# Các bài thực Hành Linux

Linux

# Mục lục

# *Bài 1* ĐĂNG NHẬP HỆ THỐNG LINUX

## 1.1. Truy cập vào máy tính đã cài đặt hệ điều hành Linux

Khởi động máy đã cài đặt Linux, xuất hiện dấu nhắc khởi động hệ điều hành:

```
Boot : linux
```

Khi HĐH Linux khởi động, xuất hiện dấu nhắc truy cập hệ thống :

```
login :
password :
```

Người dùng nhập vào username và password tương ứng, trên màn hình xuất hiện dấu nhắc của hệ thống như sau :

```
[user12@linux user12]$
```

## 1.2. Sử dụng Telnet để truy cập vào máy Linux từ xa

Truy cập vào Server LINUX từ máy Windows. Yêu cầu máy Windows đã cài đặt mạng. Để kiểm tra hệ thống mạng, từ dấu nhắc cửa lệnh trên Windows, gõ lệnh :

```
C:\>ping 200.201.202.180
```

Nếu trên màn hình xuất hiện : `Reply from 200.201.202.180 ...`

thì nghĩa là máy tính có khả năng truy cập vào Server LINUX, ngược lại, nếu có thông báo nào khác thông báo như trên thì nên kiểm tra lại cấu hình mạng trên máy. Tiếp theo, ta gõ lệnh :

**telnet 200.201.202.180**

Sau một khoảng thời gian thiết lập liên kết, trên cửa sổ telnet xuất hiện :

```
login :
password :
```

- Người dùng nhập vào username và password tương ứng.

*Ví dụ* : Đăng nhập vào với tài khoản *user12*, trên màn hình xuất hiện như sau :

```
login: user12
Password:
Last login: Wed Apr  7 08:35:50 from 131.16.16.21
[user12@linux user12]$
```

## 1.3. Thoát khỏi hệ thống

Thoát khỏi phiên làm việc : `#exit` hoặc `#logout`

Chấm dứt hoạt động của hệ thống : `#shutdown –h now`

*Bài 2* **SỬ DỤNG E-Mail**

Thư điện tử hiện nay đang trở thành phương tiện chính để liên lạc trên mạng. Thư điện tử dễ sử dụng, tiện lợi và nhanh chóng. Trong phần này ta sử dụng dịch vụ sendmail của hệ thống Linux.

## 2.1. Gởi thư bằng sendmail

***Cú pháp* : mail <address1> <address2> <address3> . . .**

**$mail user01 root**

- Tiếp theo, trên màn hình xuất hiện

**Subject :**

- Bạn gõ vào chủ đề bức thư. Nhấn Enter, bắt đầu nhập vào nội dung thư.
- Sau khi nhập vào nội dung thư, nhấn CTRL-D để gởi thư đi.
- Trên màn hình xuất hiện :

**CC :**

- Nhập vào tên những người cùng nhận thư hoặc nhấn Enter để bỏ qua.

## 2.2. Nhận thư

- Khi có thư đến, trên màn hình xuất hiện thông báo :

**You have mail**

- Để đọc thư, gõ vào lệnh : **$mail**
- Trên màn hình sẽ liệt kê các bức thư theo thứ tự 1, 2, 3 ... Để đọc nội dung thư nào, gõ vào số thứ tự của bức thư đó.
- Dấu & nhắc rằng bạn đang ở chương trình đọc thư.
- Để xóa thư đang đọc, tại dấu nhắc bạn gõ : **&d**
- Để thoát chương trình đọc thư, tại dấu nhắc bạn gõ : **&q**

Ví dụ Một phiên gởi mail của user12 :

```
[user12@linux user12]$ mail user15 root
Subject: Chao ban
        Thuc hanh LINUX
Cc:
[user12@linux user12]$
```

## 2.3. Các thao tác hỗ trợ

- Để hủy bỏ thư trước khi gởi, bạn nhấn CTRL-C hai lần.
- Đọc nội dung một tập tin trên thư mục hiện hành vào mail : **~r filename**

- Thay đổi chủ đề của thư : **~s**
- Xem tất cả các thư lưu trong hộp thư : **$more mbox**

# Các lệnh thao tác trên sendmail

| | |
|---|---|
| t <message list> | type messages |
| n | goto and type next message |
| e <message list> | edit messages |
| f <message list> | give head lines of messages |
| d <message list> | delete messages |
| s <message list> | file append messages to file |
| u <message list> | undelete messages |
| R <message list> | reply to message senders |
| r <message list> | reply to message senders and all recipients |
| pre <message list> | make messages go back to /usr/spool/mail |
| p <message list> | print message |
| m <user list> | mail to specific users |
| q | quit, saving unresolved messages in mbox |
| x | quit, do not remove system mailbox |
| h | print out active message headers |
| ! | shell escape |
| cd [directory] | chdir to directory or home if none given |

# *Bài 3* CÁC LỆNH TRÊN LINUX

## 3.1. Tổ chức hệ thống tập tin trên Linux

| | |
|---|---|
| /etc | Cấu hình hệ thống cục bộ theo máy |
| /usr/bin | Chứa hầu hết các lệnh người dùng. |
| /dev | Các tập tin thiết bị. |
| /usr/man | Chứa các tài liệu trực tuyến. |
| /usr/include | Chứa các tập tin include chuẩn của C. |
| /var/log | Các tập tin lưu giữ thông tin làm việc hiện hành của người dùng. |
| /home | Chứa các thư mục con của các user. |
| /usr/lib | Chứa các tập tin thư viện của các chương trình người dùng. |

Khi truy cập vào hệ thống, thư mục làm việc của người dùng được xem như là thư mục chủ. Ví dụ : Thư mục chủ của user01 sẽ là **/home/user01**

Nếu đường dẫn bắt đầu bằng dấu "/", hệ thống xem đó như là một tên đường dẫn đầy đủ bắt đầu từ thư mục gốc.

## 3.2. Các lệnh thao tác trên hệ thống tập tin

Các tham số luôn bắt đầu bởi dấu "-", và trong hầu hết các trường hợp nhiều tham số một chữ cái có thể kết hợp dùng một dấu "-".

*Ví dụ:* Thay vì dùng lệnh **ls -l -F**, ta có thể dùng lệnh tương đương **ls -lF.**

| Kí tự | Chức năng |
|---|---|
| ***?[ ]** | Kí tự đại diện hay theo mẫu |
| **&** | Chạy ứng dụng ở chế độ nền (background), trả lại dấu nhắc hệ thống cho các tác vụ khác . |
| **;** | Dấu phân cách nhiều lệnh trên một dòng lệnh. |
| **** | Tắt tác dụng của những kí tự đặc biệt như ***, ?, [, ], &, ;, >, <, \|** |
| **>** | Định hướng dữ liệu xuất ra file. |
| **<** | Định hướng dữ liệu nhập từ file. |
| **>>** | Định hướng dữ liệu xuất ra cuối file nếu file đã tồn tại. |
| **\|** | Định hướng dữ liệu xuất là dữ liệu nhập cho lệnh tiếp theo. |
| **$** | Sử dụng biến môi trường. |

### 3.2.1. Tạo mới thư mục

***Cú pháp*** : **mkdir <dir1> <dir2> ... <dirN>**

**<dir1> . . . <dirN>** là tên các thư mục cần tạo.

```
[user01@linux user01]$ mkdir baitap
```

```
[user01@linux user01]$ mkdir document
[user01@linux user01]$ mkdir baitap\ltc
[user01@linux user01]$ ls
[user01@linux user01]$ mkdir baitap/ltc
[user01@linux user01]$ mkdir baitap/perl
```

### 3.2.2. Thay đổi thư mục hiện hành

***Cú pháp*** : **cd <directory>**

**<directory>** là thư mục muốn chuyển đến.

**.** : yêu cầu chuyển đến thư mục hiện hành.

**. .** : chuyển đến thư mục cha.

```
[user01@linux user01]$ cd baitap
[user01@linux user01]$ cd /home
[user01@linux user01]$ cd
```

### 3.2.3. Xem thư mục làm việc hiện hành

***Cú pháp :*** **pwd**

```
[user12@linux user12]$ pwd
/home/user12
[user12@linux user12]$
```

### 3.2.4. Xem thông tin về tập tin và thư mục

***Cú pháp*** **: ls <file1> <file2> … <fileN> <Tham số>**

**<file1> . . . <fileN>** là danh sách tên tập tin hay thư mục.

<Tham số> :

-F : dùng để hiển thị một vài thông tin về kiểu của tập tin

-l : (long) liệt kê kích thước tập tin, người tạo ra, các quyền người sử dụng.

```
[user12@linux user12]$ ls -lF
total 75
drwxrwxr-x   2 user12      user12        1024 Apr  7 09:41 baitap/
drwxrwxr-x   2 user12      user12        1024 Apr  7 09:41 doc/
-rwxrwxr-x   1 user12      user12       71 Mar 31 10:39 hello*
-rw-rw-r--   1 user12      user12        126 Apr  7 09:26 baitho.txt
-rw-rw-r--   1 user12      user12       70 Apr  7 08:26 hello.c
[user12@linux user12]$
```

**ls –lF**

**ls *a*** : hiển thị tất cả tập tin hay thư mục con có kí tự a

**ls F*E** : hiển thị danh sách bắt đầu bằng F và kết thúc bằng E

### 3.2.5. Di chuyển một hay nhiều tập tin

***Cú pháp*** **: mv <file1> <file2> … <fileN> <destination>**

**<file1> . . . <fileN>** là danh sách tên tập tin cần di chuyển

**<destination>** là tập tin hay thư mục đích.

Lệnh **mv** có thể dùng để đổi tên tập tin.

- Chuyển nhiều tập tin

```
$ mv  * directory
```

- Di chuyển thư mục

```
[user01@linux user01]$ mkdir ctrinh
[user01@linux user01]$ ls –lF
[user01@linux user01]$ mv ctrinh baitap
```

Di chuyển thư mục **/home/user01/ctrinh** vào thư mục **/home/user01/baitap**

### 3.2.6. Sao chép tập tin

***Cú pháp*** **: cp <source> <destination>**

```
[user01@linux user01]$ cd baitap
[user01@linux baitap]$ vi tho.txt
[user01@linux baitap]$ mv tho.txt baitho.doc
[user01@linux baitap]$ ls
baitho.doc  ctrinh      hello.c     ltc         perl
[user01@linux baitap]$ cp baitho.doc ~/document
```

- Sao chép tất cả các tập tin vào một danh mục

```
$ cp   *   directory
```

### 3.2.7. Tạo liên kết với tập tin

Tạo liên kết với tập tin là tạo thêm cho tập tin tên mới và đường dẫn tương ứng.

***Cú pháp*** **: ln  <source> <destination>**

**ls -l** : xem số liên kết của tập tin.

Muốn xóa một tập tin ta phải xoá tất cả các liên kết của nó.

```
[user01@linux user01]$ pwd
[user01@linux user01]$ ls  -l
[user01@linux user01]$ ls  -l  baitap
[user01@linux user01]$ ln  baitap/file1   file.link
[user01@linux user01]$ ls  -l  baitap
[user01@linux user01]$ ls  -l  file.link
```

### 3.2.8. *Tìm kiếm một tập tin*

Lệnh find cho phép tìm kiếm một hay nhiều tập tin trong một cây danh mục.

- Tìm theo tên: ***find  <path> -name  <filename>***
- Tìm theo số i-node của tập tin: ***find  <path> -inum <number>***
- Tìm theo tên người sở hữu : ***find <path> -user  <username>***

Để tránh các thông báo lỗi đưa ra màn hình, ta có thể đổi hướng đầu ra lỗi chuẩn (*standard error*) tới một tập tin rỗng (/dev/null ):

```
$ find  /  -name  filename  - print  2>/dev/null
```

Ví dụ:

```
$ pwd
/home/user01
$ find  / -name  ttyc2d1  - print  2>/dev/null
/dev/ttyc2d1
$ ls  -i  /unix
2810  - r- xr- - r- -  2  bin  bin  508516  Mar  10  1989  /unix
$find  /  -inum  2810  - print  2>/dev/null
```

### *3.2.9. Xoá* thư mục rỗng

***Cú pháp* : rmdir <dir1> <dir2> ... <dirN>**

**<dir1> ... <dirN>** là tên những thư mục cần xóa.

rmdir /home/baitap  xóa thư mục /home/baitap

### *3.2.10. Xóa các tập tin hoặc thư mục*

***Cú pháp* : rm <file1> <file2> ... <fileN>**

### *3.2.11. Xem hướng dẫn sử dụng lệnh*

***Cú pháp* : man <command>**
**ho?c  <command> --help**
**<command> /?**

Trong đó **<command>** là tên của một cần xem hướng dẫn.

```
[user12@linux user12]$ man ls
[user12@linux user12]$ cp --help
[user12@linux user12]$ cp --help >cp.txt
```

### 3.2.12. Hiển thị nội của các tập tin

***Cú pháp* : more <file1> <file2> ... <fileN>**

***<file1> <file2> ... <fileN>*** là những tập tin cần hiển thị.

```
[user12@linux user12]$more baitho.txt          // hiển thị tập tin baitho.txt
[user12@linux user12]$more mbox          // Xem tất cả thư lưu trong hộp thư
```

### 3.2.13. Nối các tập tin

***Cú pháp* : cat <file1> <file2> ... <fileN> [>filename]**

Lệnh dùng để hiển thị toàn bộ nội dung của nhiều tập tin cùng một lúc.

**<file1> <file2> ... <fileN>** là những tập tin cần hiển thị nội dung.

Ví dụ:

Hiển thị nội dung hai tập tin baitho.txt và vanban.doc

```
$cat baitho.txt vanban.doc
```

Kết nối nội dung hai tập tin baitho.txt và vanban.doc vào tập tin thop.doc

```
$cat baitho.txt vanban.doc >thop.doc
```

### 3.2.14. Xuất nội dung thông báo

***Cú pháp*** **: echo <arg1> <arg2> … <argN>**

Trong đó **<arg1> <arg2> … <argN>** là các đối số dòng lệnh.

```
[user12@linux user12]$ echo "Chao cac ban"
Chao cac ban sinh vien"
[user12@linux user12]$echo *                    → Hiển thị nội dung thư mục
```

### 3.2.15. Nén và giải nén tập tin

***Cú pháp*** **: gzip <filename>**

Nén một tập tin. Tên tập tin nén giống như tên ban đầu, kèm theo đuôi .gz

```
[user12@linux user12]$ gzip vanban.txt          ->  vanban.txt.gz
```

***Cú pháp*** **: gunzip <filename>**
**gzip -d <filename>**

Lệnh dùng để giải nén tập tin.

```
[user12@linux user12]$gunzip vanban.txt.gz
```

## 3.3. Các lệnh hệ thống

### 3.3.1. Lệnh at

Thực hiện lệnh theo thời gian định trước

```
[user12@linux user12]$ at 8:15am Feb 27
echo Happy birthday | mail emily  <CR>
<^d>
[user12@linux user12]$atrm jobnumber      xóa lệnh trong hàng đợi
[user12@linux user12]$at -l               hiển thị danh sách các lệnh trong hàng đợi
```

### 3.3.2. Lệnh hostname

Hiển thị tên máy tính đang làm việc.

Hệ thống lưu thông tin về tên máy trong tập tin /etc/hosts

```
[user12@linux user12]$ hostname
linux.edu.vn
```

### 3.3.3. Lệnh ps

Xem danh sách các tiến trình đang hoạt động trên hệ thống.

```
[user12@linux user12]$ ps
 PID TTY STAT TIME COMMAND
```

```
4516  p4 S    0:00 -bash
4703  p4 S    0:00 /usr/bin/mc -P
4705  r0 S    0:00 bash -rcfile .bashrc
4727  r0 R    0:00 ps
[user12@linux user12]$ kill 4703          //Hủy bỏ tiến trình mc có số hiệu 4703
Terminated
```

### 3.3.4. Lệnh clear

Xóa màn hình.

### 3.3.5. Lệnh date

Hiển thị ngày tháng hiện hành của hệ thống

### 3.3.6. Lệnh cal <month> <year>

Xem lịch tương ứng với tháng và năm chỉ định.

### 3.3.7. Lệnh mount

***Cú pháp*** **: mount [-t <type>] <device> <mountpoint>**

- Lệnh dùng để kết nối hệ điều hành với các thiết bị khác trên hệ thống.
- Lệnh này *chỉ thực hiện* được khi bạn vào hệ thống với tư cách là root.

**type** : Kiểu tập tin

**device** : Tập tin điều khiển thiết bị kết nối.

**mountpoint** : Vị trí thư mục trên hệ điều hành dùng để kết nối với file thiết bị.

- Tạo kết nối với đĩa logic 1 : #mount /dev/hda1 /mnt/hdisk
- Tạo kết nối với đĩa mềm MS-DOS: #mount /dev/fd0 /mnt/floppy
- Tạo kết nối với đĩa mềm LINUX : #mount -t ext2 /dev/fd0 /mnt/floppy
- Tạo kết nối với đĩa CDROM : #mount /dev/hda1 /mnt/cdrom
- Hủy kết nối với đĩa mềm : #umount /dev/fd0

Chú ý : ***Hệ thống Linux xem các thiết bị kết nối như các một tập tin đặc biệt.***

### 3.3.8. Tiện ích mc

Tiện ích mc trên Linux có giao diện làm việc giống như trình NC Command của MS - DOS. Để khởi động mc gõ lệnh như sau :

**#mc**

### *3.3.9. Tiện ích máy tính bc*

Chương trình *bc* cung cấp một bộ máy tính tay giúp người dùng có thể tính toán các biểu thức, các hàm toán học ...

#### *3.3.9.1. Khởi động bc*

Từ dấu nhắc hệ thống, bạn gõ :

```
#bc ↵
```

xuất hiện dấu nhắc, bạn có thể nhập vào các biểu thức tính toán :

```
(4+5)*(12-10) ↵
18
1000000000000*1000000000000↵
1000000000000000000000000
```

Để ấn định số chữ số thập phân, dùng lệnh scale = n :

```
scale=3↵
1/6↵
.166
```

Lập trình trong bc.

```
define giaithua(n)
{
if (n<=1) return (1);
else return (gt(n-1)*n);
}
gt(5)
120
```

Để chuyển đổi sang các cơ số khác nhau, dùng các lệnh ibase và obase

| | |
|---|---|
| ibase=cơsố | Định dạng cơ số đầu vào |
| obase=cơsố | Định dạng cơ số xuất ra |

ibase và obase ngầm định là cơ số 10

```
ibase=16↵
FF↵
255
obase=2↵
FF↵
11111111
ibase↵
obase↵
```

#### *3.3.9.2. Kết thúc bc*

Nhấn CTRL-D để thoát.

```
#man bc
```
-> Xem các hướng dẫn sử dụng bc.

# *Bài 4* QUẢN LÝ TÀI KHOẢN VÀ PHÂN QUYỀN SỬ DỤNG

Mô tả cơ chế bảo vệ tập tin của LINUX: người sử dụng, nhóm người sử dụng, các quyền truy xuất trên tập tin.

## 4.1. Quản lý tài khoản của hệ thống

### 4.1.1. Tài khoản người dùng

Mỗi người sử dụng trên hệ thống được mô tả qua các thông tin sau:

- username : tên người sử dụng
- password : mật khẩu (nếu có)
- uid : số nhận dạng (user identify number )
- gid : số của nhóm (group identify number )
- comment : chú thích
- Thư mục chủ của tài khoản (home directory )
- Shell đăng nhập (chương trình chạy lúc bắt đầu phiên làm việc)

Các thông tin trên được chứa trong tập tin */etc/passwd*

### 4.1.2. Tài khoản nhóm người dùng

Một nhóm người sử dụng được mô tả bằng các thông tin sau:

- groupname : tên của nhóm
- gid : số của nhóm (gid: group identify number)
- danh sách các tài khoản thuộc nhóm

Các thông tin trên được chứa trong tập tin */etc/group*

## 4.2. Phân quyền người dùng trên hệ thống tập tin

### 4.2.1. Các quyền truy xuất trên tập tin

Khi tập tin được tạo lập, các thông tin sau đây đồng thời được ghi lại:

- ***uid*** của người tạo tập tin
- ***gid*** của người tạo tập tin
- Các quyền thâm nhập tập tin khác . . .
- Tập tin được bảo vệ bởi một tập hợp các bit định nghĩa quyền thâm nhập

| | | |
|---|---|---|
| **r w x** | **r w x** | **r w x** |
| **suid** | **sgid** | |
| **owner** | **group** | **other** |

Trong đó:

r Quyền đọc nội dung tập tin, thư mục

w Quyền tạo và xoá nội dung tập tin, tạo và xóa tập tin trong thư mục

x Quyền thực thi tập tin. Quyền truy xuất qua lại trên thư mục.

- *Các quyền với thư mục chỉ có hiểu lực ở một mức nhất định, thư mục con có thể được bảo vệ trong khi thư mục cha thì không.*
- *Lệnh **ls -lF** liệt kê danh sách các tập tin và các thuộc tính của chúng trong một danh mục, qua đó ta có thể xem các thông tin như loại tập tin, quyền truy nhập, người sở hữu và kích thước của tập tin. . .*

### 4.2.2. Lệnh chmod

Lệnh **chmod** cho phép thay đổi quyền trên tập tin của người dùng. Chỉ những người sở hữu tập tin này mới có thể thay đổi được mức đặc quyền đối với tập tin này. Có thể thực hiện lệnh theo hai cách:

*4.2.2.1. Dùng các ký hiệu tượng trưng:*

***Cú pháp* : chmod {a,u,g,o}{+,-,=}{r,w,x} <filename>**

Trong đó : u (user), g (group), o (other), a (all)

Các toán tử : + thêm quyền. - bớt quyền. = gán giá trị khác

*4.2.2.2. Dùng thông số tuyệt đối*

***Cú pháp* : chmod <mode> <filename>**

trong đó mode là một số cơ số 8 ( octal )

```
   r w x        r - x        r - -
   1 1 1        1 0 1        1 0 0
    7            5            4
$chmod  754  filename
$chmod  g-w,o+r   baitho.doc
$chmod  a+r baocao.txt
$chmod  +r baocao.txt
$chmod  og-x baocao.txt          không cho thực thi
$chmod  u+rwx baocao.txt         cho  phép người sở hữu có thể đọc, viết và thực thi.
$chmod  o-rwx baocao.txt         không cho truy nhập tập tin.
$chmod  777   *                  Đặt các quyền cho tất cả các đối tượng sử dụng .
                                 trên toàn bộ tập tin trong thư mục hiện hành
```

### 4.2.3. Thay đổi người hoặc nhóm sở hữu tập tin

- Lệnh **chown** cho phép thay đổi người sở hữu, nhóm sở hữu trên tập tin.
- Lệnh **chgrp** cho phép thay đổi nhóm sở hữu trên tập tin.

# THỰC HÀNH

## 1. Thay đổi quyền trên tập tin

```
#cat bai1.sh
#ls -lF bai1.sh
#chmod u+x,g+wx bai1.sh
#ls -lF bai1.sh
#chmod 644 bai1.sh
#ls -lF bai1.sh
#chmod 764 bai1.sh
#ls -lF bai1.sh
#chmod 777 bai1.sh
#ls -lF bai1.sh
```

## 2. Tạo tài khoản hệ thống

Tạo nhóm cntt2004

```
#groupadd cntt2004
```

Xem tập tin /etc/group

```
#cat /etc/group
```

Tạo một account user01 mới thuộc nhóm cntt2004

```
#useradd - g cntt2004 -c "Tai khoan user01" user01
#passwd user01
```

Xem tập tin /etc/passwd, /etc/shadow

```
#cat /etc/passwd
#cat /etc/shadow
```

Thử đăng nhập vào hệ thống với tài khoản là user01

Tạo một account user02

```
#useradd user02
#passwd user02
```

Đưa user02 vào nhóm cntt2004

```
#usermod -g cntt2004 user02
```

Thử đăng nhập vào hệ thống với tài khoản là user02

Xóa user02

```
#userdel user02
#cat /etc/passwd
```

## 3. Thay đổi quyền sử dụng cho các đối tượng trên tập tin

a. Tạo một tập tin mới /home/baocao.txt

b. Đổi chủ sở hữu của tập tin /home/baocao.txt là user01

```
#chown user01 /home/baocao.txt
```

c. Phân quyền rwxr--r-- cho các đối tượng trên tập tin /home/baocao.txt.

```
#chmod 744 /home/baocao.txt
```

d. Đăng nhập vào hệ thống với tài khoản user01. Thử thay đổi nội dung tập tin /home/baocao.txt.

e. Đăng nhập vào hệ thống với tài khoản khác. Thử thay đổi nội dung tập tin /home/baocao.txt. Nhận xét ?

**4. Phân quyền sử dụng cho các đối tượng**

a. Tạo nhóm người sử dụng có tên cntt2004.

b. Bổ sung các user01, user02 vào nhóm cntt2004.

```
#usermod -g cntt2004 user01
#usermod -g cntt2004 user02
```

c. Tạo thư mục /home/common

```
#mkdir /home/common
```

d. Đổi nhóm sở hữu của thư mục /home/common là nhóm cntt2004.

```
#chown :cntt2004 /home/common
```

hoặc

```
#chgrp cntt2004 /home/common
```

e. Phân quyền rwx cho đối tượng nhóm cntt2004 trên thư mục /home/common

```
#chmod g+rwx /home/common
#ls -lF /home
```

f. Đăng nhập vào hệ thống với tài khoản user01. Tạo thư mục mới trong /home/common.

g. Đăng nhập vào hệ thống với một tài khoản khác không thuộc nhóm cntt2004. Thử tạo thư mục mới trong /home/common. Nhận xét ?.

# *Bài 5* SỬ DỤNG TRÌNH SOẠN THẢO VI

Giới thiệu trình soạn thảo vi, các thao tác soạn thảo tập tin bằng vi.

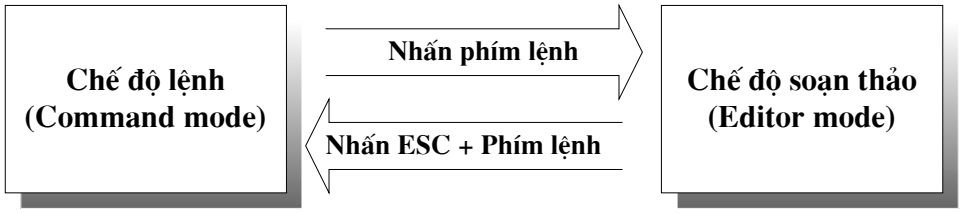
## 5.1. Giới thiệu

*vi* là chương trình soạn thảo các tập tin văn bản trên các hệ thống Unix :

- Màn hình được xem như một cửa sổ mở trên tập tin.
- Có khả năng di chuyển con trỏ đến bất kỳ vị trí nào trên màn hình.
- Cửa sổ có thể di chuyển tự do trên tập tin.

Phần lớn các phím dùng độc lập hoặc kết hợp với phím Shift và Ctrl để tạo ra các lệnh của vi. Các lệnh của vi có thể được gọi khi có dấu " : " ở dòng cuối màn hình.

Có 2 chế độ (mode) trong khi sử dụng **vi:** *Append mode* và *Command mode*

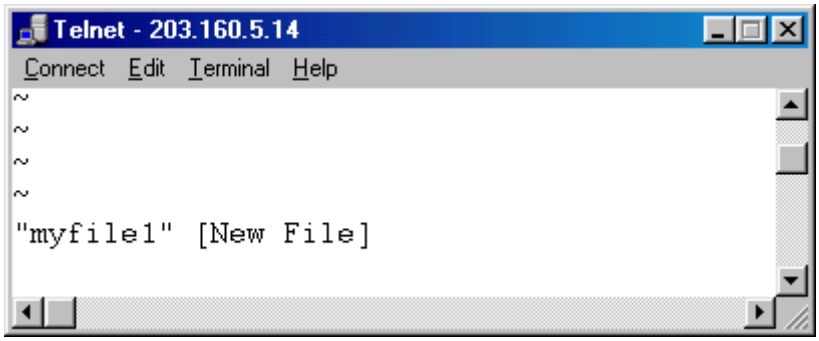



## 5.2. Khởi động vi

Ta có thể gọi *vi* với tên tập tin văn bản : **$ vi filename**

*Ví dụ :* **vi bai1.txt <*Enter*>**

Màn hình soạn thảo hiện ra như sau (*ở đây đang dùng Telnet để nối vào UNIX*) :




- Dấu ngã (~) trước mỗi dòng cho biết dòng đó còn rỗng (trống)
- Dòng dưới cùng cho biết tên file đang mở, trạng thái của file: nếu là file mới thì "[new file]", nếu mở file cũ thì sẽ hiển thị số dòng, số ký tự trong file (hình dưới).

## 5.3. Soạn thảo văn bản

- Chèn ký tự trên một dòng a < text > < ESC >
- Sử dụng các phím soạn thảo văn bản.

- Nhấn phím ESC để kết thúc chế độ chèn văn bản.

## 5.4. Thoát khỏi vi

Muốn ra khỏi vi và ghi lại nội dung tập tin, bạn **nhấn phím ESC** và dùng một trong các lệnh như sau:

**:ZZ** hoặc **:wq** hoặc **:x**

Thoát khỏi vi và không ghi lại các thay đổi trước đó

**: q!**

Khi ở trong chế độ soạn thảo của ***vi***, muốn chạy chương trình shell, dùng lệnh :

**: ! <shellcommand>**

hoặc gọi shell, sau đó chạy các lệnh của người dùng, khi kết thúc bấm Ctrl-D để trở lại ***vi***:

```
: ! sh
$ <command>
$ Ctrl-D
```

### 5.4.1. Dùng vi với danh sách các lệnh đã chạy của Shell

Lệnh **fc** (*fix command*) cho phép ta soạn thảo bằng *vi* và chạy lại các lệnh đã chạy của Shell. Cách dùng như sau:

- Soạn thảo và cho chạy lệnh cuối cùng: **$fc**
- Soạn thảo một nhóm lệnh và cho chạy: **$ fc m n**
- Xem danh sách 16 lệnh cuối cùng:

**$ fc -l** hoặc history

**$ fc -lr** (danh sách theo thứ tự ngược lại)

- Tạo một tập tin chứa một số lệnh đã chạy (của history):
  **$fc -nl n1 n2 > cmd**

***cmd*** là một tập tin chứa các lệnh của history từ lệnh **n1** đến lệnh **n2**

**Bảng tóm tắt các lệnh của *vi***

| | |
|---|---|
| <i> | Inserts text before cursor |
| <I> | Enters text at start of line |
| <a> | Inserts text after cursor |
| <A> | Enters text at end of line |
| <o> | Opens a new line below cursor |
| <O> | Opens a new line above cursor |
| <d><w> | Deletes word |
| <d><d> | Deletes entire line |

| | |
|---|---|
| <D> | Deletes to end of line |
| <x> | Deletes character under cursor |
| <c><w> | Changes word |
| <c><c> | Changes line |
| <C> | Changes to end of line |
| <R> | Replaces character under cursor |
| <J> | Joins lines together |
| <e> | Moves to end of word |
| <w> | Moves to next word |
| <$> | Moves to end of line |
| <l> | Moves one space right |
| <k> | Moves one line up |
| <j> | Moves one line down |
| <h> | Moves one space left |
| <f><x> | Moves cursor to first occurrence of x |
| <F><x> | Moves cursor to last occurrence of x |
| <;> | Repeats the last f/F command |
| *number*<\|> | Moves cursor to specified column *number* |
| <H> | Moves cursor to top line on-screen (not top line of file) |
| <L> | Moves cursor to bottom line on-screen |
| <M> | Moves cursor to middle line on-screen |
| <G> | Moves cursor to bottom line of file |
| *number*<G> | Moves cursor to specified line *number* (same as<ESC>:*number*) |
| <^> | Moves to beginning of line |
| <m>*x* | Marks current position with letter *x* |
| <Ctrl-d> | Scrolls for ward one half of the screen |
| <Ctrl-u> | Scrolls backward one half of the screen |
| <Ctrl-f> | Scrolls for ward one screen |
| <Ctrl-b> | Scrolls backward one screen |
| <Ctrl-l> | Redraws the screen |
| <Ctrl-G> | Shows the filename, current line, and column number |
| <z><z> | Redraws the screen with current line in middle of screen |
| <y><y> | Yanks entire line into buffer |
| <p> | Puts contents of buffer below cursor |
| <P> | Puts contents of buffer above cursor |
| *x*"[*number*]"<yy> | Yanks the indicated number of lines into the buffer named *x* (*x* can be any single character a–z) |

| | |
|---|---|
| *x*<p> | Places the contents of buffer *x* after the cursor |
| :w *[file]* | Writes contents to disk as *file* |
| *:q* | *Quits vi* |
| :q! | Quits file without saving changes |
| :wq | Saves changes and quits vi |
| :r *file* | Reads specified *file* into editor |
| :e *file* | Edits *file* |
| :!*command* | Executes specified shell *command* |
| :*number* | Moves to specified line number |
| :f | Prints out current line and filename (same as <Ctrl-G>) |
| /*string* | Searches forward for *string* |
| ?*string* | Searches backward for *string* |
| :*x*,*ys*/*oldstring*/*newstring* | Replaces *oldstring* with *newstring* from line *x* to line *y* (entering y = $ will replace to end of file) |
| <ESC><u> | Undoes last command |
| <n> | Finds next occurrence of string. Repeats last command |
| ~ | Changes character to opposite case |
| <ESC> | Switches to command mode |

## THỰC HÀNH

1. Dùng chương trình vi để soạn thảo tập tin vanban.doc

```
$vi  vanban.doc
```

2. Sao chép văn bản

| | |
|---|---|
| 4dd | Cắt 4 dòng và đưa vào vùng đệm |
| Ctrl+d | Chuyển xuống cuối văn bản |
| p | Sao từ vùng đệm vào sau dòng hiện hành |

3. Đặt và bỏ chế độ hiển thị số dòng :

```
:set  nu
:set  nonu
```

4. Lưu nội dung tập tin và thoát khỏi vi:

```
:wq
```

5. Xem lai nội dung tập tin vanban.doc.

*Bài 6* **LẬP TRÌNH SHELL**

## 6.1. Chương trình tính tổng 1-> n

- Minh họa các cấu trúc while do done, và cách sử dụng [], $(()).

- Tập tin **tong1.sh**

```
#!/bin/sh
echo "Chuong trinh tinh tong 1- $1"
index=0
tong=0
while [ $index -lt $1 ]
do
        index=$(($index + 1))
        tong=$(($tong + $index))
done
echo "Tong 1-$1= $tong"
exit 0
```

- Chạy chương trình :

```
chmod a+x tong1.sh
./tong1 100
```

## 6.2. Chương trình tính giai thừa của một số

- Minh họa các cấu trúc while do done, và cách sử dụng [], $(()).

- Tập tin **giaithua.sh**

```
#!/bin/sh
echo "Chuong trinh tinh $1!"
index=0
gt=1
while [ $index -lt $1 ]
do
        index=$(($index + 1))
        gt=$(($gt * $index))
done
echo "$1!= $gt"
exit 0
```

- Chạy chương trình :

```
chmod a+x giaithua.sh
./giaithua 5
```

## 6.3. Chương trình đếm số dòng của một tập tin

- Minh họa các cấu trúc if then fi, while do done, và cách sử dụng [], $(()).

- Tập tin **demdong.sh**

```
#!/bin/sh
echo "Chuong trinh dem so dong cua tap tin $1"
{
n=0
while read line
```

```
do
        n=$(($n + 1))
done
echo "So dong cua tap tin $1 la : $n"
}<$1
exit 0
```

- Chạy chương trình :

```
chmod a+x demdong.sh
./demdong bai1.txt
```

## 6.4. Chương trình đếm số từ của một tập tin

- Minh họa các cấu trúc for do done, while do done.

- Tập tin **demtu.sh**

```
#!/bin/sh
echo "Chuong trinh dem so tu cua tap tin $1"
{
n=0
while read line
do
        for wd in $line
        do
                n=$(($n + 1))
        done
done
echo "Tong so tu cua tap tin $1 la : $n"
}<$1
exit 0
```

- Chạy chương trình :

```
chmod a+x demtu.sh
./demtu bai1.txt
```

## 6.5. Chương trình tìm dòng có độ dài lớn nhất trong một tập tin

- Minh họa các cấu trúc if then fi, while do done.

- Tập tin **dongmax.sh**

```
#!/bin/sh
echo "Chuong trinh tim dong dai nhat trong tap tin $1"
{
n=0
max=0
dong=""
while read line
do
        n=`expr length "$line"`
        if [ $n –gt $max ]
        then
                dong="$line"
```

```
                max=$n
        fi
done
echo "Dong trong tap tin $1 co do dai max = $max la : $dong"
}<$1
exit 0
```

- Chạy chương trình :

```
chmod a+x dongmax.sh
./dongmax bai1.txt
```

## 6.6. Chương trình tìm một xâu trong một tập tin

- Minh họa các cấu trúc if then fi, while do done.

- Tập tin **timxau.sh**

```
#!/bin/sh
echo "Chuong trinh tim xau $1 trong tap tin $2"
{
wordlen=`expr length "$1"`                      # Do dai tu can tim
while read textline
do
        textlen=`expr length "$textline"`       # Do dai cua dong vua doc
        end=$(($textlen – wordlen + 1"
        index=1
        while [ $index –le $end ]
        do
                temp=`expr substr "$textline" $index $wordlen
                if [ "$temp" = $1 ]
                then
                        echo "Tim thay $1 tai dong $textline"
                        break
                fi
                index=$(($index + 1))
        done
done
}<$2
exit 0
```

- Chạy chương trình :

```
chmod a+x timxau.sh
./timxau abc bai1.txt
```

*Bài 7* **Lập trình C & C$^{++}$**

Trình biên dịch GNU là công cụ phát triển thông dụng nhất sẵn có trong hệ điều hành Linux, được dùng để biên dịch các kernael của hệ điều hành. Ngoài ra **gcc** cung cấp các thư viện và các tập tin Header cần thiết để biên dịch và chạy các chương trình của người dùng.

Các chương trình C thường có phần tên mở rộng là **.c**

Các chương trình C$^{++}$ thường có phần tên mở rộng là **.cc** các hoặc **.C**

Để biên dịch và thực thi một chương trình C bạn làm như sau :

1. Soạn thảo chương trình. Lưu tập tin với tên và phần mở rộng thích hợp.

**# vi example.c**

2. Thoát vi, từ dấu nhắc hệ thống bạn gõ lệnh :

***Cú pháp :*** **gcc -o filedestination filesource**

**#gcc -o hello hello.c**

3. Nếu có lỗi, trình biên dịch sẽ thông báo số thứ tự dòng lệnh lỗi. Nếu biên dịch thành công, để chạy chương trình gõ lệnh :

**#./filedestination**

*Ví dụ* **#./hello**

Lưu ý cách dùng ./ trước tên chương trình, nghĩa là máy sẽ chỉ tìm kiếm chương trình khả thi tại thư mục hiện hành.

Để dịch cùng một lúc nhiều tập tin chương trình trong thư mục hiện hành, bạn dùng lệnh : **make** hoặc **make all**

Sau đây là một số chương trình ví dụ :

1. Chương trình ***hello.c***

```
#include <stdio.h>
#include <math.h>
main()
{
int i;
double a;
for(i=1;i<11;i++) {
        a=i*1.0;
        printf("%2d. %3d %4d %7.5f\n",i,i*i,i*i*i);
}
}
```

***2.*** Chương trình ***sample.c***

```
#include <stdio.h>
void printnum ( int );              /* Khai bỏo hàm*/
void printchar ( char );            /* Khai bỏo hàm */
main () {
        double tmp;                         /* Khai bỏo bi?n toàn c?c */
        tmp = 1.234;
        printf ("%f\n",tmp);                /* In giỏ tr? c?a bi?n toàn c?c tmp */
        printnum (5);                       /* In giỏ tr? s? 5 */
        printf ("%f\n",tmp);                /* In giỏ tr? c?a bi?n toàn c?c tmp */
        printchar ('k');                    /* in ký t? k */
        printf ("%f\n",tmp);                /* In giỏ tr? c?a bi?n toàn c?c tmp */
}
/* é?nh nghia hàm dó khai bỏo ? trờn */
/* Khai bỏo cú t? khoỏ void nghia là hàm khụng tr? v? m?t giỏ tr? */
void printnum (int inputnum) {
        int tmp;
        tmp = inputnum;
        printf ("%d \n",tmp);
}
void printchar (char inputchar) {
        char tmp;
        tmp = inputchar;
        printf ("%c \n",tmp);
}
```

*Bài 8* **QUẢN LÝ TIẾN TRÌNH**

## 8.1. Giới thiệu

Tiến trình là một môi trường thực hiện, bao gồm một phân đoạn lệnh và một phân đoạn dữ liệu. Cần phân biệt với khái niệm chương trình chỉ gồm tập hợp lệnh.

Trên hệ điều hành Linux, tiến trình được nhận biết thông qua số hiệu của tiến trình, gọi là ***pid***. Cũng như đối với user, nó có thể nằm trong nhóm. Vì thế để phân biệt ta nhận biết qua số hiệu nhóm gọi là ***pgrp***. Một số hàm của C cho phép lấy được những thông số này:

**int getpid()** /* trả về giá trị int là pid của tiến trình hiện tại*/

**int getppid()** /*trả về giá trị int là pid của tiến trình cha của tiến trình hiện tại */

**int getpgrp()** /* trả về giá trị int là số hiệu của nhóm tiến trình*/

**int setpgrp()** /*trả về giá trị int là số hiệu nhóm tiến trình mới tạo ra*/

Ví dụ:

Lệnh *:* **printf("Toi la tien trinh %d thuoc nhom %d",getpid(),getgrp());**

Kết quả sẽ là: **Toi là tien trinh 235 thuoc nhom 231**

### 8.1.1. Tạo một tiến trình - lệnh fork

**int fork()** tạo ra một tiến trình con. Giá trị trả lại là 0 cho tiến trình con và dấu hiệu pid cho tiến trình cha. Giá trị sẽ là -1 nếu không tạo được tiến trình mới.

Theo nguyên tắc cơ bản của hệ thống, tiến trình con và cha sẽ có cùng đoạn mã. Đoạn dữ liệu của tiến trình mới là một bản sao chép chính xác đoạn dữ liệu của tiến trình cha. Tuy nhiên tiến trình con vẫn khác tiến trình cha ở pid, thời gian xử lý, ...

### 8.1.2. Dừng một tiến trình

Lệnh kill của Shell có thể dùng để chấm dứt hoạt động của một tiến trình. ví dụ như khi muốn dừng tiến trình 234 ta dùng lệnh: **kill 234**

C cũng có lệnh kill như sau:

**int kill(pid, sig);**

**int pid;** là dấu hiệu nhận biết của một tiến trình.

**int sig;** hằng tín hiệu giao tiếp tiến trình.

### 8.1.3. Giao tiếp giữa các tiến trình

Việc giao tiếp giữa các tiến trình được thực hiện thông qua các tín hiệu chuẩn của hệ thống. Tín hiệu là một sự ngắt quãng logic được gửi đến các tiến trình bởi hệ thống để thông báo cho chúng về những sự việc không bình thường trong môi trường hoạt động của chúng (như lỗi bộ nhớ, lỗi vào ra). Nó cũng cho phép các tiến trình liên lạc với nhau. Một tín hiệu (trừ SIGKILL) có thể được xem xét theo ba cách khác nhau:

1. Tiến trình có thể được bỏ qua: Ví dụ chương trình có thể bỏ qua sự ngắt quãng của người sử dụng hệ thống (đó là sự bỏ qua khi một tiến trình đang được sử dụng ở phần nền.
2. Tiến trình có thể được thực hiện: Trong trường hợp này, khi nhận được 1 tina stiệu, việc thực hiện 1 tiến trình được chuyển về một quy trình do người sử dụng xác định trước, sau đó trở lại nơi nó bị ngắt.
3. Lỗi có thể được tiến trình trả về sau khi nhận được tín hiệu này.

Dưới đây là một số tín hiệu thường gặp:

| | |
|---|---|
| **SIGHUP** | Tín hiệu này được phát đến các tiến trình vào lúc cuối khi mà nó tự ngắt. Nó cũng được phát đến mọi tiến trình có tiến trình chính tự ngắt. |
| **SIGINT** | Tín hiệu này được phát đến các tiến trình khi ta ra lệnh ngắt. |
| **SIGQUIT** | Tương tự như trên khi ta gõ vào ^D. |
| **SIGILL** | Lệnh không hợp lệ, tín hiệu được phát ra khi phát hiện 1 lệnh không đúng ở cấp độ vật lý (ví dụ như 1 tiến trình thực hiện một lệnh mà máy tính chông có lệnh này). |
| **SIGTRAP** | Tín hiệu được phát ra sau mỗi lệnh trong trường hợp tiến trình có sử dụng lệnh ptrace(). |
| **SIGIOT** | Bẫy được phát khi có các vấn đề về vật lý. |
| **SIGEMT** | Bẫy của lệnh phát, được phát ra khi có lỗi vật lý trong khi thực hiện. |
| **SIGFPE** | Được phát ra khi có lỗi về tính toán như một số có dấu phẩy nối có định dạng không hợp lý. Gần như luôn chỉ ra lỗi khi lập trình. |
| **SIGKILL** | Trang bị để kết thúc tiến trình. Không thể bỏ qua hoặc cắt tín hiệu này. |
| **SIGBUS** | Được phát khi gặp lỗi trên bus. |
| **SYSGEGV** | Được phát ra khi gặp lỗi trên phân đoạn sự truy cập dữ liệu bên ngoài phân đoạn dữ liệu được cấp phát cho tiến trình. |

| | |
|---|---|
| **SIGSYS** | Đối số không đúng cho hệ thống gọi. |
| **SIGPIPE** | Viết trên một ống dẫn không mở để đọc. |
| **SIGALRM** | Phát ra khi đồng hồ của một tiến trình ngừng lại. Đồng hồ được hoạt động bằng lệnh alrm(). |
| **SIGTERM** | Được phát ra khi một tiến trình kết thúc bình thường. Cũng có thể dùng để dừng 1 hệ thống để kết thúc tất cả các tiến trình hoạt động. |

### 8.1.4. Liên lạc giữa hai tiến trình

Từ một chương trình đơn giản dưới đây sử dụng các lệnh phát và nhận tín hiệu, sau đó giúp liên lạc giữa hai tiến trình.

Nội dung của ví dụ là sự liên lạc giữa một tiến trình cha và một tiến trình con thông qua các tín hiệu đã được trình bày phần trước.

```
#include       <errno. h>
#include       <signal. h>
void fils_atc()
{
  printf(" Tien trinh bi loai bo !!!\n");
  kill(getpid(), SIGINT);
}
/***********************************/
void fils()
{
  signal(SIGUSR1, fils_atc);
  printf(" Hinh thanh tien trinh moi. Nhung chuan bi loai bo tien trinh nay !!\n");
  while(1);
 }
/*******************************/
main()
{
int ppid, pid;

if ((pid = fork())==0) fils();
else
  {
    sleep(3);
     printf(" Chap nhan !! Tien trinh se bi loai bo.\n");
     kill(pid, SIGUSR1);
   }
}
```

Trong ví dụ trên, tiến trình con có sử dụng hàm signal(SIGUSR1, fils_atc). Hàm này có tác dụng mỗi khi tiến trình con nhận được tín hiệu SIGUSR1 thì hàm fils_atc() sẽ được thực thi.

Như vậy ở ví dụ trên một tiến trình con đã được tạo ra nhưng nó lại không muốn tiếp tục tồn tại. Do vậy sau khi tạm dừng lại sleep(3), tiến trình cha đã gởi đến cho tiến trình con một tín hiệu là SIGUSR1 bằng lệnh:

**kill(pid, SIGUSR1);**

ở tiến trình con, tín hiệu SIGUSR1 đã được gán với hàm fils_atc(). Hàm này ra một thông báo báo hiệu tiến trình này sắp chết rồi tự gởi đến chính mình (tiến trình con) tín hiệu SIGINT, tín hiệu ngắt tiến trình. Và tiến trình con đã chết.

**kill(getpid(), SIGINT);**

Một số nhược điểm khi liên lạc trực tiếp bằng tín hiệu:

- Một tín hiệu có thể bị bỏ qua, kết thúc một tiến trình hoặc bị chặn lại. Đó là lý do chính đưa ra các tín hiệu không thích ứng được để tiến hành liên lạc giữa các tiến trình. Một thông điệp điệp dưới hình thức tín hiệu có thể sẽ bị mất nếu nó được nhận lúc loại tín hiệu này tạm thời bị bỏ qua.
- Một vấn đề khác là các tín hiệu có quyền rất lớn, khi đến chúng làm ngắt quãng công việc hiện tại. Ví dụ việc nhận một tín hiệu trong khi tiến trình đang đợi một sự kiện (mà có thể đến khi sử dụng các lệnh open(), read(), ...) làm cho việc thực thi hàm bị chệch hướng. Khi trở lại, lệnh chính bị ngắt gởi lại một thông điệp báo lỗi mà hoàn toàn không xử lý được.

Ngoài việc liên lạc trực tiếp như ở ví dụ trên, còn cho phép một phương pháp liên lạc giữa các tiến trình  khác, đó là liên lạc qua "đường ống".

## 8.2. Lập trình đa tiến trình

### 8.2.1. ống dẫn liên lạc

ống dẫn là một cơ chế cơ bản để liên lạc gián tiếp giữa các tiến trình. Đó là các file đặc biệt (FIFO), ở đó các thông tin được truyền đi 1 đầu và thoát ra ở một đầu khác.

Một số đặc điểm của "ống dẫn":

- Các ống dẫn chỉ mang tính chất tạm thời, chỉ tồn tại trong thời gian thực hiện của một tiến trình tạo ra nó.
- Muốn tạo ra một ống dẫn phải bắt đầu bằng một lệnh đặc biệt: ***pipe()***.
- Nhiều tiến trình có thể viết và đọc trên cùng một ống dẫn. Tuy nhiên, không có một cơ chế nào để phân biệt thông tin cho các tiến trình ở đầu ra.
- Dung lượng ống dẫn bị hạn chế (khoảng 4KB). Do đó khi chúng ta cố gắng viết khi ống dẫn bị đầy thì sẽ gặp phải trường hợp tắc nghẽn.

- Các tiến trình liên lạc qua ống dẫn phải có mối quan hệ họ hàng và các ống dẫn nối phải được mở trước khi tạo ra các tiến trình con.
- Không thể tự thay đổi vị trí thông tin trong ống.

### 8.2.2. Thao tác với "ống dẫn liên lạc"

Tạo một ống dẫn:

```
int     p_desc[2];
int     pipe(p_desc);
```

Giá trị trả về là 0 nếu thành công, -1 nếu thất bại.

p_desc[0] : chứa các số hiệu mô tả nhờ đó có thể đọc trong ống dẫn.

p_desc[1] : chứa các số hiệu mô tả nhờ đó có thể viết trong ống dẫn.

Như vậy việc viết trong p_desc[1] là để truyền dữ liệu trong ống và việc đọc trong p_desc[0] để nhận chúng.

Ví dụ:

```
#include <errno. h>
#include <signal. h>
main()
{
int     i,ret, p_desc[2];
char    c;
pipe(p_desc);
write(p_desc[1], "AB", 2);
for (i=1; i<=3,i ++) {
        ret=read(p_desc[0], &c, 1);
        if (ret == 1)
          printf(" Gia tri:  %c\n",c);
        else
           perror("Loi ong dan rong");
        }
}
```

Ví dụ trên chỉ ra rằng ta có thể truyền và nhận thông tin trên ống dẫn. Chúng ta đã dùng hàm read() và write() để viết (truyền) và đọc (nhận) trên ống dẫn.

### 8.2.3. Liên lạc giữa tiến trình cha và tiến trình con

Trong ví dụ dưới đây, một tiến trình tạo ra một ống dẫn, tạo ra một tiến trình con, viết một văn bản vào ống dẫn.Tiến trình con thừa hưởng ống dẫn và các ký hiệu mô tả của ống dẫn, thực hiện đọc trong ống dẫn:

```
#include <errno. h>
#include <stdio. h>

void code_fils(int number)   {
 int     fd, nread;
char    texte[100];
```

```
fd=number;
printf(" So hieu mo ta la %d\n",fd);
switch (nread=read(fd, texte, sizeof(texte)))
{
case -1:
        perror("Loi doc.");
case 0:
        perror("EOF");
default:
        printf("Van ban nhan duoc co %d ky tu: %s\n",fd, texte);
}
}
main()  {
int     fd[2];
char    chaine[10];

if (pipe(fd)==-1)
        { perror("Loi khoi tao pipe.");
        exit(1);
}

switch (fork())  {
case    -1:
        perror(" Loi khoi tao tien trinh.");
        break;
case    0:
        if (close(fd[1])==-1)
        perror(" Error.");
        code_fils(fd[0]);
        exit(0);
}
close(fd[0]);
if (write(fd[1]),"hello",6)==-1)
        perror("Loi truyen.");
}
```

Kết quả chương trình:

```
So hieu mo ta la: 5
Van ban nhan duoc co 6 ky tu: hello
```

Chú ý rằng, tiến trình con đọc trong ống dẫn mà không viết ở đó nên nó bắt đầu bằng cách đóng phần viết fd[1] để tiết kiệm các tín hiệu mô tả của tổ hợp. Tương tự, vì tiến trình cha chỉ sử dụng phần viết nên nó đóng phần đọc lại (fd[0]). Sau đó tiến trình cha viết vào ống dẫn 6 ký tự và tiến trình con đã đọc chúng.

# *Bài 9* Lập trình mạng TCP/IP

## 9.1. Lập trình client /server theo giao thức TCP/IP

- ***Chương trình* tcpClient.c**

**/* Chuong trinh tcpClient.c */**

```
/* Khai báo các file thư viện cần thiết để gọi hàm socket*/

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>                    /*gethostbyname*/
#include <arpa/inet.h>
#include <netdb.h>
#include <stdio.h>
#include <unistd.h>                        /* close */

#define SERVER_PORT 1500
#define MAX_MSG 100

int main (int argc, char *argv[]) {

  /* Khởi tạo các biến dùng trong chương trình */
  int sd, rc, i;
  struct sockaddr_in localAddr, servAddr;
  struct hostent *h;

  if(argc < 3) {
    printf("usage: %s <IPserver> <data1> <data2> ... <dataN>\n",argv[0]);
    exit(1);
  }

  /* Hàm gethostbyname() lấy về địa chỉ IP theo tên nhập vào trong tập tin /etc/hosts */

  h = gethostbyname(argv[1]);
  if(h==NULL) {
    printf("%s: unknown host '%s'\n",argv[0],argv[1]);
    exit(1);
  }

  servAddr.sin_family = h->h_addrtype;
  memcpy((char *) &servAddr.sin_addr.s_addr, h->h_addr_list[0], h->h_length);
  servAddr.sin_port = htons(SERVER_PORT);

  /*     Gán các giá trị cho đối tượng socket.
         Tạo socket cho máy Client.  Lưu lại số mô tả socket */

  sd = socket(AF_INET, SOCK_STREAM, 0);
  if(sd<0) {
    perror("cannot open socket ");
    exit(1);
  }

  /*     Đặt tên socket cho chương trình Client
         Gán địa chỉ kết nối cho socket theo giao thức Internet */

  localAddr.sin_family = AF_INET;
  localAddr.sin_addr.s_addr = htonl(INADDR_ANY);
  localAddr.sin_port = htons(0);

 /* Hàm htons() dùng để chuyển đổi trật tự byte của số nguyên trước khi gởi đi – do hệ
    thống sử dụng cơ chế giao tiếp TCP/IP */

/* Ràng buộc tên với socket */
```

```
  rc = bind(sd, (struct sockaddr *) &localAddr, sizeof(localAddr));
  if(rc<0) {
    printf("%s: cannot bind port TCP %u\n",argv[0],SERVER_PORT);
    perror("error ");
    exit(1);
  }

  /* Thực hiện kết nối đến server theo tên/địa chỉ nhập vào từ dòng lệnh */

  rc = connect(sd, (struct sockaddr *) &servAddr, sizeof(servAddr));
  if(rc<0) {
    perror("cannot connect ");
    exit(1);
  }

  /* Sau khi socket đã kết nối, thực hiện gửi các dữ liệu đến chương trình Server */

  for(i=2;i<argc;i++) {

    rc = send(sd, argv[i], strlen(argv[i]) + 1, 0);

    if(rc<0) {
      perror("cannot send data ");
      close(sd);
      exit(1);

    }/* if */
    printf("%s: data%u sent (%s)\n",argv[0],i-1,argv[i]);

  }/* for */
return 0;
}/*main*/
```

- ***Chương trình*** **tcpServer.c**

```
/* Chuong trinh tcpServer.c */
/* Khai báo các file thư viện cần thiết để gọi hàm socket*/
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <stdio.h>
#include <unistd.h>                    /* close */


#define SUCCESS 0
#define ERROR   1

#define END_LINE 0x0
#define SERVER_PORT 1500
#define MAX_MSG 100

/* function readline */
int read_line();
```

```
int main (int argc, char *argv[]) {

  int sd, newSd, cliLen;

  struct sockaddr_in cliAddr, servAddr;
  char line[MAX_MSG];


  /*    Gán các giá trị cho đối tượng socket.
        Tạo socket cho máy Server. Lưu lại số mô tả socket */

  sd = socket(AF_INET, SOCK_STREAM, 0);
   if(sd<0) {
    perror("cannot open socket ");
    return ERROR;
  }

  /*    Đặt tên socket cho chương trình Server
        Gán địa chỉ kết nối cho socket theo giao thức Internet */
  servAddr.sin_family = AF_INET;
  servAddr.sin_addr.s_addr = htonl(INADDR_ANY);
  servAddr.sin_port = htons(SERVER_PORT);

  if(bind(sd, (struct sockaddr *) &servAddr, sizeof(servAddr))<0) {
    perror("cannot bind port ");
    return ERROR;
  }

  /*    Tạo hàng đợi lắng nghe kết nối của client
        Cho phép hàng đợi nhận tối đa 5 kết nối */
  listen(sd,5);

 /* Lặp liên tục chờ và lxy kết nối của client */
  while(1) {

    printf("%s: waiting for data on port TCP %u\n",argv[0],SERVER_PORT);
    cliLen = sizeof(cliAddr);

    /* Chấp nhận kết nối */
    newSd = accept(sd, (struct sockaddr *) &cliAddr, &cliLen);
    if(newSd<0) {
      perror("cannot accept connection ");
      return ERROR;
    }

    /* init line */
    memset(line,0x0,MAX_MSG);

    /* Đọc dữ liệu do Client gởi đến - xử lý dữ liệu nhận được  */
    while(read_line(newSd,line)!=ERROR) {

      printf("%s: received from %s:TCP%d : %s\n", argv[0], inet_ntoa(cliAddr.sin_addr),
                                                    ntohs(cliAddr.sin_port), line);

      /* init line */
      memset(line,0x0,MAX_MSG);
```

```
    } /* while(read_line) */

  } /* while (1) */

}


/* WARNING */
/* this function is experimental. I don't know yet if it works  */
/* correctly or not. Use Steven's readline() function to have  something robust.*/
/* rcv_line is my function readline(). Data is read from the socket when */
/* needed, but not byte after bytes. All the received data is read.      */
/* This means only one call to recv(), instead of one call for  each received byte. */
/* You can set END_CHAR to whatever means endofline for you. (0x0A is \n)*/
/* read_lin returns the number of bytes returned in line_to_return       */

/* Hàm có chức năng đọc dữ liệu từ socket*/
int read_line(int newSd, char *line_to_return) {

  static int rcv_ptr=0;
  static char rcv_msg[MAX_MSG];
  static int n;
  int offset;

  offset=0;

  while(1) {
    if(rcv_ptr==0) {
      /* read data from socket */
      memset(rcv_msg,0x0,MAX_MSG);              /* init buffer */
      n = recv(newSd, rcv_msg, MAX_MSG, 0);     /* wait for data */
      if (n<0) {
          perror(" cannot receive data ");
          return ERROR;
      } else if (n==0) {
          printf(" connection closed by client\n");
          close(newSd);
          return ERROR;
      }
    }

    /* if new data read on socket  OR if another line is still in buffer */
    /* copy line into 'line_to_return' */

    while(*(rcv_msg+rcv_ptr)!=END_LINE && rcv_ptr<n) {
      memcpy(line_to_return+offset,rcv_msg+rcv_ptr,1);
      offset++;
      rcv_ptr++;
    }

    /* end of line + end of buffer => return line */
    if(rcv_ptr==n-1) {
      /* set last byte to END_LINE */
      *(line_to_return+offset)=END_LINE;
      rcv_ptr=0;
      return ++offset;
    }
```

```
    /* end of line but still some data in buffer => return line */
    if(rcv_ptr <n-1) {
      /* set last byte to END_LINE */
      *(line_to_return+offset)=END_LINE;
      rcv_ptr++;
      return ++offset;
    }

    /* end of buffer but line is not ended => */
    /*  wait for more data to arrive on socket */
    if(rcv_ptr == n) {
      rcv_ptr = 0;
    }

  } /* while */
}/*main*/
```

## 9.2. Lập trình client /server theo giao thức UDP/IP

- ***Chương trình*** **udpClient.c**

```
/* udpClient.c */

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>                  /* memset() */
#include <sys/time.h>                /* select() */

#define REMOTE_SERVER_PORT 1500
#define MAX_MSG 100


int main(int argc, char *argv[]) {

  int sd, rc, i;
  struct sockaddr_in cliAddr, remoteServAddr;
  struct hostent *h;

  /* check command line args */
  if(argc<3) {
    printf("usage : %s <server> <data1> ... <dataN> \n", argv[0]);
    exit(1);
  }

  /* get server IP address (no check if input is IP address or DNS name */
  h = gethostbyname(argv[1]);
  if(h==NULL) {
    printf("%s: unknown host '%s' \n", argv[0], argv[1]);
    exit(1);
  }
```

```
  printf("%s: sending data to '%s' (IP : %s) \n", argv[0], h->h_name,
        inet_ntoa(*(struct in_addr *)h->h_addr_list[0]));

  remoteServAddr.sin_family = h->h_addrtype;
  memcpy((char *) &remoteServAddr.sin_addr.s_addr,
        h->h_addr_list[0], h->h_length);
  remoteServAddr.sin_port = htons(REMOTE_SERVER_PORT);

  /* socket creation */
  sd = socket(AF_INET,SOCK_DGRAM,0);
  if(sd<0) {
    printf("%s: cannot open socket \n",argv[0]);
    exit(1);
  }

  /* bind any port */
  cliAddr.sin_family = AF_INET;
  cliAddr.sin_addr.s_addr = htonl(INADDR_ANY);
  cliAddr.sin_port = htons(0);

  rc = bind(sd, (struct sockaddr *) &cliAddr, sizeof(cliAddr));
  if(rc<0) {
    printf("%s: cannot bind port\n", argv[0]);
    exit(1);
  }


  /* send data */
  for(i=2;i<argc;i++) {
    rc = sendto(sd, argv[i], strlen(argv[i])+1, 0,
                (struct sockaddr *) &remoteServAddr,
                sizeof(remoteServAddr));

    if(rc<0) {
      printf("%s: cannot send data %d \n",argv[0],i-1);
      close(sd);
      exit(1);
    }

  }

  return 1;

}
```

- ***Chương trình*** **udpServer.c**

```
/* udpServer.c */

#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <stdio.h>
#include <unistd.h> /* close() */
```

```
#include <string.h> /* memset() */

#define LOCAL_SERVER_PORT 1500
#define MAX_MSG 100

int main(int argc, char *argv[]) {

  int sd, rc, n, cliLen;
  struct sockaddr_in cliAddr, servAddr;
  char msg[MAX_MSG];

  /* Tạo socket trên máy Server - Đặt tên cho socket của chương trình Server */

  sd=socket(AF_INET, SOCK_DGRAM, 0);
  if(sd<0) {
    printf("%s: cannot open socket \n",argv[0]);
    exit(1);
  }

  /* bind local server port – ràng buộc tên với socket */

  servAddr.sin_family = AF_INET;
  servAddr.sin_addr.s_addr = htonl(INADDR_ANY);
  servAddr.sin_port = htons(LOCAL_SERVER_PORT);
  rc = bind (sd, (struct sockaddr *) &servAddr,sizeof(servAddr));

  if(rc<0) {
    printf("%s: cannot bind port number %d \n",  argv[0], LOCAL_SERVER_PORT);
    exit(1);
  }

  printf("%s: waiting for data on port UDP %u\n", argv[0],LOCAL_SERVER_PORT);

  /* Thực hiện vòng lặp vô hạn trên Server để chờ và xử lý kết nối đến từ máy client */
  while(1) {

    /* Khởi tạo bộ đệm */
    memset(msg,0x0,MAX_MSG);

    /* Nhận dữ liệu gởi đến từ client */
    cliLen = sizeof(cliAddr);
    n = recvfrom(sd, msg, MAX_MSG, 0, (struct sockaddr *) &cliAddr, &cliLen);

    if(n<0) {
      printf("%s: cannot receive data \n",argv[0]);
      continue;
    }

    /* In dữ liệu nhận được */
    printf("%s: from %s:UDP%u : %s \n",  argv[0],inet_ntoa(cliAddr.sin_addr),
                                              ntohs(cliAddr.sin_port),msg);

  }/*while*/
return 0;
}
```

*Bài 10*

# DỊCH VỤ TRUYỀN FILE FTP

FTP (File Transfer Protocol) là dịch vụ cho phép truyền các tập tin giữa hai máy tính Client và Server, quản lý các thư mục và truy cập vào thư tín điện tử. FTP không được thiết lập để truy cập vào một máy khác và chạy các chương trình ở máy đó, chỉ dùng cho việc truyền tập tin.

Để kết nối FTP, gõ lệnh sau : **ftp <IPAddressServer>**

| **Lệnh người dùng FTP** | **Mô tả** |
|---|---|
| **ascii** | Chuyển sang chế độ truyền ascii |
| **bell** | âm thanh của chương trình sau khi truyền mỗi tập tin |
| **binary** | Chuyển sang chế độ truyền nhị phân |
| **cd *directory*** | Chuyển đổi thư mục hiện hành trên server |
| **cdup** | Lùi thư mục hiện hành về một cấp trước đó |
| **close** | Huỷ kết nối |
| **delete *filename*** | Xoá một tập tin trên server |
| **dir *directory*** | Hiển thị thư mục *directory* của server |
| **get *filename*** | Truyền tập tin trên server về máy cục bộ |
| **hash** | Hiển thị/làm mất dấu # cho mỗi khối các ký tự đã truyền được |
| **help** | Hiển thị các trợ giúp |
| **lcd *directory*** | Chuyển đổi thư mục hiện hành trên máy cục bộ |
| **ls *directory*** | Xem danh sách các tập tin trong thư mục directory trên Server |
| **mdelete *files*** | Xóa nhiều tập tin trên máy Server |
| **mdir *directories*** | Liệt kê các tập tin trong nhiều thư mục trên máy Server |
| **mget *files*** | Tải nhiều tập tin trên máy Server về thư mục hiện hành của máy cục bộ |
| **mkdir <*directory*>** | Tạo thư mục trên máy Server |
| **mput *files*** | Gửi một số tập tin từ máy cục bộ lên máy Server |
| **open *host*** | Kết nối với Server host từ xa |
| **put *filename*** | Truyền tập tin từ máy cục bộ lên máy Server |
| **pwd** | Hiển thị thư mục hiện hành trên server |

| | |
|---|---|
| **status** | Hiển thị trạng thái của ftp |
| **rename *file1 file2*** | Đổi tên *file1* trên máy Server thành *file2* |
| **quote** | Cung cấp một lệnh FTP một cách trực tiếp |
| **quit** | Chấm dứt kết nối và thoát khỏi ftp |
| ? | Hiển thị danh sách lệnh |

Khi truy cập vào hệ thống, nếu chưa có account, người dùng có thể login với account đặc biệt là **anonymous**, không có mật khẩu.

## Thực hành

| | |
|---|---|
| C:\>ftp ↵ | Khởi động ftp từ thư mục hiện hành **C:** |
| (to) : 200.201.202.180 | |
| user : user01 | Nhập vào tên user |
| Password : | Nhập vào mật khẩu tương ứng |
| ftp> dir | Xem nội dung thư mục |
| ftp> ? | Xem nội dung các lệnh của ftp |
| ftp>put autoexec.bat autoexec.dos | Chuyển tập tin từ Client lên Server với tên mới là autoexec.dos |
| ftp> ls | Xem kết quả truyền file |
| ftp>get autoexec.dos LINUX.TXT | Lấy tập tin autoexec.dos trên Server về Client với tên mới là LINUX.TXT |
| ftp>mget autoexec.dos | Lấy tập tin autoexec.dos trên Server về Client thư mục C:\ |
| ftp>cd /home/user01 | Chuyển đến thư mục hiện hành là user01 là thư mục có toàn quyền của user user01 |
| ftp>mdir document | Tạo trong thư mục user01 thư mục mới có tên document |
| ftp> help dir | Xem hướng dẫn sử dụng lệnh dir |
| ftp>help get | Xem hướng dẫn sử dụng lệnh get |
| ftp> quit | Kết thúc phiên làm việc |

*Bài 11* **CÁC TẬP TIN CẤU HÌNH MẠNG**

## 1. Tập tin /etc/hosts

```
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1       localhost.localdomain   localhost
200.201.202.1   linuxsvr.dng.vn linuxsvr
```

## 2. Tập tin /etc/sysconfig/network

```
NETWORKING=yes
FORWARD_IPV4=false
HOSTNAME=linuxsvr.edu.vn
DOMAIN=edu.vn
GATEWAY=200.201.202.1
```

## 3. Tập tin /etc/sysconfig/network-scripts/ifcfg-eth0

```
DEVICE=eth0
BOOTPROTO=none
ONBOOT=yes
USERCTL=no
PEERDNS=no
TYPE=Ethernet
IPADDR=200.201.202.1
NETMASK=255.255.255.0
NETWORK=200.201.202.0
BROADCAST=200.201.202.255
```

## 4. Chạy chương trình X- Windows hỗ trợ cấu hình hệ thống :

**redhat-config-network**

## 5. Khởi động lại dịch vụ mạng

**[root@linuxsvr root]#/etc/init.d/network restart**

```
Shutting down interface eth0:                [  OK  ]
Shutting down loopback interface:            [  OK  ]
Setting network parameters:                  [  OK  ]
Bringing up loopback interface:              [  OK  ]
Bringing up interface eth0:                  [  OK  ]
```

## 6. Kiểm tra bằng lệnh :

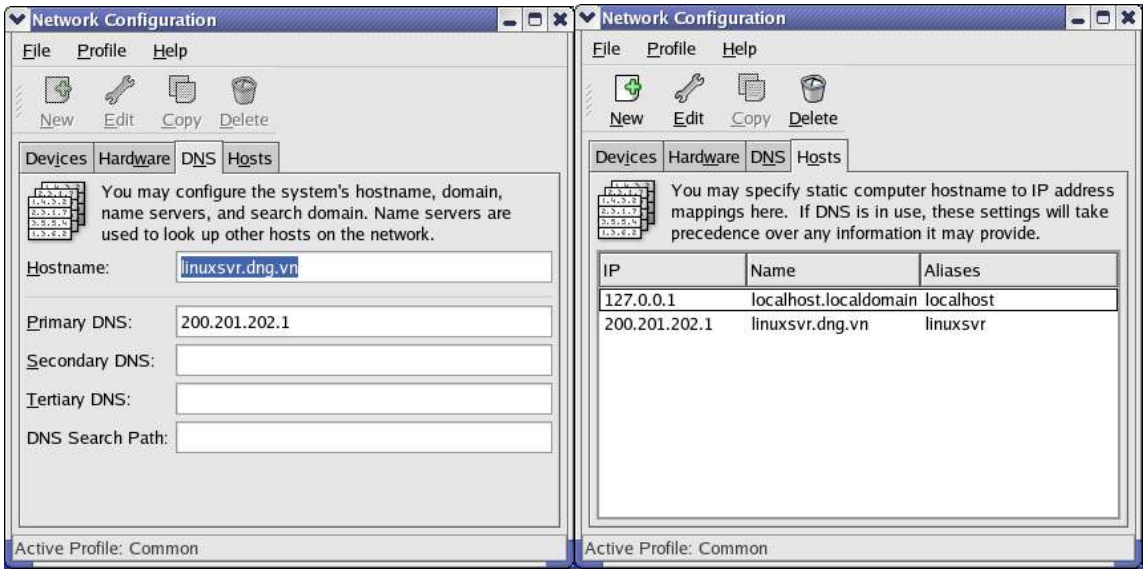
[root@linuxsvr root]#**hostname**
linuxsvr.dng.vn

## 7. Xem thông tin về cấu hình thiết bị mạng

```
[root@linuxsvr root]#ifconfig
eth0      Link encap:Ethernet  HWaddr 00:06:7B:02:71:21
          inet addr:200.201.202.1  Bcast:200.201.202.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2326 errors:0 dropped:0 overruns:0 frame:0
          TX packets:70927 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:218392 (213.2 Kb)  TX bytes:6939053 (6.6 Mb)
          Interrupt:9 Base address:0x4c00

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:933 errors:0 dropped:0 overruns:0 frame:0
          TX packets:933 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:87261 (85.2 Kb)  TX bytes:87261 (85.2 Kb)
```

---




Hình 1. Cấu hình dịch vụ mạng bằng tiện ích redhat-config-network.

*Bài 12* **CẤU HÌNH DỊCH VỤ DNS**

## 12.1. Các tập tin cấu hình dịch vụ DNS

### *12.1.1. Tập tin /etc/host.conf*

```
order hosts,bind
```

### *12.1.2. Tập tin /etc/resolv.conf*

```
:search dng.vn
nameserver 200.201.202.1
```

### *12.1.3. Tập tin /etc/named.conf*

```
# named.conf - configuration for bind
# Generated automatically by redhat-config-bind, alchemist et al.
# Any changes not supported by redhat-config-bind should be put
# in /etc/named.custom
controls {
      inet 127.0.0.1 allow { localhost; } keys { rndckey; };
};
include "/etc/named.custom";
include "/etc/rndc.key";
zone  "0.0.127.in-addr.arpa" {
      type master;
      file  "0.0.127.in-addr.arpa.zone";
};
zone  "localhost" {
      type master;
      file  "localhost.zone";
};
zone  "dng.vn" {
      type master;
      file  "dng.vn.zone";
};
zone  "edu.vn" {
      type master;
      file  "edu.vn.zone";
};
```

### *12.1.4. Tập tin /var/named/dng.vn.zone*

```
$TTL 86400
@      IN      SOA     dng.  root.localhost (
                       1 ; serial
                       28800 ; refresh
                       7200 ; retry
                       604800 ; expire
                       86400 ; ttl
                       )
       IN      NS      200.201.202.1.
```

```
www                     IN      A       200.201.202.1
tankhoi01       IN      A       200.201.202.1
tankhoi02       IN      A       200.201.202.2
```

### *12.1.5. Tập tin /var/named/edu.vn.zone*

```
$TTL 86400
@       IN      SOA     edu.  root.localhost (
                        2 ; serial
                        28800 ; refresh
                        7200 ; retry
                        604800 ; expire
                        86400 ; ttl
                        )
                       IN      NS      200.201.202.1.
www                     IN      A       200.201.202.1
tankhoi01       IN      A       200.201.202.1
tankhoi02       IN      A       200.201.202.2
```

### *12.1.6. Tập tin /var/named/0.0.127.in-addr.arpa.zone*

```
$TTL 86400
@       IN      SOA     localhost.      root.linuxsvr.dng.vn (
                        36 ; serial
                        28800 ; refresh
                        7200 ; retry
                        604800 ; expire
                        86400 ; ttk
                        )
@       IN      NS      localhost.
1       IN      PTR     localhost.
1       IN      PTR     www.
1       IN      PTR     tankhoi01.
2       IN      PTR     tankhoi02.
1       IN      PTR     www.
1       IN      PTR     tankhoi01.
2       IN      PTR     tankhoi02.
```

### *12.1.7. Tập tin /var/named/localhost.zone*

```
$TTL 86400
@       IN      SOA     @  root.localhost (
                        1 ; serial
                        28800 ; refresh
                        7200 ; retry
                        604800 ; expire
                        86400 ; ttl
                        )
      IN      NS      localhost.
@       IN      A       127.0.0.1
```

### *12.1.8. Lệnh khởi động dịch vụ DNS*

**/etc/init.d/named restart**

## 12.2. Các lệnh và tiện ích hỗ trợ

### *12.2.1. Lệnh nslookup*

**#nslookup**
Note: nslookup is deprecated and may be removed from future releases.
Consider using the `dig' or `host' programs instead. Run nslookup with
the `-sil[ent]' option to prevent this message from appearing.
> www.dng.vn
Server: 200.201.202.1
Address: 200.201.202.1#53

Name: www.dng.vn
Address: 200.201.202.1
> tankhoi02.edu.vn
Server: 200.201.202.1
Address: 200.201.202.1#53

Name: tankhoi02.edu.vn
Address: 200.201.202.2

### *12.2.2. Lệnh host*

**#host tankhoi01.dng.vn**

tankhoi01.dng.vn has address 200.201.202.1

### *12.2.3. Lệnh dig*

**# dig dng.vn**
; <<>> DiG 9.2.1 <<>> dng.vn
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 58922
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0

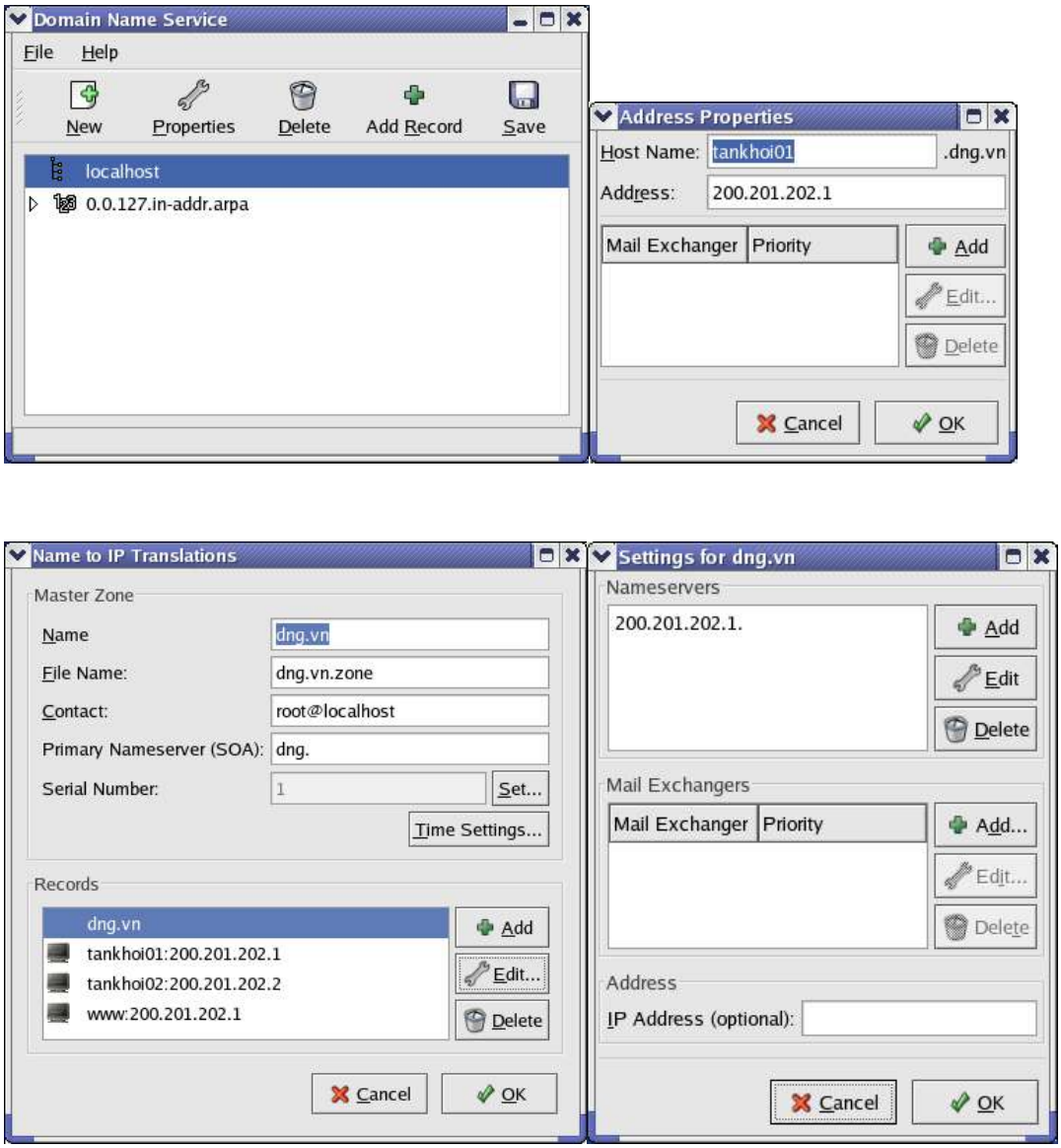
;; QUESTION SECTION:
;dng.vn. IN A

;; AUTHORITY SECTION:
dng.vn. 86400 IN SOA dng. root.localhost.dng.vn. 1 28800 7200
604800 86400

;; Query time: 28 msec
;; SERVER: 200.201.202.1#53(200.201.202.1)
;; WHEN: Mon Mar 22 09:14:13 2004
;; MSG SIZE rcvd: 78

### 12.2.4. Tiện ích redhat-config-bind

**#redhat-config-bind**




Hình 2. Cấu hình dịch vụ BIND bằng tiện ích redhat-config-bind.

---

**@2004, Nguyễn Tấn Khôi**

**Khoa CNTT Trường Đại học Bách Khoa Đà Nẵng**