
nam@ubuntu:~/Downloads/20120138_20120555_20120187_20120146/Source/nachos/nachos-3.4/code$ ./userprog/nachos -rs 1023 -x ./test/ReadString
Nhập chuỗi kí tự: le thanh nam

Chuỗi đã nhập: le thanh nam
Shutdown, initiated by user program.Machine halting!

Ticks: total 814949164, idle 814948259, system 860, user 45
Disk I/O: reads 0, writes 0
Console I/O: reads 14, writes 48
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
nam@ubuntu:~/Downloads/20120138_20120555_20120187_20120146/Source/nachos/nachos-3.4/code$
```

j) Chương trình help:

- Yêu cầu: in ra dòng thông báo thể hiện thông tin của nhóm và mô tả vắn tắt về chương trình sort và ascii
- Cách làm: sử dụng hàm PrintString ở trên
- Kết quả:

```
Cleaning up...
nam@ubuntu:~/Downloads/20120138_20120555_20120187_20120146/Source/nachos/nachos-3.4/code$ ./userprog/nachos -rs 1023 -x ./test/help
Nhóm gồm 4 thành viên:
- Le Thanh Nam
- Nguyen Xuan Quan
- Nguyen Thi Chau Ngoc
- Nguyen Viet Thai
=====
Giới thiệu về chương trình sort và ascii
- Sort: Sắp xếp mảng số nguyên có n (n <= 100) được nhập từ người dùng theo thuật toán Bubble Sort
- Ascii: In ra bảng mã ascii, bắt đầu các kí tự con các kí tự không yêu cầu

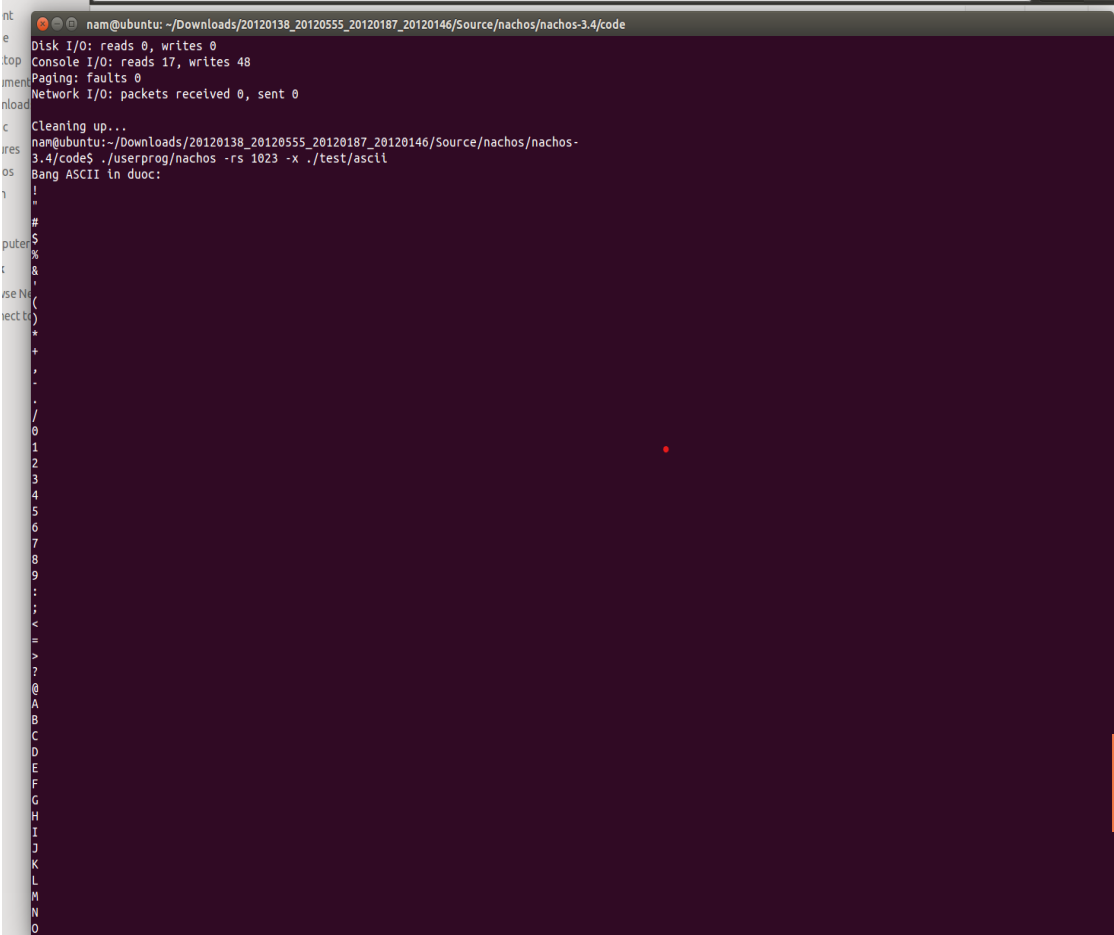
Shutdown, initiated by user program.Machine halting!

Ticks: total 40255, idle 36000, system 4180, user 75
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 360
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
nam@ubuntu:~/Downloads/20120138_20120555_20120187_20120146/Source/nachos/nachos-3.4/code$
```

j.1) Chương trình ascii:

- Yêu cầu: in ra bảng ASCII đối với các kí tự không thể đọc được thì không cần phải in.
- Cách làm: dùng hàm PrintString ở trên
- Kết quả:



```
nt
e
top
mer
nload
c
res
os
a
puter
t
se N
nect to

Disk I/O: reads 0, writes 0
Console I/O: reads 17, writes 48
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
nam@ubuntu:~/Downloads/20120138_20120555_20120187_20120146/Source/nachos/nachos-3.4/code$ ./userprog/nachos -rs 1023 -x ./test/ascii
Bang ASCII in duoc:
!
"
#
$
%
&
'
(
)
*
+
,
-
.
/
0
1
2
3
4
5
6
7
8
9
:
;
<
=
>
?
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
```

j.2) Chương trình bubble sort:

- Yêu cầu: người dùng nhập vào một dãy bé hơn 100 kí tự, sau đó người dùng chọn in kí tự đó ra theo thứ tự tăng dần hoặc giảm dần theo thuật toán bubble sort
- Cách làm: sử dụng thuật toán bubble sort như bình thường, các thao tác nhập giá trị các phần tử của mảng thì dùng hàm ReadInt, các thao tác in chuỗi dùng hàm PrintString
- Kết Quả:

```

^C
Cleaning up...
nam@ubuntu:~/Downloads/20120138_20120555_20120187_20120146/Source/nachos/nachos-3.4/code$ ./userprog/nachos -rs 1023 -x ./test/bubbleSort

Nhap vao so nguyen n: 6

Nhap so nguyen thu 1: 9

Nhap so nguyen thu 2: 6

Nhap so nguyen thu 3: 7

Nhap so nguyen thu 4: 8

Nhap so nguyen thu 5: 4

Nhap so nguyen thu 6: 3

:
1.Chuoi tang dan:
2.Chuoi giam dan:

LuaChon: 1
          3 4 6 7 8 9
Unexpected user mode exception (1)Machine halting!

Ticks: total 1940052036, idle 1940044987, system 5140, user 1909
Disk I/O: reads 0, writes 0
Console I/O: reads 16, writes 307
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
nam@ubuntu:~/Downloads/20120138_20120555_20120187_20120146/Source/nachos/nachos-3.4/code$

```

```

Cleaning up...
nam@ubuntu:~/Downloads/20120138_20120555_20120187_20120146/Source/nachos/nachos-3.4/code$ ./userprog/nachos -rs 1023 -x ./test/bubbleSort

Nhap vao so nguyen n: 6

Nhap so nguyen thu 1: 9

Nhap so nguyen thu 2: 6

Nhap so nguyen thu 3: 7

Nhap so nguyen thu 4: 8

Nhap so nguyen thu 5: 4

Nhap so nguyen thu 6: 3

:
1.Chuoi tang dan:
2.Chuoi giam dan:

LuaChon: 2
          9 8 7 6 4 3
Unexpected user mode exception (1)Machine halting!

Ticks: total 2049105482, idle 2049099658, system 4540, user 1284
Disk I/O: reads 0, writes 0
Console I/O: reads 16, writes 274
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
nam@ubuntu:~/Downloads/20120138_20120555_20120187_20120146/Source/nachos/nachos-3.4/code$

```

IV. Tham khảo

- Các tài liệu môn Hệ điều hành
- <https://nguyenvanhieu.vn/thuat-toan-sap-xep-bubble-sort/>
- Tham khảo source code nachos của Nguyễn Thành Chung:
<https://github.com/nguyenthanhchungfit/Nachos-Programing-HCMUS>