HỌC VIỆN KỸ THUẬT MẬT MÃ

KHOA CÔNG NGHỆ THÔNG TIN

TÀI LIỆU

LẬP TRÌNH NHÂN LINUX

Giảng viên: **TS. Phạm Văn Hưởng**

Sinh viên thực hiện: Hoàng Nghĩa Thái - CT040142

Nguyễn Minh Trường - CT040150

Trần Duy Nghĩa - CT040134

Hà Nội, 03-2023

Mục lục

I. SHELL 4

I.1 Quản lý tệp tin 4

I.1.1 Chức năng 4
I.1.2 Mô tả 4

I.2 Lập lịch tác vụ 7

I.2.1 Chức năng 7
I.2.2 Thực thi 7

1.1. Auto install and uninstall 10

I.2.3 Chức năng 10
I.2.4 Mô tả 10

I.3 Thiết lập ngày giờ 12

I.3.1 Chức năng 12
I.3.2 Mô tả 12

II. Code C 15

II.1 Quản lý tệp tin 15

II.1.1 Chức năng 15
II.1.2 Mô tả 15

II.2 Quản lý tiến trình 17

II.2.1 Chức năng 17
II.2.2 Mô tả 17

II.3 Quản lý Socket 18

II.3.1 Chức năng 18
II.3.2 Mô tả 18

II.4 Quản lý Network 25

II.4.1 Chức năng 25
II.4.2 Mô tả 25

III. Kernel Module 27

III.1 Mô tả 27

III.2 Biên dịch và cài đặt vào nhân 27

III.3 Kết luận 28

IV. Dịch thuật-Quản lý mạng 30

IV.1 Tổng quan về quản lý mạng 30

IV.2 Tổng quan về triển khai socket: 31

IV.3 Giao thức và các mối liên hệ với socket: 31

IV.4 Các giai đoạn xử lý mạng: 31

IV.5 Bảng định tuyến 32

IV.6 Cơ sở dữ liệu chính sách định tuyến 32

IV.7 Cơ sở dữ liệu thông tin chuyển tiếp 33

IV.8 Netfilter 34

IV.9 Gói mạng / skbs (struct sk_buff) 35

IV.10 Thiết bị mạng 35

IV.11 Kỹ thuật tăng tốc phần cứng và phần mềm 35

I. SHELL

I.1 Quản lý tệp tin

I.1.1 Chức năng

- Hiển thị danh sách tệp tin
- Tạo tệp tin
- Xoá tệp tin
- Đọc nội dung tệp tin.
- Sửa tệp tin.
- Đổi tên tệp tin
- Sao chép tệp tin

I.1.2 Mô tả

I.1.2.1 Hiển thị danh sách tệp tin

- Hiển thị toàn bộ tệp tin có sẵn

```
root@ubuntu:/home/hoangthaifc01/Desktop/shell# bash prj.sh list
total 16
drwxrwxr-x 2 hoangthaifc01 hoangthaifc01 4096 Mar 26 23:30 .
drwxr-xr-x 9 hoangthaifc01 hoangthaifc01 4096 Mar 25 21:28 ..
-rw-r--r-- 1 root          root             9 Mar 26 23:28 n5.sh
-rw-r--r-- 1 root          root           890 Mar 26 23:30 prj.sh
```

Hình I.1 Hiển thị các tệp tin có sẵn

```
if [ "$1"  = "list" ];
then
        ls -la
fi
if [ "$1" = "create" ];
```

Hình I.2 câu lệnh hiển thị tệp tin

I.1.2.2 Tạo tệp tin

- Tạo tệp tin với tên người dùng nhập vào

```
root@ubuntu:/home/hoangthaifc01/Desktop/shell# ls
n1.sh  n2.sh  n3.sh  n4.sh
root@ubuntu:/home/hoangthaifc01/Desktop/shell# bash n2.sh
Enter file name you want create:
n1
root@ubuntu:/home/hoangthaifc01/Desktop/shell# ls
n1.sh  n1.txt  n2.sh  n3.sh  n4.sh
root@ubuntu:/home/hoangthaifc01/Desktop/shell# bash n2.sh
Enter file name you want create:
n1
this name exists
```

Hình I.3 Minh hoạ chức năng tạo tệp tin

- Kiểm tra có tên tệp hay chưa và tạo

```
if [ "$1" = "create" ];
then
        if [ -e $2 ];
        then
                echo "this file exists"
        else
                touch $2
        fi
fi
```

Hình I.4 Đoạn mã chức năng tạo tệp tin

I.1.2.3 Xoá tệp tin

- Xoá theo tên tệp tin người dùng nhập vào

```
root@ubuntu:/home/hoangthaifc01/Desktop/shell# ls
n1.sh  n1.txt  n2.sh  n3.sh  n4.sh
root@ubuntu:/home/hoangthaifc01/Desktop/shell# bash n3.sh
Enter file name you want remove:
n1
root@ubuntu:/home/hoangthaifc01/Desktop/shell# ls
n1.sh  n2.sh  n3.sh  n4.sh
root@ubuntu:/home/hoangthaifc01/Desktop/shell# bash n3.sh
Enter file name you want remove:
n1
File not exists
```

- Đoạn mã: kiểm tra có tồn tại tệp tin hay không và xác thực yêu cầu xoá

```
if [ "$1" = "remove" ];
then
        if [ -e $2 ];
        then
                rm -rf $2
                echo "Removed"
        else
                echo "this file not exists"
        fi
fi
```

Hình I.9 Đoạn mã chức năng xoá tệp tin

I.1.2.4 Đọc nội dung tệp tin

```
root@ubuntu:/home/hoangthaifc01/Desktop/shell# bash prj.sh show thai.txt
cham lai mot giay de em nhin lai
nguoi giong nhu em mo khong ton tai
ngay thang tuoi xanh dan phai
em van chua yeu duoc ai
chi con minh anh sao cu kho dai
chang giong em mo nhung van o lai
chang biet em co bang ai
nhu the dung hay la sai
```

```
if [ "$1" = "show" ];
then
        if [ -e $2 ];
        then
                cat $2
        else
                echo "this file not exists"
        fi
fi
```

I.1.2.5 Sửa tệp tin

```
root@ubuntu:/home/hoangthaifc01/Desktop/shell# bash prj.sh edit thai.txt
```

```
if [ "$1" = "edit" ];
then
        if [ -e $2 ];
        then
                vim $2
        else
                echo "this file not exists"
        fi
fi
```

I.1.2.6 Đổi tên tệp tin

```
root@ubuntu:/home/hoangthaifc01/Desktop/shell# ls
n5.sh  prj.sh  thai.txt
root@ubuntu:/home/hoangthaifc01/Desktop/shell# bash prj.sh rename thai.txt nope.
txt
root@ubuntu:/home/hoangthaifc01/Desktop/shell# ls
n5.sh  nope.txt  prj.sh
```

```
if [ "$1" = "rename" ];
then
        if [ -e $2 ];
        then
                if [ $2 = "" -o $3 = "" ];
                then
                        echo "please enter old name and new name"
                else
                        mv $2 $3
                fi
        else
                echo "this file not exists"
        fi
fi
```

I.1.2.7 Sao chép tệp tin

```
root@ubuntu:/home/hoangthaifc01/Desktop/shell# ls
n5.sh  nope.txt  prj.sh
root@ubuntu:/home/hoangthaifc01/Desktop/shell# cat n5.sh
hadhasdh
root@ubuntu:/home/hoangthaifc01/Desktop/shell# bash prj.sh copy n5.sh n6.sh
root@ubuntu:/home/hoangthaifc01/Desktop/shell# ls
n5.sh  n6.sh  nope.txt  prj.sh
root@ubuntu:/home/hoangthaifc01/Desktop/shell# cat n6.sh
hadhasdh
```

```
if [ "$1" = "copy" ];
then
        if [ -e $2 ];
        then
                if [ $2 = "" -o $3 = "" ];
                then
                        echo "please enter old name and new name"
                else
                        cp $2 $3
                fi
        else
                echo "this file not exists"
        fi
fi
```

I.2 Lập lịch tác vụ

I.2.1 Chức năng

- Tạo lập lịch
- Hiển thị danh sách lập lịch
- Thay đổi lập lịch
- Xoá lập lịch

I.2.2 Thực thi

I.2.2.1 Tạo lập lịch

Chức năng	Create

Mô tả	Chức năng create sẽ tiến hành tạo lập lịch tiến trình tuỳ thuộc vào thông tin người dùng nhập vào.
Cách thực hiện	Sử dụng chương trình crontab để tạo lập lịch tiến trình

```
if [[ $command = "create" ]]
        then
      echo "Xin hay nhap doan lenh ban muon lap lich"
      read order
      echo "Gio ban se nhap gio va phut de chay."
      echo "Hay nhap gio"
      read hour
      echo "Hay nhap phut"
      read min
      echo "Gio ban can nhap ngay tac vu chay hang thang."
      echo "Neu ban muon chay tac vu hang ngay, hay nhap *"
      echo "Neu ban muon chon ngay rieng le, hay nhap cac ngay ban mong muon, phan cach boi dau \",\""
      read day
      echo "Gio ban can nhap thang tac vu ma tac vu chay."
      echo "Neu ban muon tac vu chay hang thang, hay nhap *"
      echo "Neu ban muon chon thang rieng le, hay nhap cac thang ban mong muon, phan cach boi dau \",\""
      read month
      echo "Gio ban can nhap ngay tac vu chay hang tuan."
      echo "Neu ban muon chay tac vu hang ngay, hay nhap *"
      echo "Neu ban muon chon thu rieng le, hay nhap cac thu ban mong muon, phan cach boi dau \",\""
      echo "Xin nho rang gia tri 0 la Chu nhat"
      read dow
        echo "ban muon thuc hien thao tac nay khong? (y/n)"
      read y
      if [[ $y = 'Y' ]]
```

```
        then
            echo "$min $hour $day $month $dow $order" >> "input.txt"
            crontab -l -u $LOGNAME | cat - "input.txt" | crontab -u
$LOGNAME -

    else echo "Aborted\n"
    fi
fi
```

I.2.2.2 Hiển thị lập lịch

Chức năng	List
Mô tả	Chức năng list sẽ tiến hành liệt kê những tiến trình đã được lập lịch bằng crontab.
Cách thực hiện	Sử dụng chương trình crontab để hiển thị danh sách các tiến trình được lập lịch.

```
if [[ $command = "list" ]]
    then
        crontab -l
fi
```

I.2.2.3 Xoá lập lịch

Chức năng	Create
Mô tả	Chức năng create sẽ tiến hành tạo lập lịch tiến trình tuỳ thuộc vào thông tin người dùng nhập vào.
Cách thực hiện	Sử dụng chương trình crontab để tạo lập lịch tiến trình

```
if [[ $command = "remove-all" ]]
        then
        crontab -r
fi
```

I.2.2.4 Thay đổi lập lịch

Chức năng	Edit
Mô tả	Chức năng edit sẽ cho phép thực hiện chỉnh sửa những tiến trình đã được lập lịch.

Cách thực hiện	Sử dụng chương trình crontab và text editor để thực hiện chỉnh sửa nội dung lập lịch.

```
if [[ $command = "edit" ]]
        then
         crontab -e
fi
```

1.1. Auto install and uninstall

I.2.3 Chức năng

- Install
- Uninstall

I.2.4 Mô tả

I.2.4.1 Install

Chức năng	Install
Mô tả	Chức năng install sẽ tiến hành đọc nội dung của file input.txt và cài đặt tất cả các package được liệt kê ở trong file.
Cách thực hiện	Đọc từng dòng tên các phần mềm được liệt kê ở trong file input.txt, sau đó dùng lệnh sudo apt-get với cờ -y để tiến hành cài đặt tất cả các gói.

```
if [[ $command = "install" ]]
   then
      if [ `whoami` != "root" ]
         then
         echo 'Ban can phai chay lenh sudo de thay doi he thong'
         exit $E_NOTROOT
      fi

      cat input.txt | xargs sudo apt-get -y install
fi
```

I.2.4.2 Uninstall

Chức năng	Uninstall
Mô tả	Chức năng uninstall sẽ tiến hành đọc nội dung của file input.txt và gỡ cài đặt tất cả các chương trình được liệt kê
Cách thực hiện	Đọc từng dòng tên các phần mềm được liệt kê ở trong file input.txt, sau đó dùng lệnh sudo apt-get purge với cờ -y để tiến hành cài đặt tất cả các gói.

```
if [[ $command = "uninstall" ]]
   then
      if [ `whoami` != "root" ]
         then
         echo 'Ban can phai chay lenh sudo de thay doi he thong'
         exit $E_NOTROOT
      fi

      cat input.txt | xargs sudo apt-get -y purge
fi
```

I.3 Thiết lập ngày giờ

I.3.1 Chức năng

- Hiển thị thông tin
- Thay đổi ngày
- Thay đổi thời gian
- Tự động cập nhật

I.3.2 Mô tả

I.3.2.1 Hiển thị thông tin

Chức năng	Info
Mô tả	Chức năng info sẽ hiển thị thời gian ngày tháng của hệ thống cũng như múi giờ.

Cách thực hiện	Sử dụng chương trình date để thực hiện xem giờ của hệ thống.

```
if [[ $command = "info" ]]
   then TZ="Asia/Ho_Chi_Minh" date
fi
```

I.3.2.2 Thay đổi thời gian, ngày giờ

Chức năng	Hour
Mô tả	Chức năng hour được sử dụng để cập nhật lại thời gian giờ phút giây ở trong hệ thống.
Cách thực hiện	Sử dụng chương trình date để thực hiện sửa giờ ở trong hệ thống.

Chức năng	Hour
Mô tả	Chức năng hour được sử dụng để cập nhật lại thời gian giờ phút giây ở trong hệ thống.
Cách thực hiện	Sử dụng chương trình date để thực hiện sửa giờ ở trong hệ thống.


```
if [[ $command = "set-hour" ]]
   then
      if [ `whoami` != "root" ]
         then
         echo 'Ban can phai chay lenh sudo de thay doi thoi gian'
         exit $E_NOTROOT
      fi
      echo "Xin hay nhap gio theo dinh dang hh:mm:ss?"
      read new_time
```

```
        date +%T -s $new_time
fi

if [[ $command = "set-date" ]]
   then
        if [ `whoami` != "root" ]
           then
           echo 'Ban can phai chay lenh sudo de thay doi thoi gian'
           exit $E_NOTROOT
        fi
        echo "Xin hay nhap ngay theo dinh dang yyyymmdd?"
        read new_time
        date +%Y%m%d -s $new_time
fi
```

I.3.2.3 Tự động cập nhật

Chức năng	Automatic
Mô tả	Chức năng automatic được sử dụng để tự động cập nhật lại thời gian của hệ thống
Cách thực hiện	Sử dụng chương trình npdate để thực hiện tự động cập nhật lại thời gian của hệ thống.

```
if [[ $command = "automatic" ]]
   then
        if [ `whoami` != "root" ]
           then
           echo 'Ban can phai chay lenh sudo de thay doi thoi gian'
           exit $E_NOTROOT
        fi
        echo "Thoi gian dang duoc cap nhat theo server cua Microsoft..."
        ntpdate -4 time.windows.com
```

fi

II. Code C

II.1 Quản lý tệp tin

II.1.1 Chức năng

- Tạo tệp tin.
- Đọc nội dung tệp tin.
- Sửa nội dung tệp tin.
- Xoá tệp tin.
- Thay đổi quyền tệp tin (đổi quyền, đổi chủ sở hữu, đổi nhóm sở hữu).

II.1.2 Mô tả

II.1.2.1 Tạo tệp tin

- Tạo mới tệp tin với nội dung được nhập vào từ bàn phím.

```
root@ubuntu:/home/hoangthaifc01/Desktop/manager# ls
file  file.c
root@ubuntu:/home/hoangthaifc01/Desktop/manager# ./file create thai.txt
Created file thai.txtroot@ubuntu:/home/hoangthaifc01/Desktop/manager# ls
file  file.c  thai.txt
```

Hình II.2 Minh hoạ tạo tệp tin

- Đoạn mã:

```
void create_file(int argc, char *argv[] ){
        int fd;
        fd = open (argv[2], O_CREAT, 0666);
        if(fd == -1){
                printf("Create file fail");
                exit(0);
        }
        else{
                printf("Created file %s",argv[2]);
                exit(0);
        }
}
```

II.1.2.2 Đọc nội dung tệp tin

- Hiển thị nội dung của tệp tin dựa vào tên tệp tin người dùng nhập vào.

```
root@ubuntu:/home/hoangthaifc01/Desktop/manager# ls
file  file.c  thai.txt
root@ubuntu:/home/hoangthaifc01/Desktop/manager# ./file cat thai.txt
chieu nay khong co mua roi uot tren doi hang mi
chieu nay khong co mat em cuoi nhu luc xua
duong nhu goc pho cung biet buon
tha hoa bay khap con duong
chieu nay khong co mua roi anh lang thinh
```

Hình II.7 Minh hoạ chức năng đọc nội dung tệp tin

- Đoạn mã:

```
void cat_file(int argc, char *argv[]){
        int fd;
        char buffer[1000];
        fd = open(argv[2], O_RDONLY);
        if(fd == -1){
                printf("File not found\n");
                exit(0);
        }
        while (read(fd, buffer, sizeof(buffer)) != 0){
                printf("%s",buffer);
        }
        printf("\n");
}
```

Hình II.8 Đoạn mã chức năng đọc nội dung tệp tin

II.1.2.3 Xoá tệp tin

- Xoá bỏ tệp tin dựa vào tên tệp tin người dùng nhập vào.

```
root@ubuntu:/home/hoangthaifc01/Desktop/manager# ls
file  file.c  thai.txt
root@ubuntu:/home/hoangthaifc01/Desktop/manager# ./file remove thai.txt
Removed file name: thai.txtroot@ubuntu:/home/hoangthaifc01/Desktop/manager# ls
file  file.c
```

Hình II.15 Minh họa xóa bỏ tệp tin

- **Đoạn mã**

```
void remove_file(int argc, char *argv[]) {
        int status;
        status = unlink(argv[2]);
        if(status == -1){
                printf("cant remove this file, maybe not found\n");
                exit(0);
        }
        else{
                printf("Removed file name: %s",argv[2]);
                exit(0);
        }
}
```

Hình II.16 Đoạn mã chức năng xoá tệp tin

II.2 Quản lý tiến trình

II.2.1 Chức năng

- Liệt kê danh sách tiến trình đang chạy.
- Đóng tiến trình dựa trên ID.

II.2.2 Mô tả

II.2.2.1 Liệt kê danh sách tiến trình đang chạy

- Hiển thị danh sách tiến trình bao gồm ID, tên người dùng, tên tiến trình.

```
root@ubuntu:/home/hoangthaifc01/Desktop/process# ./process ps
ID         Name
 1          systemd
 2          kthreadd
 3          rcu_gp
 4          rcu_par_gp
 5          slub_flushwq
 6          netns
 8          kworker/0:0H-events_highpri
 10         mm_percpu_wq
 11         rcu_tasks_kthread
 12         rcu_tasks_rude_kthread
 13         rcu_tasks_trace_kthread
 14         ksoftirqd/0
 15         rcu_preempt
 16         migration/0
 17         idle_inject/0
 19         cpuhp/0
 20         cpuhp/1
 21         idle_inject/1
 22         migration/1
 23         ksoftirqd/1
 25         kworker/1:0H-events_highpri
```

Hình II.22 Minh hoạ liệt kê danh sách tiến trình

- Đoạn mã:

```
void list_process(int argc, char *argv[]){
        DIR *d;
        struct dirent *dir;
        char buffer[100], status_path[1000], process_name[100];
        int status_fd;

        d = opendir("/proc");
        printf("%-10s%-10s\n","ID","Name");
        while ((dir = readdir(d)) != NULL){
                if(isdigit(dir -> d_name[0])){
                        sprintf(status_path, "/proc/%s/status", dir->d_name);
                        status_fd = open(status_path, O_RDONLY);
                        read(status_fd, buffer, sizeof(buffer));
                        sscanf(buffer, "Name:  %s", process_name);
                        printf(" %-10s%-10s\n", dir->d_name, process_name);
                }
        }
}
```

Hình II.23 Đoạn mã chức năng liệt kê danh sách tiến trình đang chạy

II.2.2.2 Đóng tiến trình

- Thực hiện đóng tiến trình (KILL) dựa vào ID người dùng nhập vào.

```
root@ubuntu:/home/hoangthaifc01/Desktop/process# ./process kill 2767
kill 2767 successsroot@ubuntu:/home/hoangthaifc01/Desktop/process#
```

Hình II.26 Minh họa chức năng đóng tiến trình dựa vào ID

- Đoạn mã:

```
void kill_process(int argc, char *argv[]){
        int process_id, kill_arg, status;
        process_id = atoi(argv[2]);
        kill_arg = atoi(argv[3]);
        status = kill(process_id, kill_arg);
        if(status == 0){
                printf("\033[0;32m");
                printf("kill %d success\n", process_id);
                printf("\033[0m");
                exit(0);
        }
        else{
                printf("\033[0;31m");
                printf("kill fail\n");
                printf("\033[0m");
                exit(0);
        }
}
```

Hình II.27 Đoạn mã chức năng đóng tiến trình dựa vào ID

II.3 Quản lý Socket

II.3.1 Chức năng

- Cho phép chat giữa nhiều Client với nhau, Server đóng vai trò quản lý và hiển thị đường truyền tin.

II.3.2 Mô tả

- Các chức năng

Chức năng	List tcp
Mô tả	Chức năng này sẽ tiến hành hiện những socket tcp lên console
Cách thực hiện	Tiến hành đọc file /proc/net/tcp và in ra thông tin những socket đang được sử dụng trong máy.

Chức năng	List udp

Mô tả	Chức năng này sẽ tiến hành hiện những socket udp lên console
Cách thực hiện	Tiến hành đọc file /proc/net/udp và in ra thông tin những socket đang được sử dụng trong máy.

Chức năng	List raw
Mô tả	Chức năng này sẽ tiến hành hiện những socket raw lên console
Cách thực hiện	Tiến hành đọc file /proc/net/raw và in ra thông tin những socket đang được sử dụng trong máy.

Chức năng	List unix
Mô tả	Chức năng này sẽ tiến hành hiện những socket unix lên console
Cách thực hiện	Tiến hành đọc file /proc/net/unix và in ra thông tin những socket đang được sử dụng trong máy.

Chức năng	List tcp6
Mô tả	Chức năng này sẽ tiến hành hiện những socket tcp6 lên console
Cách thực hiện	Tiến hành đọc file /proc/net/tcp6 và in ra thông tin những socket đang được sử dụng trong máy.

Chức năng	List udp6
Mô tả	Chức năng này sẽ tiến hành hiện những socket udp6 lên console
Cách thực hiện	Tiến hành đọc file /proc/net/udp6 và in ra thông tin những socket đang được sử dụng trong máy.

Chức năng	List raw6
Mô tả	Chức năng này sẽ tiến hành hiện những socket raw6 lên console

Cách thực hiện	Tiến hành đọc file /proc/net/raw6 và in ra thông tin những socket đang được sử dụng trong máy.

- Thực thi

```
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <stdio.h>
void printf_err(char *);
void printf_success(char *);
void check_param();
void tcp_list();
void udp_list();
void printf_err(char *message) {
     printf("\033[0;31m");
     printf("%s", message);
     printf("\033[0m");
}
void printf_success(char *message) {
     printf("\033[0;32m");
     printf("%s", message);
     printf("\033[0m");
}

void check_param(int argc, char *argv[]) {
     if (argc < 2) {
          printf_err("chua nhap lenh\n");
     }
```

```
    if (strcmp(argv[1], "tcp") == 0) {


    }
    if (strcmp(argv[1], "udp") == 0) {


    }
    if (strcmp(argv[1], "raw") == 0) {


    }
    if (strcmp(argv[1], "unix") == 0) {


    }
    if (strcmp(argv[1], "tcp6") == 0) {


    }
    if (strcmp(argv[1], "udp6") == 0) {


    }
    if (strcmp(argv[1], "raw6") == 0) {


    }
}
void tcp_list(int argc, char *argv[]) {
    int fd;
    char buffer[100];
    fd = open("/proc/net/tcp", O_RDONLY);
    while (read(fd, buffer, sizeof(buffer)) != 0) {
```

```
          printf("%s", buffer);
          memset(buffer, 0, sizeof(buffer));
     }
}
void udp_list(int argc, char *argv[]) {
     int fd;
     char buffer[100];
     fd = open("/proc/net/udp", O_RDONLY);
     while (read(fd, buffer, sizeof(buffer)) != 0) {
          printf("%s", buffer);
          memset(buffer, 0, sizeof(buffer));
     }
}
void raw_list(int argc, char *argv[]) {
     int fd;
     char buffer[100];
     fd = open("/proc/net/raw", O_RDONLY);
     while (read(fd, buffer, sizeof(buffer)) != 0) {
          printf("%s", buffer);
          memset(buffer, 0, sizeof(buffer));
     }
}
void unix_list(int argc, char *argv[]) {
     int fd;
     char buffer[100];
     fd = open("/proc/net/unix", O_RDONLY);
     while (read(fd, buffer, sizeof(buffer)) != 0) {
```

```
          printf("%s", buffer);
          memset(buffer, 0, sizeof(buffer));
     }
}
void tcp6_list(int argc, char *argv[]) {
     int fd;
     char buffer[100];
     fd = open("/proc/net/tcp6", O_RDONLY);
     while (read(fd, buffer, sizeof(buffer)) != 0) {
          printf("%s", buffer);
          memset(buffer, 0, sizeof(buffer));
     }
}
void udp6_list(int argc, char *argv[]) {
     int fd;
     char buffer[100];
     fd = open("/proc/net/udp6", O_RDONLY);
     while (read(fd, buffer, sizeof(buffer)) != 0) {
          printf("%s", buffer);
          memset(buffer, 0, sizeof(buffer));
     }
}
void raw6_list(int argc, char *argv[]) {
     int fd;
     char buffer[100];
     fd = open("/proc/net/raw6", O_RDONLY);
     while (read(fd, buffer, sizeof(buffer)) != 0) {
```

```
          printf("%s", buffer);
          memset(buffer, 0, sizeof(buffer));
     }
}
int main(int argc, char *argv[]) {
     check_param(argc, argv);

     if (strcmp(argv[1], "tcp") == 0) {
          tcp_list(argc, argv);
     }
     if (strcmp(argv[1], "udp") == 0) {
          udp_list(argc, argv);
     }
     if (strcmp(argv[1], "raw") == 0) {
          raw_list(argc, argv);
     }
     if (strcmp(argv[1], "unix") == 0) {
          unix_list(argc, argv);
     }
     if (strcmp(argv[1], "tcp6") == 0) {
          tcp6_list(argc, argv);
     }
     if (strcmp(argv[1], "udp6") == 0) {
          udp6_list(argc, argv);
     }
     if (strcmp(argv[1], "raw6") == 0) {
          raw6_list(argc, argv);
```

```
    }
    return 0;
}
```

II.4 Quản lý Network

II.4.1 Chức năng

- Hiển thị địa chỉ IP.

II.4.2 Mô tả

II.4.2.1 Hiển thị các mạng đang dùng

```
hoangthaifc01@ubuntu:~/Desktop/network$ gedit network.c
hoangthaifc01@ubuntu:~/Desktop/network$ make
gcc -o network network.c
hoangthaifc01@ubuntu:~/Desktop/network$ ./network list
device name     protocol        address
lo              IPv_6
ens33           IPv_6
lo              IPv_4           127.0.0.1
ens33           IPv_4           192.168.116.128
lo              IPv_6           ::1
ens33           IPv_6           fe80::7706:6d4e:717:2c51%ens33
hoangthaifc01@ubuntu:~/Desktop/network$ S
```

Hình II.37 Hiển thị các mạng

- Đoạn mã:

```
void list_netdevice(int argc, char *argv[]) {
        struct ifaddrs *addresses, *address;
        int status, family, family_size;
        char buffer[100];

        status = getifaddrs(&addresses);
        if (status == -1) {
                printf("khong the lay duoc dia chi\n");
                exit(0);
        }
        address = addresses;
        printf("%-15s%-15s%-15s\n", "device name", "protocol", "address");
        while (address) {
                family = address->ifa_addr->sa_family;
                family_size = family == AF_INET ? sizeof(struct sockaddr_in) : sizeof(struct
sockaddr_in6);
                memset(buffer, 0, sizeof(buffer));
                getnameinfo(address->ifa_addr, family_size, buffer, sizeof(buffer), 0, 0,
NI_NUMERICHOST);

                printf("%-15s%-15s%-15s\n", address->ifa_name, family == AF_INET ? "IPv_4" :
"IPv_6", buffer);
                address = address->ifa_next;
        }
        freeifaddrs(addresses);

}
```

Hình II.38 Đoạn mã hiển thị các mạng

III. Kernel Module

III.1 Mô tả

Khi cần một module nhưng nó lại chưa có trong kernel space, kernel sẽ đưa module ấy vào. Quá trình này có thể diễn ra một cách tự động, với trình tự sau:

- Bước 1: Kernel kích hoạt tiến trình modprobe cùng với tham số truyền vào là tên của module (ví dụ xxx.ko).
- Bước 2: Tiến trình modprobe kiểm tra file /lib/modules/<kernel-version>/modules.dep xem xxx.ko có phụ thuộc vào module nào khác không. Giả sử xxx.ko phụ thuộc vào module yyy.ko.
- Bước 3: Tiến trình modprobe sẽ kích hoạt tiến trình insmod để đưa các module phụ thuộc vào trước (yyy.ko), rồi mới tới module cần thiết (xxx.ko).

Như vậy, các module được đưa vào kernel space dưới sự giúp đỡ của tiến trình modprobe

III.2 Biên dịch và cài đặt vào nhân

- Makefile:

```
KDIR = /lib/modules/`uname -r`/build

all:
	make -C $(KDIR) M=`pwd`

clean:
	make -C $(KDIR) M=`pwd` clean
```

Hình III.3 Đoạn mã Makefile

Trong Makefile trên:

- Thẻ all chứa câu lệnh để biên dịch các module trong thư mục hiện tại.
- Thẻ clean chứa lệnh xóa tất cả các object file có trong thư mục hiện tại.

- Kbuild

```
EXTRA_CFLAGS = -Wall

obj-m        = hello.o
```

Trong file Kbuild trên:

- Biến obj-m chỉ ra rằng: object file sẽ được biên dịch theo kiểu kernel module.
- Cờ -Wall cho phép trình biên dịch hiển thị tất cả các bản tin cảnh báo trong quá trình biên dịch.

- Biên dịch với: "make" hoặc "makeall"

```
st@hoatruong:~/St/mod$ make all
make -C /lib/modules/`uname -r`/build M=`pwd`
make[1]: Entering directory '/usr/src/linux-headers-5.13.0-37-generic'
  CC [M]  /home/st/St/mod/hello.o
  MODPOST /home/st/St/mod/Module.symvers
  CC [M]  /home/st/St/mod/hello.mod.o
  LD [M]  /home/st/St/mod/hello.ko
  BTF [M] /home/st/St/mod/hello.ko
Skipping BTF generation for /home/st/St/mod/hello.ko due to unavailability of vmlinux
make[1]: Leaving directory '/usr/src/linux-headers-5.13.0-37-generic'
```

Hình III.4 Biên dịch module nhân

- Nạp module vào kernel space qua lệnh: "sudo insmod hello.ko

```
st@hoatruong:~/St/mod$ sudo insmod hello.ko
[sudo] password for st:
st@hoatruong:~/St/mod$ dmesg
```

Hình III.5 Nạp module vào nhân

- Kiểm tra module được nạp vào qua lệnh "lsmod"

III.3 Kết luận

Driver có thể được tích hợp luôn vào trong kernel hoặc được thiết kế dưới dạng module tách rời. Driver cho các thiết bị cố định, ví dụ các thiết bị trong smartphone, sẽ được tích hợp luôn vào trong kernel. Driver cho các thiết bị hay phải thay đổi, sẽ được thiết kế dưới dạng loadable module. Thông thường, các bus driver sẽ được tích hợp vào trong kernel, còn các device driver được thiết kế dưới dạng loadable module. Ngoài device driver, thì system call và file system cũng được thiết kế theo kiểu loadable module.

Kernel module có thể được đưa vào trong kernel một cách tự động, hoặc thủ công.

- Đối với trường hợp tự động, kernel kích hoạt tiến trình modprobe thông qua kmod hoặc udevd. Sau đó, modprobe sẽ đưa module cần thiết vào.

- Đối với trường hợp thủ công, ta sẽ sử dụng lệnh insmod hoặc modprobe. Còn để đưa kernel module ra khỏi kernel space, ta sẽ sử dụng lệnh rmmod.

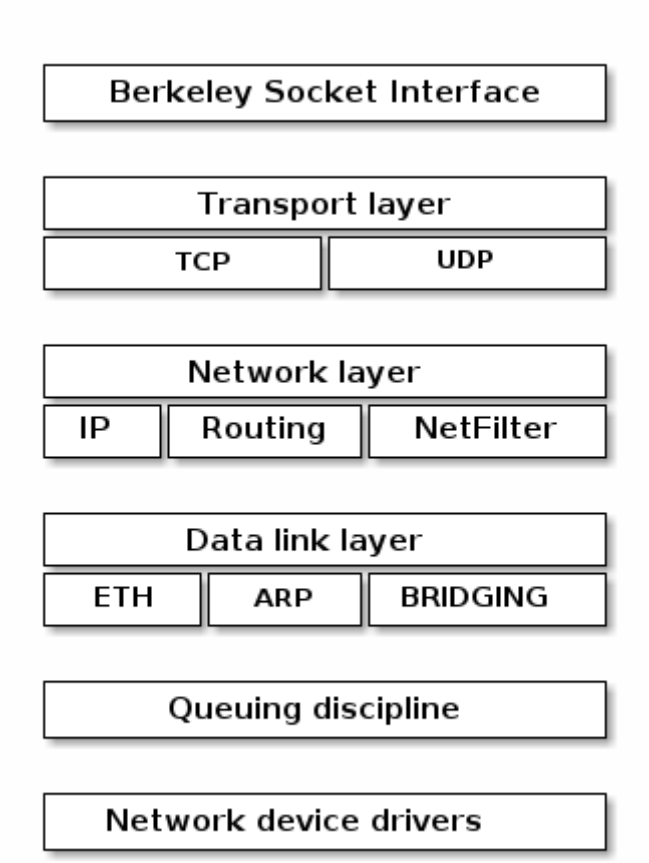
Linux kernel cũng giống như một thư viện giúp lập trình viên xây dựng các kernel module. Khi viết một kernel module, ta cần phải tham chiếu tới các file trong thư mục include/linux. Bất cứ một module nào cũng cần tham chiếu tới file <linux/module.h>. File này chứa 2 macro quan trọng, đó là module_init và module_exit. Hai macro này giúp xác định đâu là hàm khởi tạo module, đâu là hàm kết thúc module. Thông thường, ta nên đặt các macro __init trước hàm khởi tạo, và macro __exit trước hàm kết thúc để tiết kiệm bộ nhớ

IV. Dịch thuật-Quản lý mạng

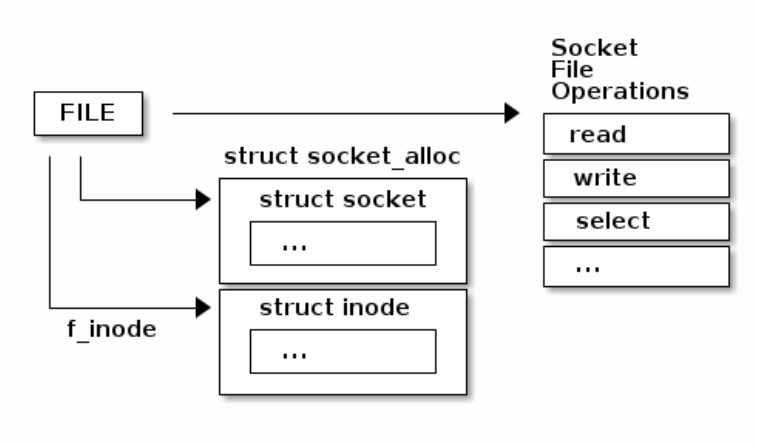
Mục tiêu:

- Triển khai Socket
- Triển khai bộ định tuyến
- Giao diện thiết bị mạng
- Kỹ thuật tăng tốc phần cứng và phần mềm

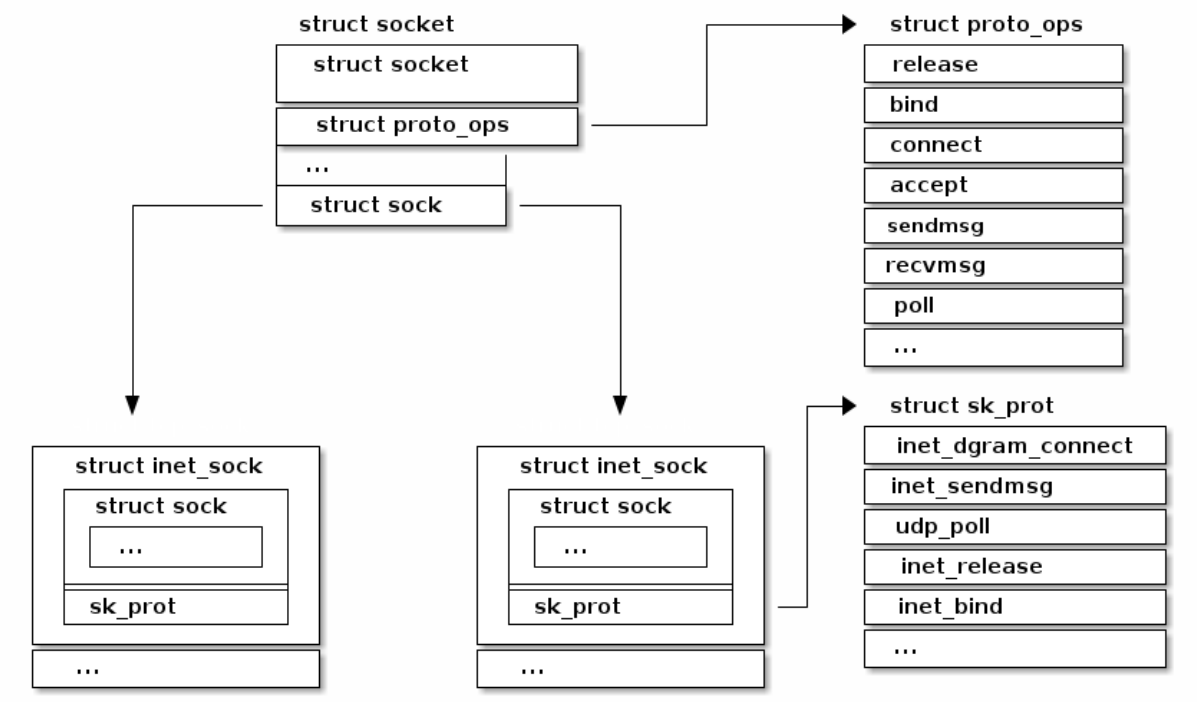
IV.1 Tổng quan về quản lý mạng



IV.2 Tổng quan về triển khai socket:



IV.3 Giao thức và các mối liên hệ với socket:



IV.4 Các giai đoạn xử lý mạng:

- Trình xử lý ngắt - trình điều khiển thiết bị tìm nạp dữ liệu từ vòng RX, tạo gói mạng và xếp nó vào ngăn xếp mạng để xử lý
- NET_SOFTIRQ - gói đi qua lớp ngăn xếp và nó được xử lý: giải mã khung Ethernet, kiểm tra gói IP và định tuyến nó, nếu gói cục bộ giải mã gói giao thức (ví dụ: TCP) và xếp nó vào một ổ cắm
- Bối cảnh quy trình - ứng dụng tìm nạp dữ liệu từ hàng đợi ổ cắm hoặc đẩy dữ liệu vào hàng đợi ổ cắm

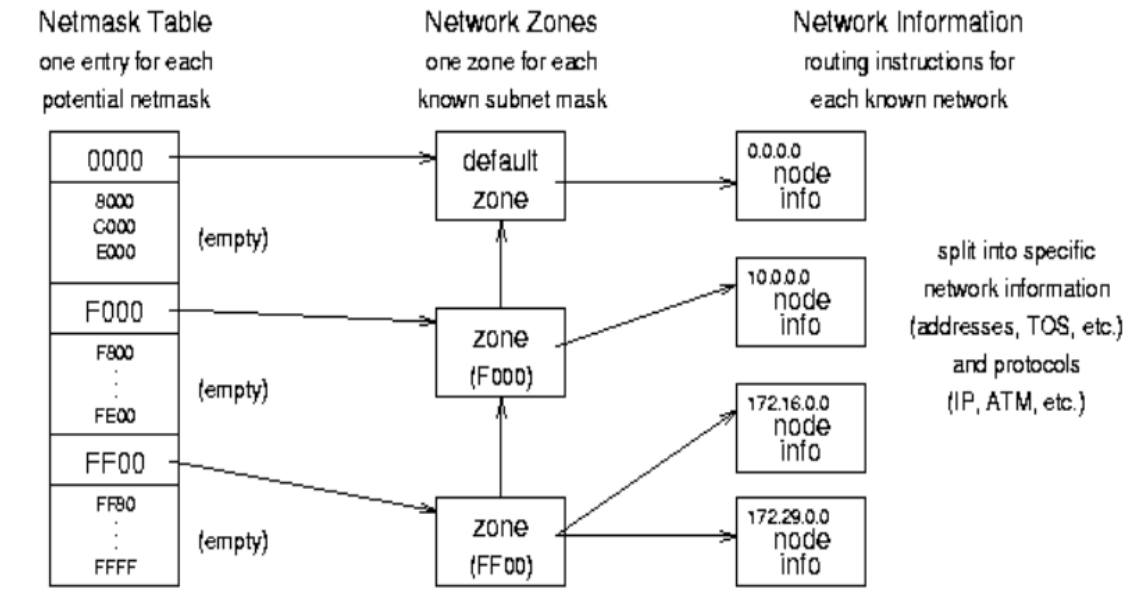
IV.5 Bảng định tuyến

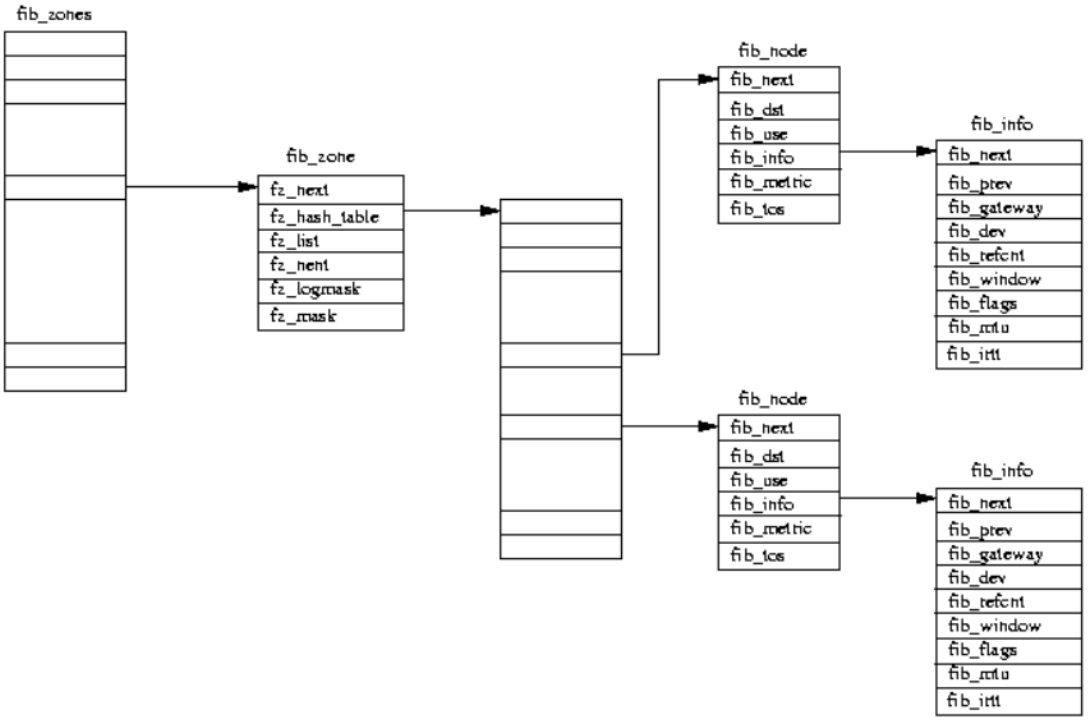
```
root@ubuntu:/home/hoangthaifc01/Desktop# ip route list table main
default via 192.168.116.2 dev ens33 proto dhcp metric 100
169.254.0.0/16 dev ens33 scope link metric 1000
192.168.116.0/24 dev ens33 proto kernel scope link src 192.168.116.128 metric 10
0
root@ubuntu:/home/hoangthaifc01/Desktop# ip route list table local
local 127.0.0.0/8 dev lo proto kernel scope host src 127.0.0.1
local 127.0.0.1 dev lo proto kernel scope host src 127.0.0.1
broadcast 127.255.255.255 dev lo proto kernel scope link src 127.0.0.1
local 192.168.116.128 dev ens33 proto kernel scope host src 192.168.116.128
broadcast 192.168.116.255 dev ens33 proto kernel scope link src 192.168.116.128
root@ubuntu:/home/hoangthaifc01/Desktop# ip rule list
0:      from all lookup local
32766:  from all lookup main
32767:  from all lookup default
root@ubuntu:/home/hoangthaifc01/Desktop#
```

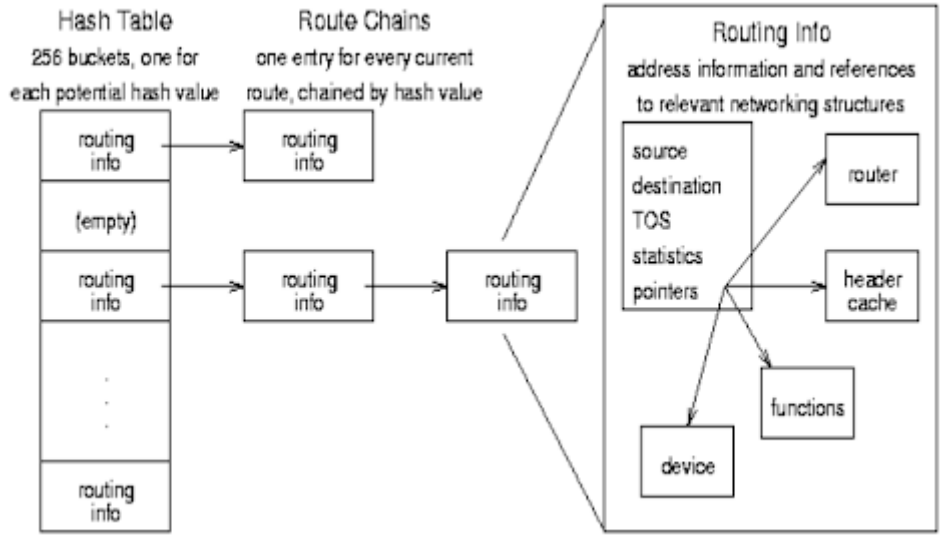
IV.6 Cơ sở dữ liệu chính sách định tuyến

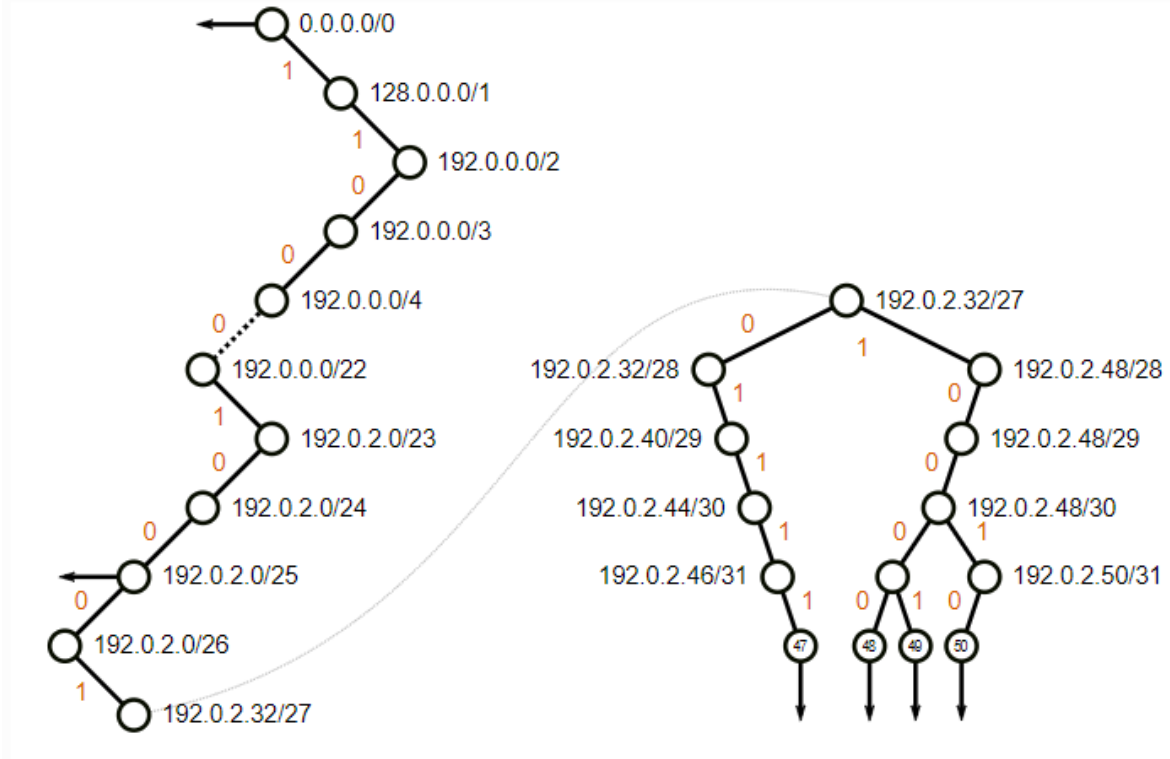
- Định tuyến "thông thường" chỉ sử dụng địa chỉ đích
- Để tăng tính linh hoạt, "Cơ sở dữ liệu chính sách định tuyến" được sử dụng cho phép định tuyến khác nhau dựa trên các trường khác như địa chỉ nguồn, loại giao thức, cổng truyền tải, v.v.
- Điều này được mã hóa dưới dạng danh sách các quy tắc được đánh giá dựa trên mức độ ưu tiên của chúng (mức độ ưu tiên 0 là mức cao nhất)
- Mỗi quy tắc có một bộ chọn (cách khớp gói) và một hành động (hành động cần thực hiện nếu gói phù hợp)
- Bộ chọn: địa chỉ nguồn, địa chỉ đích, loại dịch vụ (TOS), giao diện đầu vào, giao diện đầu ra, v.v.
- Hành động: tra cứu / unicast - sử dụng bảng định tuyến đã cho, lỗ đen - thả gói tin, không thể truy cập - gửi tin nhắn ICMP không thể truy cập và thả gói tin, v.v.

IV.7 Cơ sở dữ liệu thông tin chuyển tiếp









IV.8 Netfilter

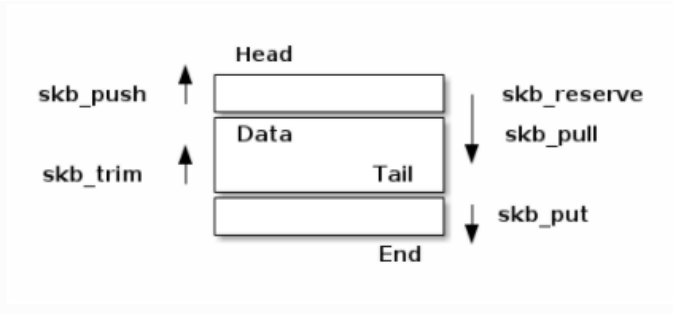
Khung thực hiện lọc gói và NAT

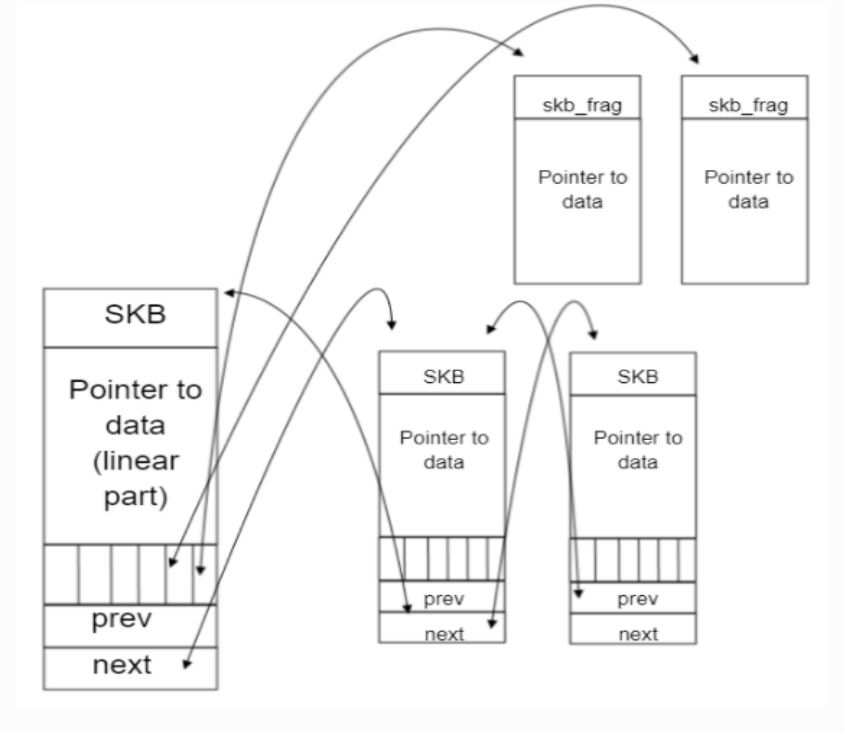
Nó sử dụng các móc được chèn vào những vị trí quan trọng trong luồng gói:

- NF_IP_PRE_ROUTING
- NF_IP_LOCAL_IN
- NF_IP_FORWARD
- NF_IP_LOCAL_OUT
- NF_IP_POST_ROUTING

- NF_IP_NUMHOOKS

IV.9 Gói mạng / skbs (struct sk_buff)





IV.10 Thiết bị mạng

- Phân tán-Thu thập
- Checksum giảm tải: Ethernet, IP, UDP, TCP
- Xử lý gián đoạn thích ứng (liên kết, thích ứng)

IV.11Kỹ thuật tăng tốc phần cứng và phần mềm

- Giảm tải hoàn toàn - Triển khai ngăn xếp TCP / IP trong phần cứng
- Vấn đề:
  - Chia tỷ lệ số lượng kết nối

- Bảo vệ
- Sự phù hợp

- Hiệu suất tỷ lệ thuận với số lượng gói được xử lý
- Ví dụ: nếu một điểm cuối có thể xử lý 60K pps
  - 1538 MSS -> 738Mbps
  - 2038 MSS -> 978Mbps
  - 9038 MSS -> 4,3Gb / giây
  - 20738 MSS -> 9,9Gb / giây
- Ngăn xếp mạng xử lý các gói lớn
- Đường dẫn TX: phần cứng phân chia các gói lớn thành các gói nhỏ hơn (TCP Segmentation Offload)
- Đường dẫn RX: phần cứng tổng hợp các gói nhỏ thành các gói lớn hơn (Tải trọng nhận lớn - LRO)