1. OS : Là 1 chương trình nhưng nó chạy đầu tiên của hệ thống, nó sở hữu và có toàn quyền quyết định các thành phần khác của hệ thống

* Tính năng chính của OS : có nhiều các tính năng chính, k có gì nổi trội so với những cái khác

1. Kiến trúc hệ điều hành Linux
2. Có 2 chế độ : console và graphic
3. Console : chỉ có Terminal đen trắng , k có chuột, phải dùng bàn phím, tuy nhiên rất tiết kiệm tài nguyên
4. Graphic : Có chuột, thao tác trực quan hơn.
5. Chương trình : Các file Binary đc buil từ source code và nằm trên hệ thống

Tiến trình : Là các chương trình nhưng đã đc Load vào hệ thống. Chúng tiêu thụ tài nguyên của CPU

1. Đa nhiệm : Các tác vụ tưởng chừng như chạy cùng 1 lúc nhưng k phải vậy, thời gian thực hiện 1 tác vụ rất nhanh, và ta sẽ chen những tác vụ khác nhau vào thời gian chờ của mỗi tác vụ, thứ tự thực hiện ta cài đặt được

* Có thể chạy đa nhiệm trên 1 core đc , tùy thuộc vào bộ lập lịch

1. File system trong Linux

Thư mục Roots là tổng thể ( như kiểu This PC trong windows ), trong Roots chứa các Folder khác , và từ Roots có thể băm ra các Folder khác

1. Quá trình biên dịch 1 chương trình C

* Giai đoạn 1 : Tiền xử lý
* Loại bỏ comment
* Mở rông các Macro
* Mở rộng các Include File
* Biên dịch câu điều kiện
* Kết quả thu được là 1 file “.i”

VD : gcc -E main.c -o main.i

* Giai đoạn 2 : dịch ngôn ngữ bậc cao sang asm :
* Mã nguồn từ file .i sẽ được chuyển thành file .s

VD : gcc -S main.i -o main.s

* Giai đoạn 3: Dịch file .asm sang mã máy ( Assembly):
* File .s ở giai đoạn trước được sử dụng tiếp trong giai đoạn này
* Thông qua Assembler, output mà chúng ta thu được là file .o

VD : gcc -c main.s -o main.o

* Giai đoạn 4: Giai đoạn Linker
* Mỗi file .o thu được ở giai đoạn 3 là 1 phần của chương trình
* Ở giai đoạn Linker sẽ liên kết chúng để thu được 1 file thực thi hoàn chỉnh

VD : gcc -o main main.o

Bài 1 : Viết File .c trên Linux

# Bật terminal lên :

Gõ “vi ….” : … là tên file muốn đặt

Sau đấy console sẽ hiện lên để soạn thảo, lưu ý trong console không dùng được chuột .

# Gõ “i” và bắt đầu viết chương trình

VD : viết chương trình hello world

# Sau khi viết xong, nhấn ESC và shift + insert , sau ấy nhấn shift + “;” rồi gõ “wq” để lưu chương trình

# Gõ gcc “tên chương trình “ để bulit chương trình

# Gõ ls để check

# Khi này, mặc định file vừa viết mặc định ra “a.out”, ta phải xóa file này đi bằng cách gõ “rm a.out”

# Đổi tên ( VD tên cũ là hello.c và tên mới là hello ) bằng cách gõ

# gcc hello.c -o hello

# Sau khi đổi tên, ta chạy chương trình bằng cách gõ “./hello” ( hello là tên file mới “

# Ta có thể thay từ bước 4 đến 7 bằng lệnh : gcc -o hello hello.c

* Các ký tự đặc biệt trong Linux
* Bộ soạn thảo vi chạy ở hai chế độ khác nhau:

+ Chế độ dòng lệnh command mode, những gì được gõ vào sẽ được hiểu như là lệnh của vi. Vi có rất nhiều lệnh như: tìm kiếm, thay thế, xóa, lưu tâp tin…

+ Chế độ nhập văn bản insert mode, những gì được gõ vào được hiểu là nội dung của tập tin đang soạn thảo.

* Khi bắt đầu sử dụng lệnh**vi**, **vi** mặc định ở command mode. Ấn phím lệnh**i**, **a**, **o** hoặc Inserrt từ chế độ command mode để chuyển sang insert mode.

**i**để thêm ký tự trước con trỏ,

**a** để thêm ký tự sau con trỏ,

**o** để chèn thêm dòng mới sau con trỏ.

* Ấn Esc để chuyển đổi qua lại từ command mode với insert mode.
* Một số lệnh với vi:

**:set nu** hiện thị số dòng

**:set nonu** bỏ hiện thị số dòng

**Sử dụng phím mũi tên hoặc các phím h,l,j,k để dịch trái, phải, lên, xuống**

**:1 để nhảy đến dòng đầu tiên của file**

**:n nhảy đến dòng n**

**$ nhảy xuống cuối dòng**

**:$ nhảy đến dòng cuối của file.0 nhảy về đầu dòng**

**:0 nhảy về dòng đầu tiên của file.**

**dd xóa một dòng hiện tại**

**ndd xóa n dòng**

**/ hay ? để tìm kiếm**

**:w! lưu tập tin**

**:x! lưu tập tin và thoát**

**:wq ZZ lưu tập tin và thoát**

**:q! không lưu và thoát**

Các command trong Linux Terminal

* ls : check các thành phần
* cd …./ : di chuyển đến folder …
* cd : trở về Rootfs chung
* cat ….. : view file …. ( Muốn cat được File thì phải cd đến folder chứa file đó trước)
* touch : tạo 1 file mới
* Viết file trực tiếp từ terminal : echo “……” > ‘Ten file”
* pwd : check vị trí đang đứng
* rm -rf : xóa hẳn file
* cd ../ : lùi lại 1 thư mục
* mkdir ….: tao 1 thu muc moi
* move ….. “ten thu muc’/ : di chuyen file … den thu muc
* để chạy 1 chương trình C ( bao gồm thu viện tự tạo ) , ta cần xác định thư mục chứa thư viện ( tạo 1 folder lưu bằng lệnh mkdir và move ), sau đấy dùng lệnh

gcc -o example main.c hello.c -I include/

Với example và file chạy ( màu xanh lục ), main.c và hello.c là các file source, include là thư mục chứa file header

MakeFile

1. Makefile là 1 script bên trong có chứa các thông tin :

* Cấu trúc của 1 project (file,dependency)
* Các command line dùng để tạo-hủy file
* Chương trình make sẽ đọc ND trong makefile và thực thi nó

1. Cấu trúc của 1 lệnh MakeFile



* Cách sử dụng Makefile
* VD : ta chạy 1 chương trình có 3 file là main.c hello.c và hello.h, trong đó hello.h là file header
* Ta lần lượt tạo và viết 3 file trên bằng lệnh vim
* Để chạy , ta sử dụng lệnh : “gcc -o exam main.c hello.c -I .”
* File chạy là exam, để xóa file exam , ta dùng lệnh “rm -rf exam”

Đó là quy trình viết và chạy 1 file

* Để sử dụng Makefile :

VD :

all:

gcc -o exam hello1.c main1.c -I .

clear:

rm -rf exam

* Target là all và clear
* Để chạy lệnh “gcc -o exam main.c hello.c -I .”, thay vì gõ cả câu lệnh đó, ta sử dụng lệnh “make all “ ( với all là target trong makefile)
* Để chạy lệnh “rm -rf exam”, thay vì gõ cả câu lệnh đó, ta sử dụng lệnh “make clear “ ( với all là target trong makefile)
* Nguyên lý hoạt động của 1 makefile

Sau khi gọi lệnh make target , kiểm tra xem các dependency có mặt hay k, nếu có thì thực hiện action, nếu không thì báo lỗi

1. Biến trong makefile

* Cách gán 1 : đệ quy
* Vd : var1 = $(var) -> bất cứ khi nào var thay đổi giá trị thì var2 cũng thay đổi giá trị
* **Cách gán 2 : Trực tiếp**
* VD : var2 := $(var) -> gán trực tiếp giá trị của var2 là var , giá trị của var2 được giữ nguyên cho đến khi nó được gán lại
* Cách gán 3 :
* VD : var3 ?= $(var)-> kiếm tra xem var3 có giá trị hay chưa , nếu có rồi thì k gán lại nữa, nếu chưa có thì gán theo cách 1

Lưu ý : để lấy giá trị 1 biến trong Linux , ta dùng $(ten\_bien), để command 1 dòng, ta dùng #

1. Nếu trong 1 folder có nhiều makefile , để chạy makefile chỉ định , dùng lênh “make -f ten\_file”
2. Để chỉ định đến 1 rule trong 1 makefile : make -f ten\_makefile ten\_rule
3. Để ngăn ngừa lỗi có 1 file nào đó trùng tên rule trong makefile, ở đầu make file, ta dùng biến PHONY : .PHONY: liệt\_kê\_tên\_rule
4. Các biến đặc biệt :

VD : test: hello.c hello.h

@echo $@

@echo $<

@echo S^

* $@ : giá trị đầu bên trái của dấu :
* $< giá trị bên phải đầu tiên sau dấu :
* $^ toàn bị bên phải dấu :

Khi chạy chương trình rule test trên -> kết quả là

test

hello1.c

hello1.c hello1.h

1. Cách viết các rule

VD : ta có e file main.c check.c và check.h trong 1 folder

Ta viết 1 rule để build file main.c

Ta viết như sau:

CC := gcc

CCR := -I .

buil: main.o check.o

$(CC) -o $@ main.o check.o $(CCR)

Kết quả là :

gcc -c -o main.o main.c

gcc -c -o check.o check.c

gcc -o buil main.o check.o -I .

* File main.o là file build ra của main.c
* File check.o là file build ra của check.c
* Sau khi build ra 2 file trên, chương trình sẽ dùng chính chúng để build file buil

Tuy nhiên, nếu chỉ viết như trên, khi thay đổi trong file .h thì chương trình k detect ra được, nên ta cần viết thêm rule bằng cách

HFILE := check.h

%.o: %.c $(HFILE)

$(CC) -c -o $(@) $<

# Phân chia thư mục trong 1 folder làm việc

Gồm các folder sau

1. Bin : chứa file thực thi (màu xanh lục )
2. Inc : Chứa các file Header
3. Lib : Gồm StaticLib và ShareLib
4. Obj: Gồm các File .o
5. Src : gồm các thư viện .c
6. File main.c
7. Makefile
8. Readme.md

VD : Ta có chương trình gồm file main.c và hello.c , hello.h

* Các bước tạo Static Lib và chạy chương trình
* B1: Tạo các file .o và nhét nó vào thư mục obj, các câu lệnh sử dụng

gcc -c main.c -o obj/main.o -I inc/

gcc -c src/hello.c -o obj/hello.o -I inc/

* B2 : Tạo ra static Lib và nhét nó vào thư mục Lib, câu lệnh sử dụng

ar rcs lib/static/libhello.a obj/hello.o

* B3 : tạo file thực thi và nhét nó vào trong thư mục bin bằng cách Link các file .o lại với nhau, câu lệnh sử dụng :

gcc obj/main.o -L lib/static -l hello -o bin/exam

File Trong Linux

# Tổng quan

* Regular File : Các file thông thường như text file, exe file
* Dirẻctories File : file chứa danh sách các file khác
* Character Devices File : File đại diện cho các thiết bị không có địa chỉ vùng nhớ
* Block Device File: File đại diện cho các thiết bị có địa chỉ vùng nhớ
* Link file : file đại diện cho 1 file khác
* Socket file : file đại diện cho 1 socket
* Pipe File : file đại diện cho 1 pipe
* Để lấy ra thông tin toàn bộ file , ta dùng ls -l
* Số hard link của file : số file cùng trỏ đến 1 vùng nhớ ( ta có thể kiểm tra bằng lệnh ls -l)
* Loại file và quyền của file ( kiểm tra bằng ls -l)
* ở cột ngoài cùng bên trái , nếu là dấu “-“ thì là regular file
* nếu là chữ “d” thì là Direct file
* chứ “b” là Block Device file

…

* quyền của file

User permission : r-w-x: 111(7)

Group permission: r-w-x: 111(7)

Others permission: r-w-x: 111(7)

* để sửa quyền của file -> lệnh chmod :

VD : chmod o-r : lấy đi quyền read của Others

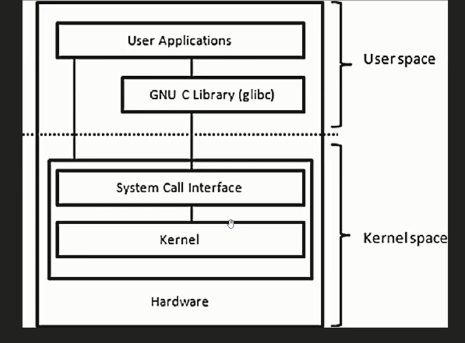
Chmod u+w : thêm quyền write cho User

Để thay đổi User , ta dùng : sudo chown ten\_user ten\_file

Để thay đổi Group , ta dùng : sudo chown ten\_user:ten\_group ten\_file

# Thao tác với file

Các system call cơ bản :



1. open :

**int open(const char \****pathname***, int** *flags***);**

**int open(const char \****pathname***, int** *flags***, mode\_t** *mode***);**

* \*const char \*pathname : tên file cần tạo , VD : test.txt
* Flags : các cờ đc quy định trước:

VD : O\_RDONLY, O\_WRONLY, or O\_RDWR. These request opening the file read-only, write-only, or read/write

* O\_CREAT : nếu file chưa có sẵn thì tạo ra nó
* Mode : tạo quyền cho file , VD : 0666
* Giá trị trả về của hàm open là 1 File Descriptor

1. Write

**ssize\_t write(int** *fd***, const void** *buf***[.***count***], size\_t** *count***);**

* **Fd là** giá trị của File Descriptor của file cần ghi
* **Buff** là chuỗi cần ghi vào
* Count là kích thước chuỗi ký tự cần ghi vào
* Hàm này trả về kích thước thực tế ghi đc vào file

1. Read
2. Lseek : di chuyen con tro file

**off\_t lseek(int** *fd***, off\_t** *offset***, int** *whence***);**

* Fd : ….
* Offset : Giá trị bù vào, nếu offset bằng 2 thì con trỏ file sẽ ở vị trí 2
* Whence :

SEEK\_SET

Offset tính từ đầu bên trái

SEEK\_CUR

Offset tính từ vị trí hiện tại

SEEK\_END

Offset tính từ cuối file

1. Close: truyen vao fd

Giải phóng tài nguyên cho process

# Quản lý file trong Linux

* Kernel điều khiển việc tương tác giữa các tiến trìn và file thông qua 3 bảng:

+ File descriptor table

+ Open file table

+ I-node table