TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN
BỘ MÔN HỆ ĐIỀU HÀNH

BÁO CÁO ĐỒ ÁN

Đề Tài : HIỂU CƠ BẢN VỀ HỆ ĐIỀU HÀNH

Giảng viên : **CHUNG THÙY LINH**
Sinh viên : **HUỲNH TRỌNG THOẠI - 1512551**
Sinh viên : **NGUYỄN ANH TUẤN - 1512636**
Lớp : **15CTT3**

Contents

I. Mức độ hoàn thành đồ án:

Chức năng chương trình	Mức độ
1. Viết lại exception.cc.	Đã hoàn thành.
2. Viết lại cấu trúc điều khiển của chương trình để nhận các Nachos system calls.	Đã hoàn thành.
3. Viết mã để tăng giá trị biến program counter.	Đã hoàn thành.
4. Cài đặt system call int ReadInt().	Đã hoàn thành.
5. Cài đặt system call void PrintInt(int number).	Đã hoàn thành.
6. Cài đặt system call char ReadChar().	Đã hoàn thành.
7. Cài đặt system call void PrintChar(char character).	Đã hoàn thành.
8. Cài đặt system call void ReadString(char[] buffer,int length).	Đã hoàn thành.
9. Cài đặt system call void PrintString(char[] buffer).	Đã hoàn thành.
10. Viết chương trình help.	Đã hoàn thành.
11. Viết chương trình ascii.	Đã hoàn thành.
12. Viết chương trình sort.	Đã hoàn thành.
Tổng kết mức độ hoàn thành yêu cầu đồ án:	**100%**

II. Nội dung

1. Cách cài đặt Nachos

- Cài đặt VMWARE 8.0
- Nhập mật khẩu: 123456
- Share 2 file nachos và synchcons (VM-> setting-> Options-> Shared Folders -> Always enabled-> Add-> Browse-> Next)
- Tìm vị trí của nó: Computer-> Filesystem-> mnt-> hgfs-> nachos & synchcons

2. Thêm lớp syschcons vào nachos

- Copy 2 file Synchcons.c và Synchcons.h vào code-> threads.
- Vào code-> threads-> system.h khai báo #include "synchcons.h".
- Vào code-> threads-> system.c khai báo một biến cục bộ (Synchconsole *gsc), cấp phát nó và giải phóng vùng nhớ.
- Do mình khai báo nó là một biến cục bộ nên trong file system.h phải khai báo (extern Synchconsole *gsc).
- code-> Makefile.common khai báo 3 file Synchcons.cc, Synchcons.h, Synchcons.o trong USERPROG_C, USERPROG_H, USERPROG_O.

3. Exceptions và system calls

a. Viết lại exception.cc.

- Vào code-> machine-> machine.h để xem các loại exception.
- Vào code-> userprog-> exception.cc viết thêm các exception.
- Vào code chuột phải chọn Open in Terminal.
- Chạy bằng lệnh gmake all để kiểm tra.
- Chạy ./userprog/nachos -rs 1023 -x ./test/halt để xuất kết quả.

b. Viết lại cấu trúc điều khiển của chương trình để nhận các Nachos system calls.

- Vào code-> userprog-> syscall.h để xem các loại system calls.

- Chuyển code if…case về switch…case cho tiện trong việc kiểm tra.
- Viết các loại system calls trong case **SyscallException**.

c. Viết mã để tăng giá trị biến program counter.

- Vào code-> machine-> mipssim.cc để xem đoạn mã.
- Vị trí trước đó gán cho vị trí hiện tại.
- Vị trí hiện tại gán cho vị trí tiếp theo.
- Vị trí tiếp theo cộng 4 byte (1 đơn vị trong PC).
- Nếu không lập trình đúng phần này thì Nachos sẽ bị vòng lặp gọi thực hiện system call này mãi mãi.

d. Cài đặt system call int ReadInt().

- Quy trình:
  + readint.c -> readint.o
  + start.c -> start.o
  + Tạo ra readint.coff => readint.
- Vào code-> userprog-> syscall.h :
  + **define SC_ReadInt 11.**
  + **int ReadInt().**
- Tạo một file **readint.c** trong test và viết nội dung của trả về một số kiểu int.
- Vào code-> test-> start.c, start.s viết hàm **ReadInt** bằng **MIPS.**
- Vào code-> test-> makefile.cc :
  + Khai báo **readint** (nếu không khai báo chương trình sẽ không chạy).
  + Viết code quá trình biên dịch tạo ra file **readint**.
- Vào code-> userprog-> exception.cc viết chương trình xử lý trong **"case SC_ReadInt"** :
  + Nhập trên console và trả về giá trị char* khi dùng hàm **Read** trong file **synchcons.h.**

+ Ta cần xây dựng đoạn code chuyển từ chuỗi kí tự (char*) sang số (int).
+ Ghi giá trị vừa tìm được vào thanh ghi $2.

- **Giải thích thêm :**
  + Trong file **readint.c**, do **ReadInt()** trả về một tham số nên giá trị nó được lưu vào thanh ghi $2.
  + Trong file start.c, ta đã lưu **SC_ReadInt** vào thanh ghi $2.
  + Trong file exception.c, trong **"case SC_ReadInt"** ta ghi số vừa nhập bằng cách ghi vào thanh ghi $2 (machine -> WriteRegister(2,sonhapvao)).

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

- Quy trình :
  + printint.c -> printint.o
  + start.c -> start.o
  + Tạo ra printint.coff => printint.
- Vào code-> userprog-> syscall.h :
  + **define SC_PrintInt 12.**
  + **void PrintInt(int x).**
- Tạo một file **printint.c** trong test và viết nội dung cần xử lý vào nó.
- Vào code-> test-> start.c, start.s viết hàm **PrintInt** bằng **MIPS**.
- Vào code-> test-> makefile.cc :
  + Khai báo **printint** (nếu không khai báo chương trình sẽ không chạy).
  + Viết code quá trình biên dịch tạo ra file **printint**.
- Vào code-> userprog-> exception.cc viết chương trình xử lý trong **"case SC_PrintInt"** :
  + Đọc giá trị của tham số truyền vào tại ô đầu tiên của thanh ghi $4 => Trả về int.
  + Ta cần xây dựng hàm chuyển từ số (int) sang mảng kí tự (char*).

+ Xuất giá trị ra màn hình bằng hàm **Write** trong file **synchcons.h.**

- **Giải thích thêm :**
  + Trong file **printint.c**, do **PrintInt(int n)** có 1 tham số truyền vào nên ta sẽ đọc giá trị ở thanh ghi $4.
  + Trong file start.c, ta đã lưu **SC_PrintInt** vào thanh ghi $2.
  + Trong file exception.c, trong "**case SC_PrintInt**" ta lấy ra được số cần xuất bằng cách đọc thanh ghi $4 (machine ->ReadRegister(4)).
- **Bắt ngoại lệ**:
  + Nhập vào số minInt là -2147483648 sẽ không bị lỗi.
  + Nhập số 0 sẽ không bị lỗi.

f. Cài đặt system call char ReadChar().

- Quy trình :
  + readchar.c -> readchar.o
  + start.c -> start.o
  + Tạo ra readchar.coff => readchar.
- Vào code-> userprog-> syscall.h :
  + **define SC_ReadChar 13.**
  + **char ReadChar().**
- Tạo một file readchar.c trong test và viết nội dung cần xử lý vào nó.
- Vào code-> test-> start.c, start.s viết hàm **ReadChar** bằng **MIPS**.
- Vào code-> test-> makefile.cc :
  + Khai báo **readchar** (nếu không khai báo chương trình sẽ không chạy).
  + Viết code quá trình biên dịch tạo ra file **readchar**.
- Vào code-> userprog-> exception.cc viết chương trình xử lý trong **"case SC_ReadChar"** :

+ Nhập trên console và trả về giá trị char* khi dùng hàm **Read** trong file **synchcons.h.**
+ Ta cần xây dựng hàm chuyển từ số (char*) sang kí tự (char).
+ Ghi giá trị vừa tìm được vào thanh ghi $2.

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

- Quy trình :
  + printchar.c -> printchar.o
  + start.c -> start.o
  + Tạo ra printchar.coff => printchar.
- Vào code-> userprog-> syscall.h :
  + **define SC_PrintChar 14.**
  + **void PrintChar(char).**
- Tạo một file **printchar.c** trong test và viết nội dung cần xử lý vào nó.
- Vào code-> test-> start.c, start.s viết hàm **PrintChar** bằng **MIPS**.
- Vào code-> test-> makefile.cc :
  + Khai báo **printchar** (nếu không khai báo chương trình sẽ không chạy).
  + Viết code quá trình biên dịch tạo ra file **printchar**.
- Vào code-> userprog-> exception.cc viết chương trình xử lý trong **"case SC_PrintChar"** :
  + Đọc giá trị của tham số truyền vào tại thanh ghi $4 => Trả về int.
  + Ta cần xây dựng hàm chuyển từ số (int) sang kí tự (char) => char*.
  + Ghi giá trị ra màn hình bằng hàm **Write** trong file **synchcons.h**.

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

- Quy trình :
  + readstring.c -> readstring.o
  + start.c -> start.o
  + Tạo ra readstring.coff => readstring.

- Vào code-> userprog-> syscall.h :
    + **define SC_ReadString 15.**
    + **void ReadString(char[],int).**
- Tạo một file readstring.c trong test và viết nội dung cần xử lý vào nó.
- Vào code-> test-> start.c, start.s viết hàm **ReadString** bằng MIPS.
- Vào code-> test-> makefile.cc :
    + Khai báo **readstring** (nếu không khai báo chương trình sẽ không chạy).
    + Viết code quá trình biên dịch tạo ra file **readstring**.
- Vào code-> userprog-> exception.cc viết chương trình xử lý trong **"case SC_ReadString"** :
    + Đọc 2 tham số truyền vào :
        o Đọc địa chỉ đầu tiên của thanh ghi $4 bởi vì đây là vị trí của chuỗi => trả về int.
        o Đọc giá trị của chiều dài tối đa của chuỗi từ thanh ghi $5 (do là tham số thứ 2).
    + Nhập trên console và trả về giá trị char* khi dùng hàm **Read** trong file **synchcons.h.**
    + Ghi giá trị vừa tìm được vào thanh ghi $2.
    + Dùng hàm **System2User**(int address,int limit,char buff[]) để trả về int (chính là độ dài của chuỗi) => Ghi chuỗi vào bộ nhớ.

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

- Quy trình :
    + printstring.c -> printstring.o
    + start.c -> start.o
    + Tạo ra printstring.coff => printstring.
- Vào code-> userprog-> syscall.h :
    + **define SC_PrintString 16.**
    + **void PrintString(char[]).**

- Tạo một file **printstring.c** trong test và viết nội dung cần xử lý vào nó.
- Vào code-> test-> start.c, start.s viết hàm **PrintString** bằng **MIPS**.
- Vào code-> test-> makefile.cc :
    + Khai báo **printstring** (nếu không khai báo chương trình sẽ không chạy).
    + Viết code quá trình biên dịch tạo ra file **printstring**.
- Vào code-> userprog-> exception.cc viết chương trình xử lý trong **"case SC_PrintString"** :
    + Đọc địa chỉ đầu tiên của thanh ghi $4 (chú ý: Đây chỉ là địa chỉ của kí tự đầu tiên trong chuỗi) => Trả về int.
    + Dùng hàm **User2System**(int address,int limit) để trả về char*.
    + Ghi giá trị ra màn hình bằng hàm **Write** trong file **synchcons.h**.

- **Sự khác nhau của tham số truyền vào là char[] và tham số truyền vào là int**
    - **Đối với tham số kiểu int:** Giá trị của nó được lưu tại ô đầu tiên (4 bytes) của thanh ghi $4.
    - **Đối với tham số kiểu char[]:** Giá trị của nó được phân chia lần lượt theo mỗi ô với mỗi ô là 4 kí tự (tương ứng 4 bytes) trong thanh ghi $4.

j. Viết chương trình help

- Quy trình :
    + help.c -> help.o
    + start.c -> start.o
    + Tạo ra help.coff => help.
- Tạo một file **help.c** trong test và viết nội dung cần xử lý vào nó (gọi lại hàm **PrintString(char[])**).
- Vào code-> test-> start.c, start.s viết hàm **Help** bằng **MIPS**.
- Vào code-> test-> makefile.cc :
    + Khai báo **help** (nếu không khai báo chương trình sẽ không chạy).
    + Viết code quá trình biên dịch tạo ra file **help**.

k. Viết chương trình ascii

- Quy trình :
  - + printascii.c -> printascii.o
  - + start.c -> start.o
  - + Tạo ra printascii.coff => printascii.
- Tạo một file **printascii.c** trong test và viết nội dung cần xử lý vào nó (Chạy vòng lặp từ 0->255 và in ra bằng cách gọi lại hàm **PrintChar(char)**).
- Vào code-> test-> start.c, start.s viết hàm **PrintASCII** bằng **MIPS**.
- Vào code-> test-> makefile.cc :
  - + Khai báo **printascii** (nếu không khai báo chương trình sẽ không chạy).
  - + Viết code quá trình biên dịch tạo ra file **printascii**.

l. Viết chương trình sort

- Quy trình :
  - + sort.c -> sort.o
  - + start.c -> start.o
  - + Tạo ra sort.coff => sort.
- Tạo một file **sort.c** trong test và viết nội dung cần xử lý vào nó :
  - + Nhập mảng dựa vào hàm **ReadInt()**.
  - + Viết thuật toán sắp xếp **BubbleSort**.
  - + Xuất mảng dựa vào hàm **PrintInt(int)**.
- Vào code-> test-> start.c, start.s viết hàm **Sort** bằng **MIPS**.
- Vào code-> test-> makefile.cc :
  - + Khai báo **sort** (nếu không khai báo chương trình sẽ không chạy).
  - + Viết code quá trình biên dịch tạo ra file **sort**.

- **Bắt ngoại lệ**: Nhập mảng trên 1 hàng.

  Vd: 1 45 65 123.

III. Tài liệu tham khảo:

a. **www.Stdio.vn**

b. **Www.Stackoverflow.com**

c. **Www.cplusplus.com**