

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**

**BỘ MÔN HỆ ĐIỀU HÀNH**

**-----🙞🕮🙜-----**

**BÁO CÁO**

**ĐỒ ÁN**



**Đề Tài : Syscall về hệ thống quản lý tập tin**

Sinh viên : **HUỲNH TRỌNG THOẠI - 1512551**

Sinh viên : **NGUYỄN ANH TUẤN - 1512636**

Giảng viên : **LÊ QUỐC HÒA**

Giảng viên : **CHUNG THÙY LINH**

Lớp : **TH2015/4**

**Mục lục**

[**I.** **Mức độ hoàn thành đồ án:** 3](#_Toc503481838)

[**II.** **Hiểu về thiết kế hệ điều hành** 5](#_Toc503481839)

[**1.** **UserMode (giới thiệu)** 5](#_Toc503481840)

[**2.** **SystemMode (giới thiệu)** 5](#_Toc503481841)

[**3.** **Giao tiếp với nền tảng MIPS** 5](#_Toc503481842)

[**III.** **Nội dung** 5](#_Toc503481843)

[**1.** **Cách cài đặt Nachos** 5](#_Toc503481844)

[**2.** **Thêm lớp syschcons vào nachos** 6](#_Toc503481845)

[**3.** **Exceptions và system calls** 6](#_Toc503481846)

[***a.*** ***Viết lại exception.cc*** 6](#_Toc503481847)

[***b.*** ***Viết lại cấu trúc điều khiển của chương trình để nhận các Nachos system calls.*** 7](#_Toc503481848)

[***c.*** ***Viết mã để tăng giá trị biến program counter.*** 7](#_Toc503481849)

[***d.*** ***Cài đặt system call int ReadInt().*** 7](#_Toc503481850)

[***e.*** ***Cài đặt system call void PrintInt(int number).*** 9](#_Toc503481851)

[***f.*** ***Cài đặt system call char ReadChar().*** 10](#_Toc503481852)

[***g.*** ***Cài đặt system call void PrintChar(char character)*** 11](#_Toc503481853)

[***h.*** ***Cài đặt system call void ReadString(char[] buffer,int length)*** 12](#_Toc503481854)

[***i.*** ***Cài đặt system call void PrintString(char[] buffer).*** 14](#_Toc503481855)

[***j.*** ***Viết chương trình help.*** 15](#_Toc503481856)

[***k.*** ***Viết chương trình ascii.*** 15](#_Toc503481857)

[***l.*** ***Viết chương trình sort*** 16](#_Toc503481858)

[**m.** ***Cài đặt system call int CreateFile(char\* name).*** 17](#_Toc503481859)

[**n.** ***Cài đặt system call OpenFileID OpenFile(char\* name, int type)*** 17](#_Toc503481860)

[***o.*** ***Cài đặt system call int CloseFile(OpenFileID id).*** 18](#_Toc503481861)

[***p.*** ***Cài đặt system call int ReadFile(char\* buffer, int charcount, OpenFileID id).*** 19](#_Toc503481862)

[***q.*** ***Cài đặt system call int Write (char\* buffer, int charcount, OpenFileID id)*** 19](#_Toc503481863)

[***r.*** ***Cài đặt system call int Seek(int pos, OpenFileID id).*** 20](#_Toc503481864)

[***s.*** ***Viết chương trình createfile*** 20](#_Toc503481865)

[***t.*** ***Viết chương trình echo*** 21](#_Toc503481866)

[***u.*** ***Viết chương trình cat.*** 21](#_Toc503481867)

[***v.*** ***Viết chương trình copy.*** 22](#_Toc503481868)

[***w.*** ***Viết chương trình reverse.*** 22](#_Toc503481869)

[***x.*** ***Viết chương trình test.*** 23](#_Toc503481870)

1. **Mức độ hoàn thành đồ án:**

|  |  |
| --- | --- |
| Chức năng chương trình | Mức độ |
| 1. Viết lại exception.cc. | Đã hoàn thành. |
| 1. Viết lại cấu trúc điều khiển của chương trình để nhận các Nachos system calls. | Đã hoàn thành. |
| 1. Viết mã để tăng giá trị biến program counter. | Đã hoàn thành. |
| 1. Cài đặt system call int ReadInt(). | Đã hoàn thành. |
| 1. Cài đặt system call void PrintInt(int number). | Đã hoàn thành. |
| 1. Cài đặt system call char ReadChar(). | Đã hoàn thành. |
| 1. Cài đặt system call void PrintChar(char character). | Đã hoàn thành. |
| 1. Cài đặt system call void ReadString(char[] buffer,int length). | Đã hoàn thành. |
| 1. Cài đặt system call void PrintString(char[] buffer). | Đã hoàn thành. |
| 1. Viết chương trình help. | Đã hoàn thành. |
| 1. Viết chương trình ascii. | Đã hoàn thành. |
| 1. Viết chương trình sort. | Đã hoàn thành. |
| 1. Cài đặt system call int CreatFile(char\* name). | Đã hoàn thành. |
| 1. Cài đặt system call OpenFileID OpenFile(char\* name,int type). | Đã hoàn thành. |
| 1. Cài đặt system call int Close(OpenFileID id). | Đã hoàn thành. |
| 1. Cài đặt system call int Read(char\* buffer, int charcount, OpenFileID id) | Đã hoàn thành. |
| 1. Cài đặt system call int Write(char\* buffer, int charcount, OpenFileID id) | Đã hoàn thành. |
| 1. Cài đặt system call int Seek(int pos, OpenFileID id). | Đã hoàn thành. |
| 1. Viết chương trình createfile. | Đã hoàn thành. |
| 1. Viết chương trình echo. | Đã hoàn thành. |
| 1. Viết chương trình cat. | Đã hoàn thành. |
| 1. Viết chương trình copy. | Đã hoàn thành. |
| 1. Viết chương trình reverse. | Đã hoàn thành. |
| 1. Viết chương trình test. | Đã hoàn thành. |
| **Tổng kết mức độ hoàn thành yêu cầu đồ án**: | **100%** |

1. **Hiểu về thiết kế hệ điều hành**
2. **UserMode (giới thiệu)**

Bao gồm các thành phần của chương trình người dùng (tập hợp mã máy), và vùng nhớ của các chương trình ứng dụng chạy trên hệ thống NachOs/MIPS.

1. **SystemMode (giới thiệu)**

Cung cấp các thành phần nhằm quản lý vùng nhớ, bộ nhớ của hệ điều hành NachOs, quản lí tiến trình và các SystemCall.

1. **Giao tiếp với nền tảng MIPS**

Bao gồm các thanh ghi, bộ nhớ chính và các cơ chế đọc ghi đơn giản, xử lí từng lệnh dựa trên tập lệnh MIPS.

1. **Nội dung**
2. **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
3. **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.
4. **Exceptions và system calls**
5. ***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.

1. ***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.

1. ***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.

1. ***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)).

1. ***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.

1. ***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.

1. ***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.

1. ***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 :
  + Đọc địa chỉ đầu tiên của thanh ghi $4 bởi vì đây là vị trí của chuỗi => trả về int.
  + 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.

1. ***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.

1. ***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.

1. ***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 label 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.

1. ***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.

1. ***Cài đặt system call int CreateFile(char\* name).***

* Vào code -> userprog -> syscall.h:
* define SC\_CreateFile 17.
* int CreateFile(char\*).
* Vào code -> test -> start.c, start.s tạo label CreateFile với nội dung là gán SC\_CreateFile vào thanh ghi số 2.
* Vào code -> userprog -> exception.cc viết chương trình xử lý trong “case SC\_CreateFile” :
* Đọc địa chỉ đầu tiên của thanh ghi thứ $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\*.
* Sử dụng biến con trỏ fileSystem ( được định nghĩa trong thread->system.h) để trỏ tới hàm Create trong class FileSystem với mục đích tạo mục file với kích thước là 0 byte. Nếu như tạo thành công thì ghi giá trị 0 vào thanh ghi $2, ngược lại ghi giá trị -1 vào thanh ghi $2.

1. ***Cài đặt system call OpenFileID OpenFile(char\* name, int type)***

* Trong openfile.h, ta khai báo cho 2 class OpenFile:

***+*** int \_type : xác định loại file

* \_type = 0: mở file đọc và ghi.
* \_type = 1: mở file chỉ đọc.
* \_type = 2: stdin.
* \_type = 3: stdout.

+ OpenFile(int sector, int type).

* Trong openfile.cc: khởi tạo các hàm đã được khai báo trong openfile.h
* Trong file filesys.h, ta khai báo cho 2 class FileSystem:
* OpenFile\* openf[10] : theo đề tạo ra tối đa 10 file và khai báo mảng con trỏ.
* int index : vị trí file.
* OpenFile\* Open(char\* name, int type): phục vụ cho việc mở các loại file khác nhau.
* FileSystem(bool format) : Khởi tạo danh sách file. Ban đầu ta khởi tạo 2 file rỗng stdin, stdout và mở 2 file ấy.
* ~FileSystem(): giải phóng bộ nhớ.
* Trong filesys.cc
* FileSystem::FileSystem(bool format) : sao chép nội dung FileSystem(bool format) vào cuối hàm.
* Khởi tạo các hàm còn thiếu trong filesys.h
* Vào code -> userprog -> syscall.h
* define SC\_OpenFile 18.
* OpenFileId OpenFile(char\* name, int type)

- Vào code -> test -> start.c, start.s tạo label OpenFile với nội dung là gán SC\_OpenFile vào thanh ghi số 2 và gọi syscall.

- Vào code -> userprog -> exception.cc viết chương trình xử lý trong “case SC\_OpenFile”.

1. ***Cài đặt system call int CloseFile(OpenFileID id).***

* Vào code -> userprog -> syscall.h:
* define SC\_CloseFile 19.
* int CloseFile(int openfileid);
* Vào code -> test -> start.c, start.s tạo label CloseFile với nội dung là gán SC\_CloseFile vào thanh ghi số 2 và gọi syscall.
* Vào code -> userprog -> exception.cc viết chương trình xử lý trong “case SC\_CloseFile”
* ***Bắt ngoại lệ:***
* Mở 8 file, tới file 9 sẽ thông báo lỗi.
* Đóng file bất kì xong mở lại không bị lỗi.

1. ***Cài đặt system call int ReadFile(char\* buffer, int charcount, OpenFileID id).***

* Trong openfile.h, ta khai báo int Read(char\* into, int numBytes) cho 2 class OpenFile.
* Trong openfile.c, khởi tạo các hàm đã được khai báo trong openfile.h.
* Vào code -> userprog ->syscall.h
* define SC\_ReadFile 20.
* int ReadFile(char\* buffer, int charcount, OpenFileID id).
* Vào code -> test -> start.c, start.s tạo label ReadFile với nội dung là gán SC\_ReadFile vào thanh ghi số 2 và gọi syscall.
* Vào code -> userprog -> exception.cc viết chương trình xử lý trong “case SC\_ReadFile”.

1. ***Cài đặt system call int Write (char\* buffer, int charcount, OpenFileID id)***

* Trong openfile.h, ta khai báo

int Write(char\* into, int numBytes) cho 2 class OpenFile.

* Vào code -> userprog ->syscall.h
* define SC\_WriteFile 21.
* int WriteFile(char\* buffer, int charcount, OpenFileID id).
* Vào code -> test -> start.c, start.s tạo label WriteFile với nội dung là gán SC\_WriteFile vào thanh ghi số 2 và gọi syscall.
* Vào code -> userprog -> exception.cc viết chương trình xử lý trong “case SC\_WriteFile”.

1. ***Cài đặt system call int Seek(int pos, OpenFileID id).***

* Trong openfile.h, khai báo int Seek(int pos) cho 2 class OpenFile.
* Trong openfile.c, khởi tạo hàm int Seek(int position).
* Vào code -> userprog ->syscall.h
* define SC\_Seek
* int SeekFile(int pos, OpenFileID FileID).
* Vào code -> test -> start.c, start.s tạo label SeekFile với nội dung là gán SC\_SeekFile vào thanh ghi số 2 và gọi syscall.
* Vào code -> userprog -> exception.cc viết chương trình xử lý trong “case SC\_SeekFile”.

1. ***Viết chương trình createfile***

* Quy trình :
* createfile.c -> createfile.o
* start.c -> start.o
* Tạo ra createfile.coff => createfile.
* Tạo một file createfile**.c** trong test và viết nội dung cần xử lý vào nó (gọi syscall)
* Vào code-> test-> makefile.cc :
* Khai báo createfile (nếu không khai báo chương trình sẽ không chạy).
* Viết code quá trình biên dịch tạo ra file createfile.

1. ***Viết chương trình echo***

* Quy trình :
* echo.c -> echo.o
* start.c -> start.o
* Tạo ra echo.coff => echo.
* Vào code-> test-> makefile.cc :
* Khai báo echo (nếu không khai báo chương trình sẽ không chạy).
* Viết code quá trình biên dịch tạo ra file echo.
* Tạo một file echo**.**c trong test và viết nội dung cần xử lý vào nó.

1. ***Viết chương trình cat.***

* Quy trình :
* cat.c -> cat.o
* start.c -> start.o
* Tạo ra cat.coff => cat.
* Vào code-> test-> makefile.cc :
* Khai báo cat (nếu không khai báo chương trình sẽ không chạy).
* Viết code quá trình biên dịch tạo ra file cat.
* Tạo một file cat**.**c trong test và viết nội dung cần xử lý vào nó.

1. ***Viết chương trình copy.***

* Quy trình :
* copy.c -> copy.o
* start.c -> start.o
* Tạo ra copy.coff => copy.
* Vào code-> test-> makefile.cc :
* Khai báo copy (nếu không khai báo chương trình sẽ không chạy).
* Viết code quá trình biên dịch tạo ra file copy.
* Tạo một file copy**.**c trong test và viết nội dung cần xử lý vào nó.

1. ***Viết chương trình reverse.***

* Quy trình :
* reverse.c -> reverse.o
* start.c -> start.o
* Tạo ra reverse.coff => reverse.
* Vào code-> test-> makefile.cc :
* Khai báo reverse (nếu không khai báo chương trình sẽ không chạy).
* Viết code quá trình biên dịch tạo ra file reverse.
* Tạo một file reverse**.**c trong test và viết nội dung cần xử lý vào nó.

1. ***Viết chương trình test.***

* Quy trình :
* test.c -> test.o
* start.c -> start.o
* Tạo ra test.coff => test.
* Vào code-> test-> makefile.cc :
* Khai báo test (nếu không khai báo chương trình sẽ không chạy).
* Viết code quá trình biên dịch tạo ra file test.
* Tạo một file test**.**c trong test với nội dung: Mở 10 file, đóng 1 file, mở file bất kì sẽ không bị lỗi.