ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

==========

HÀ QUANG THỤY - NGUYỄN TRÍ THÀNH

Giáo trình:

HỆ ĐIỀU HÀNH UNIX - LINUX

Dành cho sinh viên ngành Công nghệ thông tin,
Điện tử - Viễn thông, Toán tin ứng dụng

HÀ NỘI - 2004

MỤC LỤC

LỜI GIỚI THIỆU

CHƯƠNG 1. GIỚI THIỆU CHUNG VỀ LINUX

1.1. Giới thiệu về UNIX và Linux

1.1.1. Xuất xứ, quá trình tiến hóa và một số đặc trưng của hệ điều hành UNIX

Năm 1965, Viện công nghệ Massachusetts (MIT: Massachusetts Institute of Technology) và Phòng thí nghiệm Bell của hãng AT&T thực hiện dự án xây dựng một hệ điều hành có tên gọi là Multics (MULTiplexed Information and Computing Service) với mục tiêu: tạo lập được một hệ điều hành phủ trên vùng lãnh thổ rộng (hoạt động trên tập các máy tính được kết nối), đa người dùng, có năng lực cao về tính toán và lưu trữ. Dự án nói trên thành công ở mức độ hết sức khiêm tốn và người ta đã biết đến một số khiếm khuyết khó khắc phục của Multics.

Năm1969, Ken Thompson, một chuyên viên tại phòng thí nghiệm Bell, người đã tham gia dự án Multics, cùng Dennics Richie viết lại hệ điều hành đa-bài toán trên máy PDP-7 với tên là UNICS (UNiplexed Information and Computing Service) từ một câu gọi đùa của một đồng nghiệp. Trong hệ điều hành UNICS, một số khởi thảo đầu tiên về Hệ thống file đã được Ken Thompson và Dennis Ritchie thực hiện. Đến năm 1970 hệ điều hành được viết trên assembler cho máy PDP-11/20 và mang tên là UNIX.

Năm 1973, Riche và Thompson viết lại nhân của hệ điều hành UNIX trên ngôn ngữ C, và hệ điều hành đã trở nên dễ dàng cài đặt tới các loại máy tính khác nhau; tính chất như thế được gọi là tính khả chuyển (portable) của UNIX. Trước đó, khoảng năm 1971, hệ điều hành được thể hiện trên ngôn ngữ B (mà dựa trên ngôn ngữ B, Ritche đã phát triển thành ngôn ngữ C).

Hãng AT&T phổ biến chương trình nguồn UNIX tới các trường đại học, các công ty thương mại và chính phủ với giá không đáng kể.

Năm 1982, hệ thống UNIX-3 là bản UNIX thương mại đầu tiên của AT&T.

Năm 1983, AT&T giới thiệu Hệ thống UNIX-4 phiên bản thứ nhất trong đó đã có trình soạn thảo **vi**, thư viện quản lý màn hình được phát triển từ Đại học Tổng hợp California, Berkley.

Giai đoạn 1985-1987, UNIX-5 phiên bản 2 và 3 tương ứng được đưa ra vào các năm 1985 và 1987. Trong giai đoạn này, có khoảng 100000 bản UNIX đã được phổ biến trên thế giới, cài đặt từ máy vi tính đến các hệ thống lớn.

Đầu thập kỷ 1990. UNIX-5 phiên bản 4 được đưa ra như là một chuẩn của UNIX. Đây là sự kết hợp của các bản sau:

- AT&T UNIX-5 phiên bản 3,
- Berkley Software Distribution (BSD),
- XENIX của MicroSoft
- SUN OS

Có thể tìm thấy các nội dung liên quan tới một số phiên bản mới của UNIX tại địa chỉ website http://problem.rice.edu/.

Các nhóm nhà cung cấp khác nhau về UNIX đang hoạt động trong thời gian hiện nay được kể đến như sau:

- Unix International (viết tắt là UI). UI là một tổ chức gồm các nhà cung cấp thực hiện việc chuyển nhượng hệ thống UNIX-5 và cung cấp bản AT&T theo các

nhu cầu và thông báo phát hành mới, chẳng hạn như điều chỉnh bản quyền. Giao diện đồ họa người dùng là Open Look.

- Open Software Foundation (OSF). OSF được hỗ trợ bởi IBM, DEC, HP ... theo hướng phát triển một phiên bản của Unix nhằm tranh đua với hệ thống UNIX-5 phiên bản 4. Phiên bản này có tên là OSF/1 với giao diện đồ họa người dùng được gọi là MOTIF.
- Free SoftWare Foundation (FSF): một cộng đồng do Richard Stallman khởi xướng năm 1984 chủ trương phát hành các phần mềm sử dụng tự do, trên cơ sở một hệ điều hành thuộc loại UNIX.

Bảng sau đây liệt kê một số cài đặt UNIX khá phổ biến (thường thấy có chữ X ở cuối tên gọi của Hệ điều hành):

Tên hệ	*Nhà cung cấp*	*Nền phát triển*
AIX	International Business Machines	AT&T System V
A/UX	Apple Computer	AT&T System V
Dynix	Sequent	BSD (Berkeley SoftWare Distribution)
HP-UX	Hewlett-Packard	BSD
Irix	Silicon Graphics	AT&T System V
Linux	Free SoftWare Foundation	
NextStep	Next	BSD
OSF/1	Digital Equipment Corporation	BSD
SCO UNIX	Santa Cruz Operation	AT&T System V
Solaris	Sun Microsystems	AT&T System V
SunOS	Sun Microsystems	BSD UNIX
Ultrix	Digital Equipment Corporation	BSD UNIX
Unicos	Cray	AT&T System V
UnixWare	Novell	AT&T System V
XENIX	MicroSoft	AT&T System III-MS

Dưới đây liệt kê một số đặc trưng của hệ điều hành UNIX:

- Hệ điều hành được viết trên ngôn ngữ bậc cao; bởi vậy, rất dễ đọc, dễ hiểu, dễ thay đổi để cài đặt trên loại máy mới (tính dễ mang chuyển, như đã nói),
- Có giao diện người dùng đơn giản đủ năng lực cung cấp các dịch vụ mà người dùng mong muốn (so sánh với các hệ điều hành có từ trước đó thì giao diện của UNIX là một tiến bộ vượt bậc),
- Thỏa mãn nguyên tắc xây dựng các chương trình phức tạp từ những chương trình đơn giản hơn: trước hết có các môđun cơ bản nhất của nhân sau đó phát triển để có toàn bộ hệ điều hành,
- Sử dụng duy nhất một hệ thống File có cấu trúc cho phép dễ dàng bảo quản và sử dụng hiệu quả,

- Sử dụng phổ biến một dạng đơn giản trình bày nội tại của File như một dòng các byte cho phép dễ dàng khi viết các chương trình ứng dụng truy nhập, thao tác với các dữ liệu trong File,
- Có kết nối đơn giản với thiết bị ngoại vi: các file thiết bị đã được đặt sẵn trong Hệ thống File tạo ra một kết nối đơn giản giữa chương trình người dùng với các thiết bị ngoại vi,
- Là hệ điều hành đa người dùng, đa quá trình, trong đó mỗi người dùng có thể thực hiện các quá trình của mình một cách độc lập.
- Mọi thao tác vào - ra của hệ điều hành được thực hiện trên hệ thống File: mỗi thiết bị vào ra tương ứng với một file. Chương trình người dùng làm việc với file đó mà không cần quan tâm cụ thể tên file đó được đặt cho thiết bị nào trong hệ thống.
- Che khuất cấu trúc máy đối với người dùng, đảm bảo tính độc lập tương đối của chương trình đối với dữ liệu và phần cứng, tạo điều kiện thuận lợi hơn cho người lập trình khi viết các chương trình chạy UNIX với các điều kiện phần cứng hoàn toàn khác biệt nhau.

1.1.2. Giới thiệu sơ bộ về Linux

Linus Tovalds (một sinh viên Phần lan) đưa ra nhân (phiên bản đầu tiên) cho hệ điều hành Linux vào tháng 8 năm 1991 trên cơ sở cải tiến một phiên bản UNIX có tên Minix do Giáo sư Andrew S. Tanenbaum xây dựng và phổ biến. Nhân Linux tuy nhỏ song là tự đóng gói. Kết hợp với các thành phần trong hệ thống GNU, hệ điều hành Linux đã được hình thành. Và cũng từ thời điểm đó, theo tư tưởng GNU, hàng nghìn, hàng vạn chuyên gia trên toàn thế giới (những người này hình thành nên cộng đồng Linux) đã tham gia vào quá trình phát triển Linux và vì vậy Linux ngày càng đáp ứng nhu cầu của người dùng.
Dưới đây là một số mốc thời gian quan trọng trong quá trình hình thành và phát triển hệ điều hành Linux.

- Sau ba năm nhân Linux ra đời, đến ngày 14-3-1994, hệ điều hành Linux phiên bản 1.0 được phổ biến. Thành công lớn nhất của Linux 1.0 là nó đã hỗ trợ giao thức mạng TCP/IP chuẩn UNIX, sánh với giao thức socket BSD- tương thích cho lập trình mạng. Trình điều khiển thiết bị đã được bổ sung để chạy IP trên một mạng Ethernet hoặc trên tuyến đơn hoặc qua modem. Hệ thống file trong Linux 1.0 đã vượt xa hệ thống file của Minix thông thường, ngoài ra đã hỗ trợ điều khiển SCSI truy nhập đĩa tốc độ cao. Điều khiển bộ nhớ ảo đã được mở rộng để hỗ trợ điều khiển trang cho các file swap và ánh xạ bộ nhớ của file đặc quyền (chỉ có một ánh xạ bộ nhớ chỉ đọc được thi hành trong Linux 1.0).
- Vào tháng 3-1995, nhân 1.2 được phổ biến. Điều đáng kể của Linux 1.2 so với Linux 1.0 ở chỗ nó hỗ trợ một phạm vi rộng và phong phú phần cứng, bao gồm cả kiến trúc tuyến phần cứng PCI mới. Nhân Linux 1.2 là nhân kết thúc dòng nhân Linux chỉ hỗ trợ PC.

  Một điều cần lưu ý về các đánh chỉ số các dòng nhân (hệ điều hành) Linux. Hệ thống chỉ số được chia thành một số mức, chẳng hạn hai mức như 2.4 hoặc ba mức như 2.2.5. Trong cách đánh chỉ số như vậy, quy ước rằng với các chỉ số từ mức thứ hai trở đi, nếu là số chẵn thì dòng nhân đó đã khá ổn định và tương đối hoàn thiện, còn nếu là số lẻ thì dòng nhân đó vẫn đang được phát triển tiếp.

- Tháng 6-1996, nhân Linux 2.0 được phổ biến. Có hai đặc trưng nổi bật của Linux 2.0 là hỗ trợ kiến trúc phức hợp, bao gồm cả cổng Alpha 64-bit đầy đủ, và hỗ trợ kiến trúc đa bộ xử lý. Phân phối nhân Linux 2.0 cũng thi hành được trên bộ xử lý Motorola 68000 và kiến trúc SPARC của SUN. Các thi hành của Linux dựa trên vi nhân GNU Mach cũng chạy trên PC và PowerMac.
- Tới năm 2000, nhân Linux 2.4 được phổ biến. Một trong đặc điểm được quan tâm của nhân này là nó hỗ trợ mã ký tự Unicode 32 bít, rất thuận lợi cho việc xây dựng các giải pháp toàn diện và triệt để đối với vấn đề ngôn ngữ tự nhiên trên phạm vi toàn thế giới.

Vấn đề phân phối và giấy phép Linux

Về lý thuyết, mọi người có thể khởi tạo một hệ thống Linux bằng cách tiếp nhận bản mới nhất các thành phần cần thiết từ các site ftp và biên dịch chúng. Trong thời kỳ đầu tiên, người dùng Linux phải tiến hành toàn bộ các thao tác này và vì vậy công việc là khá vất vả. Tuy nhiên, do có sự tham gia đông đảo của các cá nhân và nhóm phát triển Linux, đã tiến hành thực hiện nhiều giải pháp nhằm làm cho công việc khởi tạo hệ thống đỡ vất vả. Một trong những giải pháp điển hình nhất là cung cấp tập các gói chương trình đã tiền dịch, chuẩn hóa.

Những tập hợp như vậy hay những bản phân phối là lớn hơn nhiều so với hệ thống Linux cơ sở. Chúng thường bao gồm các tiện ích bổ sung cho khởi tạo hệ thống, các thư viện quản lý, cũng như nhiều gói đã được tiền dịch, sẵn sàng khởi tạo của nhiều bộ công cụ UNIX dùng chung, chẳng hạn như phục vụ tin, trình duyệt web, công cụ xử lý, soạn thảo văn bản và thậm chí các trò chơi.

Cách thức phân phối ban đầu rất đơn giản song ngày càng được nâng cấp và hoàn thiện bằng phương tiện quản lý gói tiên tiến. Các bản phân phối ngày nay bao gồm các cơ sở dữ liệu tiến hóa gói, cho phép các gói dễ dàng được khởi tạo, nâng cấp và loại bỏ.

Nhà phân phối đầu tiên thực hiện theo phương châm này là Slakware, và chính họ là những chuyển biến mạnh mẽ trong cộng đồng Linux đối với công việc quản lý gói khởi tạo Linux.

Tiện ích quản lý gói RPM (RedHat Package Manager) của công ty RedHat là một trong những phương tiện điển hình.

Nhân Linux là phần mềm tự do được phân phối theo Giấy phép sở hữu công cộng phần mềm GNU GPL.

Các thành phần tích hợp Hệ điều hành Linux

Linux sử dụng rất nhiều thành phần từ Dự án phần mềm tự do GNU, từ hệ điều hành BSD của Đại học Berkeley và từ hệ thống X-Window của MIT.

Thư viện hệ thống chính của Linux được bắt nguồn từ Dự án GNU, sau đó được rất nhiều người trong cộng đồng Linux phát triển tiếp, những phát triển tiếp theo như vậy chủ yếu liên quan tới việc giải quyết các vấn đề như thiếu vắng địa chỉ (lỗi trang), thiếu hiệu quả và gỡ rối. Một số thành phần khác của Dự án GNU, chẳng hạn như trình biên dịch GNU C (gcc), vốn là chất lượng cao nên được sử dụng nguyên xy trong Linux.

Các tool quản lý mạng được bắt nguồn từ mã 4.3BSD song sau đó đã được cộng đồng Linux phát triển, chẳng hạn như thư viện toán học đồng xử lý dấu chấm động Intel và các trình điều khiển thiết bị phần cứng âm thanh PC. Các tool quản lý mạng này sau đó lại được bổ sung vào hệ thống BSD.

Hệ thống Linux được duy trì gần như bởi một mạng lưới không chặt chẽ các nhà phát triển phần mềm cộng tác với nhau qua Internet, mạng lưới này gồm các nhóm nhỏ và cá nhân

chịu trách nhiệm duy trì tính toàn vẹn của từng thành phần. Một lượng nhỏ các site phân cấp ftp Internat công cộng đã đóng vai trò nhà kho theo chuẩn de facto để chứa các thành phần này. Tài liệu Chuẩn phân cấp hệ thống file (File System Hierarchy Standard) được cộng đồng Linux duy trì nhằm giữ tính tương thích xuyên qua sự khác biệt rất lớn giữa các thành phần hệ thống.

Một số đặc điểm chính của Linux

Dưới đây trình bày một số đặc điểm chính của của hệ điều hành Linux hiện tại:

- Linux tương thích với nhiều hệ điều hành như DOS, MicroSoft Windows ...:
- Cho phép cài đặt Linux cùng với các hệ điều hành khác trên cùng một ổ cứng. Linux có thể truy nhập đến các file của các hệ điều hành cùng một ổ đĩa. Linux cho phép chạy mô phỏng các chương trình thuộc các hệ điều hành khác.
- Do giữ được chuẩn của UNIX nên sự chuyển đổi giữa Linux và các hệ UNIX khác là dễ dàng.
- Linux là một hệ điều hành UNIX tiêu biểu với các đặc trưng là đa người dùng, đa chương trình và đa xử lý.
- Linux có giao diện đồ hoạ (GUI) thừa hưởng từ hệ thống X-Window. Linux hỗ trợ nhiều giao thức mạng, bắt nguồn và phát triển từ dòng BSD. Thêm vào đó, Linux còn hỗ trợ tính toán thời gian thực.
- Linux khá mạnh và chạy rất nhanh ngay cả khi nhiều quá trình hoặc nhiều cửa sổ.
- Linux được cài đặt trên nhiều chủng loại máy tính khác nhau như PC, Mini và việc cài đặt khá thuận lợi. Tuy nhiên, hiện nay chưa xuất hiện Linux trên máy tính lớn (mainframe).
- Linux ngày càng được hỗ trợ bởi các phần mềm ứng dụng bổ sung như soạn thảo, quản lý mạng, quản trị cơ sở dữ liệu, bảng tính ...
- Linux hỗ trợ tốt cho tính toán song song và máy tính cụm (PC-cluster) là một hướng nghiên cứu triển khai ứng dụng nhiều triển vọng hiện nay.
- Là một hệ điều hành với mã nguồn mở, được phát triển qua cộng đồng nguồn mở (bao gồm cả Free Software Foundation) nên Linux phát triển nhanh. Linux là một trong một số ít các hệ điều hành được quan tâm nhiều nhất trên thế giới hiện nay.
- Linux là một hệ điều hành hỗ trợ đa ngôn ngữ một cách toàn diện nhất. Do Linux cho phép hỗ trợ các bộ mã chuẩn từ 16 bit trở lên (trong đó có các bộ mã ISO10646, Unicode) cho nên việc bản địa hóa trên Linux là triệt để nhất trong các hệ điều hành.

Tuy nhiên cũng tồn tại một số khó khăn làm cho Linux chưa thực sự trở thành một hệ điều hành phổ dụng, dưới đây là một số khó khăn điển hình:

- Tuy đã có công cụ hỗ trợ cài đặt, tuy nhiên, việc cài đặt Linux còn tương đối phức tạp và khó khăn. Khả năng tương thích của Linux với một số loại thiết bị phần cứng còn thấp do chưa có các trình điều khiển cho nhiều thiết bị,
- Phần mềm ứng dụng chạy trên nền Linux tuy đã phong phú song so với một số hệ điều hành khác, đặc biệt là khi so sánh với MS Windows, thì vẫn còn có khoảng cách.

Với sự hỗ trợ của nhiều công ty tin học hàng đầu thế giới (IBM, SUN, HP ...) và sự tham gia phát triển của hàng vạn chuyên gia trên toàn thế giới thuộc cộng đồng Linux, các khó khăn của Linux chắc chắn sẽ nhanh chóng được khắc phục.

Chính vì lẽ đó đã hình thành một số nhà cung cấp Linux trên thế giới. Bảng dưới đây là tên của một số nhà cung cấp Linux có tiếng nhất và địa chỉ website của họ.

Đáng chú ý nhất là Red Hat Linux (tại Mỹ) và Red Flag Linux (tại Trung Quốc). Red Hat được coi là lâu đời và tin cậy, còn Red Flag là một công ty Linux của Trung quốc, có quan hệ với cộng đồng Linux Việt nam và chúng ta có thể học hỏi một cách trực tiếp kinh nghiệm cho quá trình đưa Linux vào Việt nam.

Tên công ty	***Địa chỉ website***
Caldera OpenLinux	www.caldera.com
Corel Linux	www.corel.com
Debian GNU/Linux	www.debian.com
Linux Mandrake	www.mandrake.com
Red Hat Linux	www.redhat.com
Red Flag Linux	www.redflag-linux.com
Slackware Linux	www.slackware.com
SuSE Linux	www.suse.com
TurboLinux	www.turbolinux.com

1.2. Sơ bộ về các thành phần của Linux

Hệ thống Linux, được thi hành như một hệ điều hành UNIX truyền thống, gồm shell và ba thành phần (đã dạng mã chương trình) sau đây:

- Nhân hệ điều hành chịu trách nhiệm duy trì các đối tượng trừu tượng quan trọng của hệ điều hành, bao gồm bộ nhớ ảo và quá trình. Các mô đun chương trình trong nhân được đặc quyền trong hệ thống, bao gồm đặc quyền thường trực ở bộ nhớ trong.
- Thư viện hệ thống xác định một tập chuẩn các hàm để các ứng dụng tương tác với nhân, và thi hành nhiều chức năng của hệ thống nhưng không cần có các đặc quyền của mô đun thuộc nhân. Một hệ thống con điển hình được thi hành dựa trên thư viên hệ thống là hệ thống file Linux.
- Tiện ích hệ thống là các chương trình thi hành các nhiệm vụ quản lý riêng rẽ, chuyên biệt. Một số tiện ích hệ thống được gọi ra chỉ một lần để khởi động và cấu hình phương tiện hệ thống, một số tiện ích khác, theo thuật ngữ UNIX được gọi là trình chạy ngầm (daemon), có thể chạy một cách thường xuyên (thường theo chu kỳ), điều khiển các bài toán như hưởng ứng các kết nối mạng mới đến, tiếp nhận yêu cầu logon, hoặc cập nhật các file log.

Tiện ích (hay lệnh) có sẵn trong hệ điều hành (dưới đây tiện ích được coi là lệnh thường trực). Nội dung chính yếu của tài liệu này giới thiệu chi tiết về một số lệnh thông dụng nhất của Linux. Hệ thống file sẽ được giới thiệu trong chương 3. Trong các chương sau có đề cập tới nhiều nội dung liên quan đến nhân và shell, song dưới đây là một số nét sơ bộ về chúng.

1.2.1. Sơ bộ về nhân

Nhân (còn được gọi là hệ lõi) của Linux, là một bộ các môdun chương trình có vai trò điều khiển các thành phần của máy tính, phân phối các tài nguyên cho người dùng (các quá trình người dùng). Nhân chính là cầu nối giữa chương trình ứng dụng với phần cứng.

Người dùng sử dụng bàn phím gõ nội dung yêu cầu của mình và yêu cầu đó được nhân gửi tới shell: Shell phân tích lệnh và gọi các chương trình tương ứng với lệnh để thực hiện.

Một trong những chức năng quan trọng nhất của nhân là giải quyết bài toán lập lịch, tức là hệ thống cần phân chia CPU cho nhiều quá trình hiện thời cùng tồn tại. Đối với Linux, số lượng quá trình có thể lên tới con số hàng nghìn. Với số lượng quá trình đồng thời nhiều như vậy, các thuật toán lập lịch cần phải đủ hiệu quả: Linux thường lập lịch theo chế độ Round Robin (RR) thực hiện việc luân chuyển CPU theo lượng tử thời gian.

Thành phần quan trọng thứ hai trong nhân là hệ thống các môđun chương trình (được gọi là lời gọi hệ thống) làm việc với hệ thống file. Linux có hai cách thức làm việc với các file: làm việc theo byte (kí tự) và làm việc theo khối. Một đặc điểm đáng chú ý là file trong Linux có thể được nhiều người cùng truy nhập tới nên các lời gọi hệ thống làm việc với file cần đảm bảo việc file được truy nhập theo quyền và được chia xẻ cho người dùng.

1.2.2. Sơ bộ về shell

Một số nội dung chi tiết về shell (còn được gọi là ***hệ vỏ***) trong Linux được trình bày trong chương "Lập trình trên shell". Những nội dung trình bày dưới đây cung cấp một cách nhìn sơ bộ về shell và vai trò của nó trong hoạt động chung của hệ điều hành.

Người dùng mong muốn máy tính thực hiện một công việc nào đó thì cần gõ lệnh thể hiện yêu cầu của mình để hệ thống đáp ứng yêu cầu đó. Shell là bộ dịch lệnh và hoạt động như một kết nối trung gian giữa nhân với người dùng: Shell nhận dòng lệnh do người dùng đưa vào; và từ dòng lệnh nói trên, nhân tách ra các bộ phận để nhận được một hay một số lệnh tương ứng với các đoạn văn bản có trong dòng lệnh. Một lệnh bao gồm tên lệnh và tham số: từ đầu tiên là tên lệnh, các từ tiếp theo (nếu có) là các tham số. Tiếp theo, shell sử dụng nhân để khởi sinh một quá trình mới (khởi tạo quá trình) và sau đó, shell chờ đợi quá trình con này tiến hành, hoàn thiện và kết thúc. Khi shell sẵn sàng tiếp nhận dòng lệnh của người dùng, một dấu nhắc shell (còn gọi là dấu nhắc nhập lệnh) xuất hiện trên màn hình.

Linux có hai loại shell phổ biến là: C-shell (dấu nhắc %), Bourne-shell (dấu nhắc $) và một số shell phát triển từ các shell nói trên (chẳng hạn, TCshell - tcsh với dấu nhắc ngầm định > phát triển từ C-shell và GNU Bourne - bash với dấu nhắc bash # phát triển từ Bourne-shell). Dấu mời phân biệt shell nói trên không phải hoàn toàn rõ ràng do Linux cho phép người dùng thay đổi lại dấu nhắc shell nhờ việc thay giá trị các biến môi trường **PS1** và **PS2**. Trong tài liệu này, chúng ta sử dụng kí hiệu "hàng rào #" để biểu thị dấu nhắc shell.

C-shell có tên gọi như vậy là do cách viết lệnh và chương trình lệnh Linux tựa như ngôn ngữ C. Bourne-shell mang tên tác giả của nó là Steven Bourne. Một số lệnh trong C-shell (chẳng hạn lệnh **alias**) không còn có trong Bourne-shell và vì vậy để nhận biết hệ thống đang làm việc với shell nào, chúng ta gõ lệnh:

```
# alias
```

Nếu một danh sách xuất hiện thì shell đang sử dụng là C-shell; ngược lại, nếu xuất hiện thông báo "Command not found" thì shell đó là Bourne-shell.

Lệnh được chia thành 3 loại lệnh:

- Lệnh thường trực (có sẵn của Linux). Tuyệt đại đa số lệnh được giới thiệu trong tài liệu này là lệnh thường trực. Chúng bao gồm các lệnh được chứa sẵn trong shell và các lệnh thường trực khác.
- File chương trình ngôn ngữ máy: chẳng hạn, người dùng viết trình trên ngôn ngữ C qua bộ dịch ***gcc*** (bao gồm cả trình kết nối ***link***) để tạo ra một chương trình trên ngôn ngữ máy.

- File chương trình shell (Shell Scrip).

Khi kết thúc một dòng lệnh cần gõ phím ENTER để shell phân tích và thực hiện lệnh.

1.3. Giới thiệu về sử dụng lệnh trong Linux

Như đã giới thiệu ở phần trên, Linux là một hệ điều hành đa người dùng, đa nhiệm, được phát triển bởi hàng nghìn chuyên gia Tin học trên toàn thế giới nên hệ thống lệnh cũng ngày càng phong phú; đến thời điểm hiện nay Linux có khoảng hơn một nghìn lệnh.

Tuy nhiên chỉ có khoảng vài chục lệnh là thông dụng nhất đối với người dùng. Tài liệu này cũng hạn chế giới thiệu khoảng vài chục lệnh đó. Chúng ta đừng e ngại về số lượng lệnh được giới thiệu chỉ chiếm một phần nhỏ trong tập hợp lệnh bởi vì đây là những lệnh thông dụng nhất và chúng cung cấp một phạm vi ứng dụng rộng lớn, đủ thỏa mãn yêu cầu của chúng ta.

Cũng như đã nói ở trên, người dùng làm việc với máy tính thông qua việc sử dụng trạm cuối: người dùng đưa yêu cầu của mình bằng cách gõ "lệnh" từ bàn phím và giao cho hệ điều hành xử lý.

Khi cài đặt Linux lên máy tính cá nhân thì máy tính cá nhân vừa đóng vai trò trạm cuối, vừa đóng vai trò máy tính xử lý.

Dạng tổng quát của lệnh Linux có thể được viết như sau:

```
# <Tên lệnh> [<các tham số>] ¿
```

trong đó:

- Tên lệnh là một dãy ký tự, không có dấu cách, biểu thị cho một lệnh của Linux hay một chương trình. Người dùng cần hệ điều hành đáp ứng yêu cầu gì của mình thì phải chọn đúng tên lệnh. Tên lệnh là bắt buộc phải có khi gõ lệnh.
- Các tham số có thể có hoặc không có, được viết theo quy định của lệnh mà chúng ta sử dụng, nhằm cung cấp thông tin về các đối tượng mà lệnh tác động tới. Ý nghĩa của các dấu [, <, >,] được giải thích ở phần quy tắc viết lệnh.

Các tham số được phân ra thành hai loại: tham số khóa (sau đây thường dùng là "tùy chọn") và tham số vị trí. Tham số vị trí thường là tên file, thư mục và thường là các đối tượng chịu sự tác động của lệnh. Khi gõ lệnh, tham số vị trí được thay bằng những đối tượng mà người dùng cần hướng tác động tới. Tham số khóa chính là những tham số điều khiển hoạt động của lệnh theo các trường hợp riêng. Trong Linux, tham số khóa thường bắt đầu bởi dấu trừ "**-**" hoặc hai dấu trừ liên tiếp "**--**". Khi gõ lệnh, cũng giống như tên lệnh, tham số khóa phải được viết chính xác như trình bày trong mô tả lệnh. Một lệnh có thể có một số hoặc rất nhiều tham số khóa. Phụ thuộc vào yêu cầu cụ thể của mình, người dùng có thể chọn một hoặc một số các tham số khóa khi gõ lệnh. Trong mô tả lệnh, thường xuất hiện thuật ngữ ***tùy-chọn***. Tùy chọn lệnh (thực chất là ***tham số khóa***) cho phép điều chỉnh hoạt động của lệnh trong Linux, làm cho lệnh có tính phổ dụng cao. Tuỳ chọn lệnh cho phép lệnh có thể đáp ứng ý muốn của người dùng đối với hầu hết (tuy không phải lúc nào cũng vậy) các tình huống đặt ra cho thao tác ứng với lệnh.

- Ký hiệu "↵" biểu thị việc gõ phím hết dòng <Enter>. Để kết thúc một yêu cầu, người dùng nhất thiết phải gõ phím "↵".

Ví dụ, khi người dùng gõ lệnh xem thông tin về các file:

```
# ls -l g↵¿
```

trong lệnh này:

- **ls** là tên lệnh thực hiện việc đưa danh sách các tên file/ thư mục con trong một thư mục,
- **-l** là tham số khóa, cho biết yêu cầu xem đầy đủ thông tin về các đối tượng hiện ra. Chú ý, trong tham số khóa chữ cái (chữ "l") phải đi ngay sau dấu trừ "-". Tương ứng với lệnh **ls** còn có các tham số khóa **-a**, **-L**, ... và chúng cũng là các tùy chọn lệnh. Trong một số tham số khóa có nhiều chữ cái thay cho một dấu "**-**" là hai dấu "**--**" ở đầu tham số. Ví dụ, như trường hợp tham số **--file** của lệnh **date**.
- g* là tham số vị trí chỉ rõ người dùng cần xem thông tin về các file có tên gọi bắt đầu là chữ cái "g".

Trong tài liệu này, quy ước rằng khi viết một lệnh (trong mô tả lệnh và gõ lệnh) thì không cần phải viết dấu "↵" ở cuối dòng lệnh đó, song luôn ghi nhớ rằng phím ENTER ("↵") là bắt buộc khi gõ lệnh.

☞ ***Lưu ý:***

- Linux (và UNIX nói chung) được xây dựng trên ngôn ngữ lập trình C, vì vậy khi gõ lệnh phải phân biệt chữ thường với chữ hoa. Ngoại trừ một số ngoại lệ, trong Linux chúng ta thấy phổ biến là:
  - ❖ Các tên lệnh là chữ thường,
  - ❖ Một số tham số có thể là chữ thường hoặc chữ hoa (ví dụ, trong lệnh **date** về thời gian hệ thống thì hai tham số **-r** và **-R** có ý nghĩa hoàn toàn khác nhau). Tên các biến môi trường cũng thường dùng chữ hoa.
- Trong tài liệu này, tại những dòng văn bản diễn giải, sử dụng cách viết tên lệnh, các tham số khóa bằng kiểu chữ không chân, đậm như **date**, **-R**, **-r** ...
- Linux phân biệt siêu người dùng (tiếng Anh là superuser hoặc root, còn được gọi là ***người quản trị*** hay ***người dùng tối cao*** hoặc ***siêu người dùng***) với người dùng thông thường. Trong tập hợp lệnh của Linux, có một số lệnh mà chỉ siêu người dùng mới được phép sử dụng còn người dùng thông thường thì không được phép (ví dụ như lệnh **adduser** thực hiện việc bổ sung thêm người dùng). Mặt khác trong một số lệnh, với một số tham số khóa thì chỉ siêu người dùng được phép dùng, còn với một số tham số khác thì mọi người dùng đều được phép (ví dụ như lệnh **passwd** thay đổi mật khẩu người dùng).
- Một dòng lệnh có thể có nhiều hơn một lệnh, trong đó lệnh sau được ngăn cách bởi với lệnh đi ngay trước bằng dấu ";" hoặc dấu "|". Ví dụ về một số dòng lệnh dạng này:

```
# ls -l; date
# head Filetext | sort >temp
```

- Sau khi người dùng gõ xong dòng lệnh, shell tiếp nhận dòng lệnh này và phân tích nội dung văn bản của lệnh. Nếu lệnh được gõ đúng thì nó được thực hiện; ngược lại, trong trường hợp có sai sót khi gõ lệnh thì shell thông báo về sai sót và dấu nhắc shell lại hiện ra để chờ lệnh tiếp theo của người dùng. Về phổ biến, nếu như sau khi người dùng gõ lệnh, không thấy thông báo sai sót hiện ra thì có nghĩa lệnh đã được thực hiện một cách bình thường.

Trước khi đi vào nội dung chi tiết các lệnh thông dụng, chúng ta xem xét về một số quy định dùng trong mô tả lệnh được trình bày trong tài liệu này.

1.3.1. Các quy ước khi viết lệnh

Trong tài liệu này, các lệnh được trình bày theo một bộ quy tắc cú pháp nhất quán. Bộ quy tắc này cho phép phân biệt trong mỗi lệnh các thành phần nào là bắt buộc phải có, các thành phần nào có thể có hoặc không ... Dưới đây là nội dung của các quy tắc trong bộ quy tắc đó.

- Tên lệnh là bắt buộc, phải là từ đầu tiên trong bất kỳ lệnh nào, phải được gõ đúng như khi mô tả lệnh.
- Tên khái niệm được nằm trong cặp dấu ngoặc quan hệ (< và >) biểu thị cho một lớp đối tượng và là tham số bắt buộc phải có. Khi gõ lệnh thì tên khái niệm (có thể được coi là "tham số hình thức") phải được thay thế bằng một từ (thường là tên file, tên thư mục ... và có thể được coi là "tham số thực sự") để chỉ đối tượng liên quan đến thao tác của lệnh.

Ví dụ, mô tả cú pháp của lệnh **more** xem nội dung file là

```
# more <file>
```

thì từ **more** là tên lệnh, còn <file> là tham số trong đó ***file*** là tên khái niệm và là tham số bắt buộc phải có. Lệnh này có tác động là hiện lên màn hình theo cách thức cuộn nội dung của file với tên đã chỉ trong lệnh.
Để xem nội dung file có tên là ***temp***, người dùng gõ lệnh:

```
# more temp
```

Như vậy, tên lệnh **more** được gõ đúng như mô tả cú pháp (cả nội dung và vị trí) còn "***file***" đã được thay thế bằng từ "***temp***" là tên file mà người dùng muốn xem nội dung.

- Các bộ phận nằm giữa cặp dấu ngoặc vuông [và] là có thể gõ hoặc không gõ cũng được.

Ví dụ, cú pháp của lệnh **halt** là

```
# halt [tùy-chọn]
```

Với các tùy chọn là **-w**, **-n,** **-d**, **-f**, **-i** mã mỗi tùy chọn cho một cách thức hoạt động khác nhau của lệnh **halt.** Lệnh **halt** có tác động chính là làm ngừng hoạt động của hệ điều hành, tuy nhiên khi người dùng muốn có một cách hoạt động nào đó của lệnh này thì sẽ chọn một (hoặc một số) tuỳ chọn lệnh tương ứng. Một số cách gõ lệnh **halt** của người dùng như sau đây là đúng cú pháp:

```
# halt
# halt -w
# halt -n
# halt -f
```

- Các giá trị có trong cặp | và | trong đó các bộ phận cách nhau bằng dấu sổ đứng "|" cho biết cần chọn một và chỉ một trong các giá trị nằm giữa hai dấu ngoặc đó.

Ví dụ, khi giới thiệu về tùy chọn lệnh của lệnh **tail** xem phần cuối nội dung của file, chúng ta thấy:

```
-f, --follow[={tên | đặc tả}]
```

Như vậy, sau tham số khóa **--follow**, nếu xuất hiện thêm dấu bằng "=" thì phải có hoặc ***tên*** hoặc ***đặc tả***. Đây là trường hợp các chọn lựa "loại trừ nhau".

- Dấu ba chấm ... thể hiện việc lặp lại thành phần cú pháp đi ngay trước dấu này, việc lặp lại đó có thể từ không đến nhiều lần (không kể chính thành phần cú pháp đó). Cách thức này thường được dùng với các tham số như tên file.

Ví dụ, mô tả lệnh **chown** như sau:

```
chown [tùy-chọn] <chủ>[,[nhóm]]<file>...
```

Như vậy trong lệnh **chown** có thể không có hoặc có một số tùy chọn lệnh và có từ một đến nhiều tên file.

- Các bộ phận trong mô tả lệnh, nếu không nằm trong các cặp dấu [ ], <>, { } thì khi gõ lệnh thực sự phải gõ y đúng như khi mô tả (chú ý, quy tắc viết tên lệnh là một trường hợp riêng của quy tắc này).
- Việc kết hợp các dấu ngoặc với nhau cho phép tạo ra cách thức sử dụng quy tắc tổ hợp các tham số trong lệnh. Ví dụ, lệnh **more** bình thường có cú pháp là:

```
# more <file>
```

có nghĩa là thay <file> bằng tên file cần xem nội dung, nếu kết hợp thêm dấu ngoặc vuông [và], tức là có dạng sau (chính là dạng tổng quát của lệnh **more**):

```
# more [<file>]
```

thì <file> nói chung phải có trong lệnh **more**, tuy nhiên trong một số trường hợp có thể bỏ qua tham số file.

☞ ***Lưu ý:***

- Đối với nhiều lệnh, cho phép người dùng gõ ***tham số khóa kết hợp*** tương ứng với *tùy_chọn* trong mô tả lệnh. Tham số khóa kết hợp được viết theo cách **-<xâu-kí-tự>**, trong đó xâu-kí-tự gồm các chữ cái trong tham số khóa. Ví dụ, trong mô tả lệnh in lịch **cal**:

```
cal [tùy-chọn] [tháng [năm] ]
```

có ba tham số khóa là **-m, -j, -y**. Khi gõ lệnh có thể gõ một tổ hợp nào đó từ ba tham số khóa này để được tình huống sử dụng lệnh theo ý muốn. Chẳng hạn, nếu gõ lệnh

```
cal -mj 3
```

thì lệnh **cal** thực hiện theo điều khiển của hai tham số khóa **-m** (chọn Thứ Hai là ngày đầu tuần, thay vì cho ngầm định là Chủ Nhật) và **-j** (hiển thị ngày trong tháng dưới dạng số ngày trong năm kể từ đầu năm). Vì vậy, khi viết [tùy-chọn] trong mô tả lệnh biểu thị cả việc sử dụng từng tùy chọn, nhiều tùy chọn hoặc kết hợp các tuỳ chọn.

- Trong một số lệnh, có hai tham số khóa cùng tương ứng với một tình huống thực hiện lệnh, trong đó một tham số gồm một kí tự còn tham số kia lại là một từ. Tham số dài một từ là tham số chuẩn của lệnh, còn tham số một kí tự là cách viết ngắn gọn. Tham số chuẩn dùng được trong mọi Linux và khi gõ phải có đủ kí tự trong từ.

  Ví dụ, khi mô tả lệnh date có tùy chọn:

- `-d, --date=STRING`

  như vậy hai tham số **-d** và **--date=STRING** có cùng ý nghĩa.

Ngoài những quy ước trên đây, người dùng đừng quên một quy định cơ bản là cần phân biệt chữ hoa với chữ thường khi gõ lệnh.

1.3.3. Làm đơn giản thao tác gõ lệnh

Việc sử dụng bàn phím để nhập lệnh tuy không phải là một công việc nặng nề, song Linux còn cho phép người dùng sử dụng một số cách thức để thuận tiện hơn khi gõ lệnh. Một số trong những cách thức đó là:

- Sử dụng việc khôi phục dòng lệnh,
- Sử dụng các phím đặc biệt,
- Sử dụng các kí hiệu thay thế và phím <Tab>,
- Sử dụng thay thế **alias**,
- Sử dụng chương trình lệnh.

Cách thức sử dụng chương trình lệnh (shell script) sẽ được giới thiệu chi tiết trong các chương sau. Dưới đây, chúng ta xem xét cách thức sử dụng việc khôi phục dòng lệnh, phím đặc biệt và kí hiệu thay thế.

Cơ chế khôi phục dòng lệnh

Linux cung cấp một cách thức đặc biệt là khả năng khôi phục lệnh. Tại dấu nhắc shell: Người dùng sử dụng các phím mũi tên lên/xuống (↑/↓) trên bàn phím để nhận lại các dòng lệnh đã được đưa vào trước đây tại dấu nhắc shell, chọn một trong các dòng lệnh đó và biên tập lại nội dung dòng lệnh theo đúng yêu cầu mới của mình.

Ví dụ, người dùng vừa gõ xong dòng lệnh:

```
# ls -l tenfile*
```

sau đó muốn gõ lệnh **ls -l tentaptin** thì tại dấu nhắc của shell, người dùng sử dụng các phím di chuyển lên (↑) hoặc xuống (↓) để nhận được:

```
# ls -l tenfile*
```

dùng các phím tắt để di chuyển, xoá kí tự (xem phần sau) để có được:

```
# ls -l ten
```

và gõ tiếp các kí tự "taptin" để nhận được:

```
# ls -l tentaptin
```

chính là kết quả mong muốn.

Trong trường hợp số lượng kí tự thay thế là rất ít so với số lượng kí tự của toàn dòng lệnh thì hiệu quả của cách thức này rất cao.

☞ ***Lưu ý:***

- Việc nhấn liên tiếp các phím di chuyển lên (↑) hoặc xuống (↓) cho phép người dùng nhận được các dòng lệnh đã gõ từ trước mà không chỉ dòng lệnh mới được gõ. Cách thức này tương tự với cách thức sử dụng tiện ích DOSKEY trong hệ điều hành MS-DOS.

Một số phím đặc biệt khi gõ lệnh

Khi người dùng gõ lệnh có thể xảy ra một số tình huống như sau:

- Dòng lệnh đang gõ có chỗ sai sót, không đúng theo yêu cầu của người dùng vì vậy cần phải sửa lại đôi chút nội dung trên dòng lệnh đó. Trong trường hợp đó cần sử dụng các phím đặc biệt (còn gọi là phím viết tắt hay phím tắt) để di chuyển, xoá bỏ, bổ sung vào nội dung dòng lệnh.
- Sau khi sử dụng cách thức khôi phục dòng lệnh, chúng ta nhận được dòng lệnh tương tự với lệnh cần gõ và sau đó sử dụng các phím tắt để hoàn thiện lệnh.

Dưới đây giới thiệu các phím tắt và ý nghĩa của việc sử dụng chúng:

- Nhấn phím → để di chuyển con trỏ sang bên phải một vị trí
- Nhấn phím ← để di chuyển con trỏ sang bên trái một vị trí
- Nhấn phím <ESC-BACKSPACE> để xoá một từ bên trái con trỏ
- Nhấn phím <ESC-D> để xoá một từ bên phải con trỏ
- Nhấn phím <ESC-F> để di chuyển con trỏ sang bên phải một từ
- Nhấn phím <ESC-B> để di chuyển con trỏ sang bên trái một từ
- Nhấn phím <CTRL-A> để di chuyển con trỏ về đầu dòng lệnh
- Nhấn phím <CTRL-E> để di chuyển con trỏ về cuối dòng
- Nhấn phím <CTRL-U> để xóa dòng lệnh

Có thể dùng phím <ALT> thay cho phím <ESC>.

Các kí hiệu mô tả nhóm file và phím <Tab>

*Khi gõ lệnh thực sự nhiều trường hợp người dùng mong muốn một tham số trong lệnh không chỉ xác định một file mà lại liên quan đến một nhóm các file mà tên gọi của các file trong nhóm có chung một tính chất nào đó. Trong những trường hợp như vậy, người dùng cần sử dụng các kí hiệu mô tả nhóm file (wildcards), chúng ta gọi là kí hiệu mô tả nhóm (còn được gọi là kí hiệu thay thế). Người ta sử dụng các kí tự *, ? và cặp hai dấu [và] để mô tả nhóm file. Các kí tự này mang ý nghĩa như sau khi viết vào tham số tên file thực sự:*

- "*" : là ký tự mô tả nhóm gồm mọi xâu kí tự (thay thế mọi xâu). Mô tả này cho một nhóm lớn nhất trong ba mô tả.
- "?" : mô tả nhóm gồm mọi xâu với độ dài không quá 1 (thay thế một kí tự). Nhóm này là tập con của nhóm đầu tiên (theo kí tự "*").
- [xâu-kí-tự] : mô tả nhóm gồm mọi xâu có độ dài 1 là mỗi kí tự thuộc xâu nói trên. Mô tả này cho một nhóm có lực lượng bé nhất trong ba mô tả. Nhóm này là tập con của nhóm thứ hai (theo kí tự "?"). Khi gõ lệnh phải gõ cả hai dấu [và]. Một dạng khác của mô tả nhóm này là [<kí_tự_1>-<kí_tự_2>] nghĩa là giữa cặp dấu ngoặc có ba kí tự trong đó kí tự ở giữa là dấu nối (dấu -) thì cách viết này tương đương với việc liệt kê mọi kí tự từ <kí_tự_1> đến <kí_tự_2>. Chẳng hạn, cách viết [a-d] tương đương với cách viết [abcd].

Ví dụ, giả sử khi muốn làm việc với tất cả các file trong một thư mục nào đó, người dùng gõ * thay thế tham số ***file*** thì xác định được các tên file sau (chúng ta viết bốn tên file trên một dòng):

info-dir	**initlog.conf**	**inittab**	**lynx.cfg**
mail.rc	**mailcap**	**minicom.users**	**motd**

mtab **mtools.conf** **services** **shadow**
shadow- **shells** **smb.conf** **sysctl.conf**
syslog.conf **temp** **termcap** **up2date.conf**
temp **termcap**

Nếu người dùng gõ s* (để chỉ các tên có chữ cái đầu là s) thay thế tham số ***file*** thì xác định được các tên file sau:

shadow **shadow-** **shells** **sysctl.conf**
syslog.conf

Nếu người dùng gõ [si]* (để chỉ các tên có chữ cái đầu là s hoặc i, chú ý dùng cả hai kí tự [và]) thay thế tham số ***file*** thì xác định các tên file sau:

info-dir **initlog.conf** **inittab** **services**
shadow **shadow-** **shells** **smb.conf**
sysctl.conf **syslog.conf**

☞ ***Lưu ý:***

- Như vậy, Linux (và UNIX nói chung) không chỉ sử dụng hai kí tự mô tả nhóm * và ? mà còn có cách thức sử dụng cặp kí tự [và].
- Cần phân biệt cặp dấu [và] được sử dụng khi người dùng gõ lệnh có ý nghĩa hoàn toàn khác với ý nghĩa của chúng khi được sử dụng trong mô tả lệnh.

Hơn thế nữa, Linux còn cung cấp cho người dùng cách thức sử dụng phím <**TAB**> để hoàn thành nốt tên file (tên thư mục) trong lệnh. Ví dụ, khi chúng ta gõ dòng lệnh

```
# ls /u<TAB>local<TAB>b<TAB>
```

thì nó cũng tương đương như gõ dòng lệnh (và đây chính là nội dung xuất hiện tại dấu nhắc shell):

```
# ls /usr/local/bin
```

với điều kiện trong thư mục /**usr** chỉ có thư mục **local** được bắt đầu bởi chữ "l" và trong thư mục **local** cũng chỉ có thư mục **bin** được bắt đầu bởi chữ "b".
Trong trường hợp nếu như một kí tự chưa đủ xác định, người dùng cần gõ thêm kí tự tiếp theo trong tên file (tên thư mục) và nhấn phím <**TAB**> để hoàn thành dòng lệnh.

1.3.4. Tiếp nối dòng lệnh

Như đã lưu ý trên đây, một dòng lệnh có thể gồm một hoặc một số lệnh, mặt khác tham số của lệnh có thể là rất dài không thể trong khuôn khổ của một dòng văn bản được. Khi gõ lệnh, nếu dòng lệnh quá dài, Linux cho phép ngắt dòng lệnh xuống dòng dưới bằng cách thêm kí tự báo hiệu chuyển dòng "\" tại cuối dòng; trong trường hợp đó, kí tự "\" phải là ký tự cuối cùng thuộc dòng lệnh trước.

Ví dụ,

```
# cd vsd\
thumuc
```

thì dòng thứ hai là phần tiếp theo của dòng thứ nhất và kết hợp cả hai dòng này thực chất là một dòng lệnh Linux.

1.4. Trang Man

Chúng ta có thể nói rằng Linux là một hệ điều hành rất phức tạp với hàng nghìn lệnh và mỗi lệnh lại có thể có tới vài hoặc vài chục tình huống sử dụng do chúng cho phép có nhiều tùy chọn lệnh. Để thuộc hết được nội dung tất cả các lệnh của Linux là một điều hết sức khó khăn, có thể nói là không thể. Linux cho phép người dùng sử dụng cách thức gọi trang Man để có được các thông tin đầy đủ giới thiệu nội dung các lệnh. Dưới đây là một số nội dung về cách thức sử dụng trang Man.

"Man" là từ viết tắt của "manual", được coi là tài liệu trực tuyến trong Linux đã lưu trữ toàn bộ các lệnh có sẵn với các thông tin tham khảo khá đầy đủ cho phép người dùng có thể mở ra để nhận được trợ giúp.

Để mở trang Man của một lệnh, chúng ta sử dụng lệnh **man** của Linux và gõ:

```
# man <tên-lệnh>
```

Nội dung của trang Man tuy không phải là quá khó hiểu, song để hiểu hết được nó cũng đòi hỏi không ít thời gian. Tuy vậy, nếu quên nội dung một lệnh nào đó thì cách tốt nhất là hãy sử dụng trang Man.

Cấu trúc chung của một trang Man như sau:

```
COMMAND(1)          Linux Programmer's Manual          COMMAND(1)
NAME
    tên lệnh - khái quát tác dụng của lệnh
SYNOPSIS
    cú pháp của lệnh
DESCRIPTION
    mô tả cụ thể hơn về tác dụng của lệnh
OPTIONS
    liệt kê các tùy chọn lệnh và tác dụng của chúng
FILES
    liệt kê các file mà lệnh sử dụng hoặc tham chiếu đến
SEE ALSO
    liệt kê các lệnh, các tài liệu, ..., có liên quan đến lệnh
REPORTING BUGS
    địa chỉ liên hệ nếu gặp lỗi khi sử dụng lệnh
AUTHOR
    tên tác giả của lệnh
```

Người dùng thậm chí không nhớ chính xác tên lệnh. Linux còn có một cách thức hỗ trợ người dùng có thể nhanh chóng tìm được lệnh cần sử dụng trong trường hợp chỉ nhớ những chữ cái đầu của tên lệnh, đó là cách thức sử dụng phím TAB. Trong cách thức này, người dùng chỉ cần nhớ một số chữ cái đầu tiên của tên lệnh.

Có thể trình bày cách thức đó theo cú pháp sau đây:

```
# <dãy-chữ-cái><TAB><TAB>
```

Trong đó dãy-chữ-cái có từ một đến một vài chữ cái thuộc phần đầu của tên lệnh. Chú ý rằng, các chữ cái và hai phím <TAB> phải được gõ liên tiếp nhau.

Kết hợp cách thức này với cách thức sử dụng lệnh **man** (với sự phong phú về tùy chọn của lệnh **man**) nhận được một cách thức khá tuyệt vời trợ giúp người dùng.

Ví dụ, muốn sử dụng lệnh **history** nhưng lại không nhớ chính xác tên lệnh được viết ra như thế nào mà chỉ nhớ nó được bắt đầu bởi chữ **h**, hãy gõ chữ **h** đó tại dấu nhắc shell và nhấn phím TAB hai lần, sẽ thấy một danh sách các lệnh có chữ cái đầu tiên là **h** được hiện ra trên màn hình:

```
# h<TAB><TAB>
h2ph          hboot       help          hexdump     history
hostname      htdigest    h2xs  hcc     helpme      hf77
hltest        hoststat    htpasswd      halt        hcp
helptool      hinotes     host          hpcdtoppm   hash
head          hexbin      hipstopgm     hostid      hpftodit
```

Như vậy, tất cả các lệnh có tên bắt đầu với chữ **h** được hiển thị trên màn hình và cho phép người dùng có thể xác định được lệnh cần quan tâm.

Trường hợp tồn tại một số lượng lớn các lệnh có cùng chữ cái đầu tiên mà người dùng đã gõ, thay vì hiện hết mọi tên lệnh, hệ điều hành cho ra một thông báo hỏi người dùng có muốn xem toàn bộ các lệnh đó hay không. Người dùng đáp ứng thông báo đó tuỳ theo ý muốn của mình.

Ví dụ, khi người dùng gõ nội dung như sau:

```
# p<TAB><TAB>
```

thì hệ thống đáp lại là:

```
There are 289 possibilities. Do you really wish to see them all? (y or n)
```

Người dùng gõ phím "y" nếu muốn xem, hoặc gõ "n" nếu bỏ qua.

Người dùng có thể gõ nhiều hơn một chữ cái ở đầu tên lệnh và điều đó cho phép giảm bớt số tên lệnh mà hệ thống tìm được và hiển thị. Chẳng hạn, khi biết hai chữ cái đầu là "pw" và người dùng gõ:

```
# pw<TAB><TAB>
```

thì hệ thống sẽ hiện ra danh sách các tên lệnh bắt đầu bởi "pw":

```
pwck          pwconv      pwd           pwdb_chkpwd     pwunconv
```

Trong trường hợp này, người dùng sẽ nhận biết được tên lệnh đang cần tìm thuận tiện hơn.

CHƯƠNG 2. THAO TÁC VỚI HỆ THỐNG

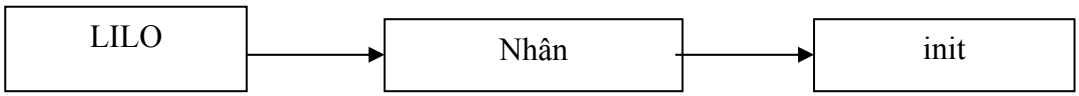
2.1. Quá trình khởi động Linux

Trong phần này, chúng ta xem xét sơ bộ quá trình khởi động hệ điều hành Linux.

Một trong những cách thức khởi động Linux phổ biến nhất là cách thức do chương trình LILO (LInux LOader) thực hiện. Chương trình LILO được nạp lên đĩa của máy tính khi cài đặt hệ điều hành Linux. LILO được nạp vào Master Boot Record của đĩa cứng hoặc vào Boot Sector tại phân vùng khởi động (trên đĩa cứng hoặc đĩa mềm). Giả sử máy tính của chúng ta đã cài đặt Linux và sử dụng LILO để khởi động hệ điều hành. LILO thích hợp với việc trên máy tính được cài đặt một số hệ điều hành khác nhau và theo đó, LILO còn cho phép người dùng chọn lựa hệ điều hành để khởi động.

Giai đoạn khởi động Linux tùy thuộc vào cấu hình LILO đã được lựa chọn trong quá trình cài đặt Linux. Trong tình huống đơn giản nhất, Linux được khởi động từ đĩa cứng hay đĩa mềm khởi động.

Quá trình khởi động Linux có thể được mô tả theo sơ đồ sau:



Theo sơ đồ này, LILO được tải vào máy để thực hiện mà việc đầu tiên là đưa nhân vào bộ nhớ trong và sau đó tải chương trình init để thực hiện việc khởi động Linux.

Nếu cài đặt nhiều phiên bản Linux hay cài Linux cùng các hệ điều hành khác (trong các trường hợp như thế, mỗi phiên bản Linux hoặc hệ điều hành khác được gán ***nhãn*** - label để phân biệt), thì thông báo sau đây được LILO đưa ra:

LILO boot:

cho phép nhập xâu là nhãn của một trong những hệ điều hành hiện có trên máy để khởi động nó. Tại thời điểm đó, người dùng cần gõ nhãn của hệ điều hành cần khởi động vào, ví dụ, gõ

LILO boot: linux

nếu chọn khởi động để làm việc trong Linux, hoặc gõ

LILO boot: dos

nếu chọn khởi động để làm việc trong MS-DOS, Windows.

☞ ***Lưu ý:***

- Nếu chúng ta không nhớ được nhãn của hệ điều hành có trong máy để chọn, hãy gõ phím <TAB> để được LILO cho biết nhãn của các hệ điều hành.

  LILO boot: <TAB>

  sẽ hiện ra danh sách các nhãn (ví dụ như): **`linux dos`** ...
  và hiện lại thông báo nói trên để ta gõ nhãn của hệ điều hành.
- LILO cũng cho phép đặt chế độ chọn ngầm định hệ điều hành để khởi động mà theo đó nếu chúng ta không có tác động gì sau thông báo chọn hệ điều hành thì LILO sẽ tự động chọn hệ điều hành ngầm định ra để khởi động. Nếu chúng ta không can thiệp vào các file tương ứng của trình LILO thì hệ điều hành Linux là hệ điều hành ngầm định.

Giả sử Linux đã được chọn để khởi động. Khi ***init*** thực hiện, chúng ta sẽ thấy một chuỗi (khoảng vài chục) dòng thông báo cho biết hệ thống phần cứng được Linux nhận diện và thiết lập cấu hình cùng với tất cả trình điều khiển phần mềm được nạp khi khởi động. Quá trình ***init*** là quá trình khởi thủy, là cha của mọi quá trình. Tại thời điểm khởi động hệ thống ***init*** thực hiện vai trò đầu tiên của mình là chạy chương trình shell trong file ***/etc/inittab*** và các dòng thông báo trên đây chính là kết quả của việc chạy chương trình shell đó. Sau khi chương trình shell trên được thực hiện xong, bắt đầu quá trình người dùng đăng nhập (login) vào hệ thống.

2.2. Thủ tục đăng nhập và các lệnh thoát khỏi hệ thống

2.2.1. Đăng nhập

Sau khi hệ thống Linux (lấy Red Hat 6.2 làm ví dụ) khởi động xong, trên màn hình xuất hiện những dòng sau:

Ret Hat Linux release 6.2 (Zoot)
Kernel 2.2.14-5.0 on an i686
May1 login:

Dòng thứ nhất và dòng thứ hai cho biết loại phiên bản Linux, phiên bản của nhân và

---

*Chúng ta có thể thay đổi các dòng hiển thị như trình bày trên đây bằng cách sửa đổi file **/etc/rc.d/rc.local** như sau:*

Thay đoạn chương trình
echo "" > /etc/issue
echo "$R" >> /etc/issue
echo "Kernel $(uname -r) on $a SMP(uname -m)" >> /etc/issue
cp -f /etc/issue /etc/issue.net
echo >> /etc/issue
thành
echo "" > /etc/issue
echo "Thông báo muốn hiển thị" >> /etc/issue
ví dụ sửa thành:
echo "" > /etc/issue
echo "This is my computer" >> /etc/issue
thì trên màn hình đăng nhập sẽ có dạng sau:

This is my computer
hostname login:

---

kiến trúc phần cứng có trên máy, dòng thứ ba là dấu nhắc đăng nhập để người dùng thực hiện việc đăng nhập. Chú ý là các dòng trên đây có thể thay đổi chút ít tùy thuộc vào phiên bản Linux.

Tại dấu nhắc đăng nhập, hãy nhập tên người dùng (còn gọi là tên đăng nhập): đây là tên kí hiệu đã cung cấp cho Linux nhằm nhận diện một người dùng cụ thể. Tên đăng nhập ứng với mỗi người dùng trên hệ thống là duy nhất, kèm theo một mật khẩu đăng nhập.

May1 login: root
Password:

Khi nhập xong tên đăng nhập, hệ thống sẽ hiện ra thông báo hỏi mật khẩu và di chuyển con trỏ xuống dòng tiếp theo để người dùng nhập mật khẩu. Mật khẩu khi được nhập sẽ không hiển thị trên màn hình và chính điều đó giúp tránh khỏi sự "nhòm ngó" của người khác.

Nếu nhập sai tên đăng nhập hoặc mật khẩu, hệ thống sẽ đưa ra một thông báo lỗi:

May1 login: root
Password:
Login incorrect

Máy1 login:

Nếu đăng nhập thành công, người dùng sẽ nhìn thấy một số thông tin về hệ thống, một vài tin tức cho người dùng... Lúc đó, dấu nhắc shell xuất hiện để người dùng bắt đầu phiên làm việc của mình.

May1 login: root
Password:
Last login: Fri Oct 27 14:16:09 on tty2
Root[may1 /root]#

Dãy kí tự trong dòng cuối cùng chính là ***dấu nhắc shell***. Trong dấu nhắc này, ***root*** là tên người dùng đăng nhập, ***may1*** là tên máy và **/root** tên thư mục hiện thời (vì đây là người dùng root). Khi dấu nhắc shell xuất hiện trên màn hình thì điều đó có nghĩa là hệ điều hành đã sẵn sàng tiếp nhận một yêu cầu mới của người dùng.

Dấu nhắc shell có thể khác với trình bày trên đây (Mục 2.7 cung cấp cách thay đổi dấu nhắc shell), nhưng có thể hiểu nó là chuỗi kí tự bắt đầu một dòng có chứa trỏ chuột và luôn xuất hiện mỗi khi hệ điều hành hoàn thành một công việc nào đó.

2.2.2. Ra khỏi hệ thống

Để kết thúc phiên làm việc người dùng cần thực hiện thủ tục ra khỏi hệ thống. Có rất nhiều cách cho phép thoát khỏi hệ thống, ở đây chúng ta xem xét một số cách thông dụng nhất.

- Cách đơn giản nhất để đảm bảo thoát khỏi hệ thống đúng đắn là nhấn tổ hợp phím **CTRL**+**ALT**+**DEL**. Khi đó, trên màn hình sẽ hiển thị một số thông báo của hệ thống và cuối cùng là thông báo thoát trước khi tắt máy. Cần chú ý là: Nếu đang làm việc trong môi trường X Window System, hãy nhấn tổ hợp phím **CTRL**+**ALT**+**BACKSPACE** trước rồi sau đó hãy nhấn **CTRL**+**ALT**+**DEL**.
- Cách thứ hai là sử dụng lệnh **shutdown** với cú pháp như sau:

```
shutdown [tùy-chọn] <time> [cảnh-báo]
```

Lệnh này cho phép dừng tất cả các dịch vụ đang chạy trên hệ thống.
Các tùy-chọn của lệnh này như sau:

- **-k** : không thực sự shutdown mà chỉ cảnh báo.
- **-r** : khởi động lại ngay sau khi shutdown.
- **-h** : tắt máy thực sự sau khi shutdown.
- **-f** : khởi động lại nhanh và bỏ qua việc kiểm tra đĩa.
- **-F** : khởi động lại và thực hiện việc kiểm tra đĩa.

- **-c :** bỏ qua không chạy lệnh shutdown. Trong tùy chọn này không thể đưa ra tham số thời gian nhưng có thể đưa ra thông báo giải thích trên dòng lệnh gửi cho tất cả các người dùng.
- **-t *số-giây* :** qui định init(8) chờ khoảng thời gian số-giây tạm dừng giữa quá trình gửi cảnh báo và tín hiệu kill, trước khi chuyển sang một mức chạy khác.

và hai tham số vị trí còn lại:

- **time :** đặt thời điểm shutdown. Tham số time có hai dạng. Dạng tuyệt đối là gg:pp (gg: giờ trong ngày, pp: phút) thì hệ thống sẽ shutdown khi đồng hồ máy trùng với giá trị tham số. Dạng tương đối là +<số> là hẹn sau thời khoảng <số> phút sẽ shutdown; coi shutdown lập tức tương đương với +0.
- **cảnh-báo :** thông báo gửi đến tất cả người dùng trên hệ thống. Khi lệnh thực hiện tất cả các máy người dùng đều nhận được cảnh báo.

Ví dụ, khi người dùng gõ lệnh:

```
shutdown +1 Sau mot phut nua he thong se shutdown!
```

trên màn hình của tất cả người dùng xuất hiện thông báo "Sau mot phut nua he thong se shutdown! " và sau một phút thì hệ thống shutdown thực sự.

- Cách thứ ba là sử dụng lệnh **halt** với cú pháp như sau:

```
halt [tùy-chọn]
```

Lệnh này tắt hẳn máy.
Các tuỳ chọn của lệnh **halt**:

- **-w :** không thực sự tắt máy nhưng vẫn ghi các thông tin lên file /var/log/wtmp (đây là file lưu trữ danh sách các người dùng đăng nhập thành công vào hệ thống).
- **-d :** không ghi thông tin lên file /var/log/wtmp. Tùy chọn -n có ý nghĩa tương tự song không tiến hành việc đồng bộ hóa.
- **-f :** thực hiện tắt máy ngay mà không thực hiện lần lượt việc dừng các dịch vụ có trên hệ thống.
- **-i :** chỉ thực hiện dừng tất cả các dịch vụ mạng trước khi tắt máy.

Chúng ta cần nhớ rằng, nếu thoát khỏi hệ thống không đúng cách thì dẫn đến hậu quả là một số file hay toàn bộ hệ thống file có thể bị hư hỏng.

☞ ***Lưu ý:***

- Có thể sử dụng lệnh **exit** để trở về dấu nhắc đăng nhập hoặc kết thúc phiên làm việc bằng lệnh **logout**.

2.2.3. Khởi động lại hệ thống

Ngoài việc thoát khỏi hệ thống nhờ các cách thức trên đây (ấn tổ hợp ba phím Ctrl+Alt+Del, dùng lệnh **shutdown** hoặc lệnh **halt**), khi cần thiết (chẳng hạn, gặp phải tình huống một trình ứng dụng chạy quẩn) có thể khởi động lại hệ thống nhờ lệnh **reboot**.
Cú pháp lệnh **reboot**:

```
reboot [tùy-chọn]
```

Lệnh này cho phép khởi động lại hệ thống. Nói chung thì chỉ siêu người dùng mới được phép sử dụng lệnh **reboot**, tuy nhiên, nếu hệ thống chỉ có duy nhất một người dùng đang làm việc thì lệnh **reboot** vẫn được thực hiện song hệ thống đòi hỏi việc xác nhận mật khẩu.

Các tùy chọn của lệnh **reboot** như sau là **-w**, **-d**, **-n**, **-f**, **-i** có ý nghĩa tương tự như trong lệnh **halt**.

2.2.4. Khởi động vào chế độ đồ hoạ

Linux cho phép nhiều chế độ khởi động, những chế độ này được liệt kê trong file /etc/inittab. Dưới đây là nội dung của file này:

```
# inittab This file describes how the INIT process should set up
# the system in a certain run-level.
#
# Author: Miquel van Smoorenburg, <miquelsừdrinkel.nl.mugnet.org>
# Modified for RHS Linux by Marc Ewing and Donnie Barnes
#

# Default runlevel. The runlevels used by RHS are:
# 0 - halt (Do NOT set initdefault to this) - Đây là chế độ dừng hoạt động của hệ thống
# 1 - Single user mode - Đây là chế độ đơn người dùng, ta có thể đăng nhập vào chế độ
  này trong trường hợp muốn khắc phục một số sự cố.
# 2 - Multiuser, without NFS (The same as 3, if you do not have networking) - Đây là chế
  độ đa người dùng, giao diện text, không hỗ trợ kết nối mạng.
# 3 - Full multiuser mode – Chế độ đa người dùng, giao diện text
# 4 – unused – Không sử dụng chế độ này
# 5 - X11 - Đây là chế độ đa người dùng, giao diện đồ hoạ
# 6 - reboot (Do NOT set initdefault to this) – Chế độ khởi động lại máy tính
#
id:3:initdefault: - Đây là chế độ ngầm định hệ thống sẽ sử dụng để khởi động

# System initialization.
si::sysinit:/etc/rc.d/rc.sysinit

l0:0:wait:/etc/rc.d/rc 0
l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6

# Things to run in every runlevel.
ud::once:/sbin/update
# Trap CTRL-ALT-DELETE
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
#ca::ctrlaltdel:/bin/echo "You can't do that"

# When our UPS tells us power has failed, assume we have a few minutes
# of power left. Schedule a shutdown for 2 minutes from now.
# This does, of course, assume you have powerd installed and your
```

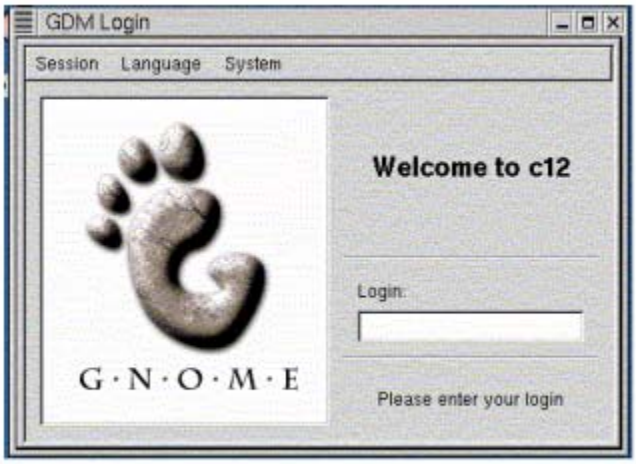
```
# UPS connected and working correctly.
pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System Shutting Down"

# If power was restored before the shutdown kicked in, cancel it.
pr:12345:powerokwait:/sbin/shutdown -c "Power Restored; Shutdown Cancelled"


# Run gettys in standard runlevels
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
#3:2345:respawn:/sbin/mingetty tty3
#4:2345:respawn:/sbin/mingetty tty4
#5:2345:respawn:/sbin/mingetty tty5
#6:2345:respawn:/sbin/mingetty tty6
# Run xdm in runlevel 5
# xdm is now a separate service
x:5:respawn:/etc/X11/prefdm –nodaemon
```

Trong đó chế độ khởi động số 3 là chế độ khởi động vào chế độ Text, và chế độ 5 là khở động vào chế độ đồ hoạ. Như vậy để cho máy tính khởi động vào chế độ đồ hoạ ta sửa lại dòng cấu hình

id:3:initdefault:

thành

id:5:initdefault:



Màn hình đăng nhập đồ họa

Trong Linux có một số loại giao diện đồ hoạ do một số tổ chức viết ra. Hai tổ chức nổi tiếng là GNOME (http://www.gnome.org) và KDE (http://www.kde.org) đã viết ra các giao diện đồ hoạ mang tên trùng với tổ chức đó là GNOME và KDE.

Cũng tuỳ vào việc được cài giao diện GNOME hay KDE mà khi khởi động vào chế độ đồ hoạ, máy tính có các giao diện tương ứng. Trên hình trên là giao diện GNOME mà khi khởi động vào chế độ đồ hoạ. Mặt khác, các giao diện này liên tục được phát triển, do đó ở mỗi phiên bản sẽ có sự khác nhau. Trong giao diện đăng nhập đồ họa, hệ thống hiển thị hộp thoại cho phép người dùng nhập vào tên tài khoản; sau khi người dùng nhập tên tài khoản

của mình, hệ thống hỏi tiếp mật khẩu. Nếu cả tên tài khoản và mật khẩu đều chính xác thì người dùng được phép vào hệ thống và một giao diện làm việc mới sẽ hiện ra như hình dưới.

Màn hình GNOME

2.3. Lệnh thay đổi mật khẩu

Mật khẩu là vấn đề rất quan trọng trong các hệ thống đa người dùng và để đảm bảo tính bảo mật tối đa, cần thiết phải chú ý tới việc thay đổi mật khẩu. Thậm chí trong trường hợp hệ thống chỉ có một người sử dụng thì việc thay đổi mật khẩu vẫn là rất cần thiết.

Mật khẩu là một xâu kí tự đi kèm với tên người dùng để đảm bảo cho phép một người vào làm việc trong hệ thống với quyền hạn đã được quy định. Trong quá trình đăng nhập, người dùng phải gõ đúng tên và mật khẩu, trong đó gõ mật khẩu là công việc bắt buộc phải thực hiện. Tên người dùng có thể được công khai song mật khẩu thì tuyệt đối phải được đảm bảo bí mật.

- Việc đăng ký tên và mật khẩu của siêu người dùng được tiến hành trong quá trình khởi tạo hệ điều hành Linux.
- Việc đăng ký tên và mật khẩu của một người dùng thông thường được tiến hành khi một người dùng mới đăng ký tham gia sử dụng hệ thống. Thông thường siêu người dùng cung cấp tên và mật khẩu cho người dùng mới (có thể do người dùng đề nghị) và dùng lệnh **adduser** (hoặc lệnh **useradd**) để đăng ký tên và mật khẩu đó với hệ thống. Sau đó, người dùng mới nhất thiết cần thay đổi mật khẩu để bảo đảm việc giữ bí mật cá nhân tuyệt đối.

Lệnh **passwd** cho phép thay đổi mật khẩu ứng với tên đăng nhập người dùng. Cú pháp lệnh **passwd**:

```
passwd [tùy-chọn] [tên-người-dùng]
```

với các tùy chọn như sau:

- **-k :** thay đổi mật khẩu người dùng. Lệnh đòi hỏi phải xác nhận quyền bằng việc gõ mật khẩu đang dùng trước khi thay đổi mật khẩu. Cho phép người dùng thay đổi mật khẩu của mình độc lập với siêu người dùng.
- **-f :** đặt mật khẩu mới cho người dùng song không cần tiến hành việc kiểm tra mật khẩu đang dùng. Chỉ siêu người dùng mới có quyền sử dụng tham số này.
- **-l :** khóa một tài khoản người dùng. Việc khóa tài khoản thực chất là việc dịch bản mã hóa mật khẩu thành một xâu ký tự vô nghĩa bắt đầu bởi kí hiệu "!". Chỉ siêu người dùng mới có quyền sử dụng tham số này.
- **-stdin :** việc nhập mật khẩu người dùng chỉ được tiến hành từ thiết bị vào chuẩn không thể tiến hành từ đường dẫn (pipe). Nếu không có tham số này cho phép nhập mật khẩu cả từ thiết bị vào chuẩn hoặc từ đường dẫn.
- **-u :** mở khóa (tháo bỏ khóa) một tài khoản (đối ngẫu với tham số -l). Chỉ siêu người dùng mới có quyền sử dụng tham số này.
- **-d :** xóa bỏ mật khẩu của người dùng. Chỉ siêu người dùng mới có quyền sử dụng tham số này.
- **-S :** hiển thị thông tin ngắn gọn về trạng thái mật khẩu của người dùng được đưa ra. Chỉ siêu người dùng mới có quyền sử dụng tham số này.

Nếu tên-người-dùng không có trong lệnh thì ngầm định là chính người dùng đã gõ lệnh này.

Ví dụ khi người dùng **user1** gõ lệnh:

```
# passwd user1
```

hệ thống thông báo:

Changing password for user user1

New UNIX password:

để người dùng nhập mật khẩu mới của mình vào. Sau khi người dùng gõ xong mật khẩu mới, hệ thống cho ra thông báo:

BAD PASSWORD: it is derived from your password entry

Retype new UNIX password:

để người dùng khẳng định một lần nữa mật khẩu vừa gõ dòng trên (nhớ phải gõ lại đúng hệt như lần trước). Chớ nên quá phân vân vì thông báo ở dòng phía trên vì hầu hết khi gõ mật khẩu mới luôn gặp những thông báo kiểu đại loại như vậy, chẳng hạn như:

BAD PASSWORD: it is too simplistic/systematic

Và sau khi chúng ta khẳng định lại mật khẩu mới, hệ thống cho ra thông báo:

Passwd: all authentication tokens updated successfully.

cho biết việc thay đổi mật khẩu thành công và dấu nhắc shell lại hiện ra.
Khi siêu người dùng gõ lệnh:

```
# passwd -S root
```

sẽ hiện ra thông báo

Changing password for user root

Password set, MD5 encryption

cho biết thuật toán mã hóa mật khẩu mà Linux sử dụng là một thuật toán hàm băm có tên là MD5.

☞ ***Lưu ý:***

- Có một lời khuyên đối với người dùng là nên chọn mật khẩu không quá đơn giản quá (nhằm tránh người khác dễ dò tìm ra) hoặc không quá phức tạp (tránh khó khăn cho chính người dùng khi phải ghi nhớ và gõ mật khẩu). Đặc biệt không nên sử dụng họ tên, ngày sinh, số điện thoại ... của bản thân hoặc người thân làm mật khẩu vì đây là một trong những trường hợp mật khẩu đơn giản nhất.
- Nếu thông báo mật khẩu quá đơn giản được lặp đi lặp lại một vài lần và không có thông báo mật khẩu mới thành công đã quay về dấu nhắc shell thì nên gõ lại lệnh và chọn một mật khẩu mới phức tạp hơn đôi chút.

2.4. Lệnh xem, thiết đặt ngày, giờ hiện tại và xem lịch trên hệ thống

2.4.1 Lệnh xem, thiết đặt ngày, giờ

Lệnh **date** cho phép có thể xem hoặc thiết đặt lại ngày giờ trên hệ thống.

Cú pháp của lệnh gồm hai dạng, dạng xem thông tin về ngày, giờ:

```
date [tùy-chọn] [+định-dạng]
```

và dạng thiết đặt lại ngày giờ cho hệ thống:

```
date [tùy-chọn] [MMDDhhmm[ [CC[YY] ]-ss]]
```

Các tùy-chọn như sau:

- **-d, --date=xâu-văn-bản** : hiển thị thời gian dưới dạng ***xâu-văn-bản***, mà không lấy "thời gian hiện tại của hệ thống" như theo ngầm định; ***xâu-văn-bản*** được đặt trong hai dấu nháy đơn hoặc hai dấu nháy kép.
- **-f, --file=file-văn-bản** : giống như một tham số **--date** nhưng ứng với nhiều ngày cần xem: mỗi dòng của file-văn-bản có vai trò như một **xâu-văn-bản** trong trường hợp tham số **--date**.
- **-I, --iso-8601[=mô-tả]** : hiển thị ngày giờ theo chuẩn ISO-8601 (ví dụ: 2000-11-8).
  - -I tương đương với tham số **--iso-8601='date'**
  - Với **--iso-8601**: nếu ***mô-tả*** là '**date**' (hoặc không có) thì hiển thị ngày, nếu ***mô-tả*** là '**hours**' hiển thị ngày+giờ, nếu ***mô-tả*** là **'minutes'**: ngày+giờ+phút; nếu ***mô-tả*** là **'seconds'**: ngày + giờ + phút + giây.
- **-r, --reference= file** : hiển thị thời gian sửa đổi ***file*** lần gần đây nhất.
- **-R, --rfc-822** : hiển thị ngày theo RFC-822 (ví dụ: Wed, 8 Nov 2000 09:21:46 -0500).
- **-s, --set=xâu-văn-bản** : thiết đặt lại thời gian theo kiểu **xâu-văn-bản**.
- **-u, --utc, --universal** : hiển thị hoặc thiết đặt thời gian theo UTC (ví dụ: Wed Nov 8 14:29:12 UTC 2000).
- **--help** : hiển thị thông tin trợ giúp và thoát.

Trong dạng lệnh **date** cho xem thông tin ngày, giờ thì tham số định-dạng điều khiển cách hiển thị thông tin kết quả. Định-dạng là dãy có từ một đến nhiều cặp gồm hai kí tự, trong mỗi cặp kí tự đầu tiên là % còn kí tự thứ hai mô tả định dạng.

Do số lượng định dạng là rất nhiều vì vậy chúng ta chỉ xem xét một số định dạng điển hình (để xem đầy đủ các định dạng, sử dụng lệnh **man date**).

Dưới đây là một số định dạng điển hình:

- `%%` : Hiện ra chính kí tự %.
- `%a` : Hiện ra thông tin tên ngày trong tuần viết tắt theo ngôn ngữ bản địa.
- `%A` : Hiện ra thông tin tên ngày trong tuần viết đầy đủ theo ngôn ngữ bản địa.
- `%b` : Hiện ra thông tin tên tháng viết tắt theo ngôn ngữ bản địa.
- `%B` : Hiện ra thông tin tên tháng viết đầy đủ theo ngôn ngữ bản địa.

Trong dạng lệnh **date** cho phép thiết đặt lại ngày giờ cho hệ thống thì tham số [MMDDhhmm[[CC[YY] [.ss]] mô tả ngày, giờ mới cần thiết đặt, trong đó:

MM: hai số chỉ tháng,
DD: hai số chỉ ngày trong tháng,
hh: hai số chỉ giờ trong ngày,
mm: hai số chỉ phút,
CC: hai số chỉ thế kỉ,
YY: hai số chỉ năm trong thế kỉ.

Các dòng ngay dưới đây trình bày một số ví dụ sử dụng lệnh **date**, mỗi ví dụ được cho tương ứng với một cặp hai dòng, trong đó dòng trên mô tả lệnh được gõ còn dòng dưới là thông báo của Linux.

```
# date
Wed Jan 3 23:58:50 ICT 2001
# date -d='01/01/2000'
Sat Jan 1 00:00:00 ICT 2000
# date -iso-8601='seconds'
2000-12-01T00:36:41-0500
# date -d='01/01/2001'
Mon Jan 1 00:00:00 ICT 2001
# date 010323502001.50
Wed Jan 3 23:50:50 ICT 2001
# date +%a%A
Wed Wednesday
# date +%a%A%b%B
Wed Wednesday Jan January
# date +%D%%%j
01/05/01%005
```

2.4.2. Lệnh xem lịch

Lệnh **cal** cho phép xem lịch trên hệ thống với cú pháp như sau:

```
cal [tùy-chọn] [<tháng> [<năm>]]
```

nếu không có tham số, lịch của tháng hiện thời sẽ được hiển thị.

Các tuỳ-chọn là:

- **-m :** chọn ngày Thứ hai là ngày đầu tiên trong tuần (mặc định là ngày Chủ nhật).
- **-j :** hiển thị số ngày trong tháng dưới dạng số ngày trong năm (ví dụ: ngày 1/11/2000 sẽ được hiển thị dưới dạng là ngày thứ 306 trong năm 2000, số ngày bắt đầu được tính từ ngày 1/1).
- **-y** : hiển thị lịch của năm hiện thời.

Ví dụ:

```
# cal 1 2001
```

January 2001

Su	Mo	Tu	We	Th	Fr	Sa
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

Khi nhập dòng lệnh trên, trên màn hình sẽ hiển thị lịch của tháng 1 năm 2001, mặc định chọn ngày chủ nhật là ngày bắt đầu của tuần. Dưới đây là ví dụ hiển thị số ngày trong tháng 3 dưới dạng số ngày trong năm 2001.

```
# cal -j 3 2001
```

March 2001

Su	Mo	Tu	We	Th	Fr	Sa
				60	61	62
63	64	65	66	67	68	69
70	71	72	73	74	75	76
77	78	79	80	81	82	83
84	85	86	87	88	89	90

2.5. Xem thông tin hệ thống

Lệnh **uname** cho phép xem thông tin hệ thống với cú pháp là:

```
uname [tùy-chọn]
```

Nếu không có tuỳ chọn thì hiện tên hệ điều hành.

Lệnh có các tùy chọn là:

- **-a, --all :** hiện tất cả các thông tin.
- **-m, --machine** : kiểu kiến trúc của bộ xử lý (i386, i486, i586, i686...).
- **-n, --nodename** : hiện tên của máy.
- **-r, --release :** hiện nhân của hệ điều hành.
- **-s, --sysname :** hiện tên hệ điều hành.
- **-p, --processor :** hiện kiểu bộ xử lý của máy chủ.

Ví dụ, nếu gõ lệnh

```
# uname -a
```

thì màn hình sẽ hiện ra như sau:

```
Linux linuxsrv.linuxvn.net 2.2.14-5.0 #1 Tue Mar 7 21:07:39 EST 2000 i686 unknown
#
```

Thông tin hiện ra có tất cả 6 trường là:

Tên hệ điều hành: Linux
Tên máy: linuxsrv.linuxvn.net
Tên nhân của hệ điều hành: 2.2.14-5.0
Ngày sản xuất: #1 Tue Mar 7 21:07:39 EST 2000
Kiểu kiến trúc bộ xử lý: i686
Kiểu bộ xử lý của máy chủ: unknown

Ví dụ nếu gõ lệnh:

```
# uname -spr
```

thì màn hình sẽ hiện ra như sau:

```
Linux 2.2.14-5.0 unknown
```

là tên hệ điều hành, tên nhân và kiểu bộ xử lý của máy chủ.

☞ ***Lưu ý:***

- Chúng ta làm rõ thêm nội dung lưu ý trong ***mục 1.3.1*** về tham số khóa kết hợp: Trong ví dụ trên đây khi viết tham số **-spr** là yêu cầu thực hiện lệnh **usame** với nghĩa kết hợp tình huống theo cả ba tham số khóa **-s**, **-p**, **-r**. Chú ý rằng, không thể viết **-s -p -r** thay cho **-spr** được. Như đã lưu ý ở ***mục 1.3.1*** trong nhiều lệnh của Linux cho phép viết kết hợp các tham số khóa theo cách thức như trên miễn là các tham số đó không xung khắc với nhau.

2.6. Thay đổi nội dung dấu nhắc shell

Trong Linux có hai loại dấu nhắc: dấu nhắc cấp một (dấu nhắc shell) xuất hiện khi nhập lệnh và dấu nhắc cấp hai (dấu nhắc nhập liệu) xuất hiện khi lệnh cần có dữ liệu được nhập từ bàn phím và tương ứng với hai biến nhắc tên là **PS1** và **PS2**.

PS1 là biến hệ thống tương ứng với dấu nhắc cấp 1: Giá trị của **PS1** chính là nội dung hiển thị của dấu nhắc shell. Để nhận biết thông tin hệ thống hiện tại, một nhu cầu đặt ra là cần thay đổi giá trị của các biến hệ thống **PS1** và **PS2**.

Linux cho phép thay đổi giá trị của biến hệ thống **PS1** bằng lệnh gán trị mới cho nó. Lệnh này có dạng:

```
# PS1='<dãy kí tự>'
```

Năm (5) kí tự đầu tiên của lệnh gán trên đây (**PS1='**) phải được viết liên tiếp nhau. Dãy kí tự nằm giữa cặp hai dấu nháy đơn (có thể sử dụng cặp hai dấu kép ") và không được phép chứa dấu nháy. Dãy kí tự này bao gồm các cặp kí tự điều khiển và các kí tự khác, cho phép có thể có dấu cách. Cặp kí tự điều khiển gồm hai kí tự, kí tự đầu tiên là dấu sổ xuôi "\" còn kí tự thứ hai nhận một trong các trường hợp liệt kê trong bảng dưới đây. Bảng dưới đây giới thiệu một số cặp ký tự điều khiển có thể được sử dụng khi muốn thay đổi dấu nhắc lệnh:

Cặp ký tự điều khiển	*Ý nghĩa*
\!	Hiển thị thứ tự của lệnh trong lịch sử
\#	Hiển thị thứ tự của lệnh
\$	Hiển thị dấu đô-la ($). Đối với siêu người dùng (super user), thì hiển thị dấu số hiệu (#)
\\	Hiển thị dấu sổ (\)
\d	Hiển thị ngày hiện tại
\h	Hiển thị tên máy (hostname)
\n	Ký hiệu xuống dòng
\s	Hiển thị tên hệ shell
\t	Hiển thị giờ hiện tại
\u	Hiển thị tên người dùng
\W	Hiển thị tên thực sự của thư mục hiện thời (ví dụ thư mục hiện thời là /mnt/hda1 thì tên thực sự của nó là /hda1)
\w	Hiển thị tên đầy đủ của thư mục hiện thời (ví dụ **/mnt/hda1**)

Ví dụ, hiện thời dấu nhắc shell có dạng:

```
root[may1 /hda1]#
```

Sau khi gõ lệnh

```
root@may1 /hda1]# PS1='[\h@\u \w : \d]\$'
```

thì dấu nhắc shell được thay đổi là:

```
[may1@root /mnt/hda1 : Fri Oct 27 ]#
```

ngoài việc đổi thứ tự giữa tên người dùng và máy còn cho chúng ta biết thêm về ngày hệ thống quản lý và tên đầy đủ của thư mục hiện thời.
Linux cung cấp cách thức hoàn toàn tương tự như đối với biến **PS1** để thay đổi giá trị biến hệ thống **PS2** tương ứng với dấu nhắc cấp hai.

2.7. Lệnh gọi ngôn ngữ tính toán số học

Linux cung cấp một ngôn ngữ tính toán với độ chính xác tùy ý thông qua lệnh **bc**. Khi yêu cầu lệnh này, người dùng được cung cấp một ngôn ngữ tính toán (và cho phép lập trình tính toán có dạng ngôn ngữ lập trình C) hoạt động theo thông dịch. Trong ngôn ngữ lập trình được cung cấp (tạm thời gọi là ngôn ngữ ***bc***), tồn tại rất nhiều công cụ hỗ trợ tính toán và lập trình tính toán: kiểu phép toán số học phong phú, phép toán so sánh, một số hàm chuẩn, biến chuẩn, cấu trúc điều khiển, cách thức định nghĩa hàm, cách thức thay đổi độ chính xác, đặt lời chú thích ... Chỉ cần sử dụng một phần nhỏ tác động của lệnh **bc**, chúng ta đã có một "máy tính số bấm tay" hiệu quả.

Cú pháp lệnh **bc**:

```
bc [tùy-chọn] [file...]
```

với các tuỳ chọn sau đây:

- **-l, --mathlib** : thực hiện phép tính theo chuẩn thư viện toán học (ví dụ: 5/5=1.00000000000000000000).

- **-w, --warn** : khi thực hiện phép tính không tuân theo chuẩn POSIX (POSIX là một chuẩn trong Linux) thì một cảnh báo xuất hiện.
- **-s, --standard** : thực hiện phép tính chính xác theo chuẩn của ngôn ngữ POSIX **bc.**
- **-q, --quiet** : không hiện ra lời giới thiệu về phần mềm GNU khi dùng **bc.**

Tham số file là tên file chứa chương trình viết trên ngôn ngữ ***bc***, khi lệnh **bc** thực hiện sẽ tự động chạy các file chương trình này (Nếu có nhiều tham số thì có nghĩa sẽ chạy nhiều chương trình liên tiếp nhau).

Dưới đây là một ví dụ sử dụng lệnh **bc** ở dạng đơn giản nhất.

Khi gõ lệnh tại dấu nhắc:

```
# bc -l
```

màn hình xuất hiện lời giới thiệu về GNU khi dùng **bc** và ngôn ngữ ***bc*** được kích hoạt để phục vụ người dùng.

```
bc 1.05
Copyright 1991, 1992, 1993, 1994, 1997, 1998 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type `warranty'.
5^3
125
12+12+78*7-62/4
554.50000000000000000000
a=4
a^a
256
a*78
312
b=45
a*b
180
a/b
.08888888888888888888
a%b
.00000000000000000040
```

ở đây * là phép nhân, ^ là phép tính luỹ thừa, / là phép chia lấy thương, % là chia lấy phần dư.

☞ ***Lưu ý:***

- Ngôn ngữ lập trình tính toán ***bc*** là một ngôn ngữ rất mạnh có nội dung hết sức phong phú cho nên trong khuôn khổ của tài liệu này không thể mô tả hết các nội dung của ngôn ngữ đó được. Chúng ta cần sử dụng lệnh **man bc** để nhận được thông tin đầy đủ về lệnh **bc** và ngôn ngữ tính toán ***bc***.
- ở đây trình bày sơ bộ một số yếu tố cơ bản nhất của ngôn ngữ đó (***bt*** là viết tắt của biểu thức, ***b*** là viết tắt của biến):

Các phép tính: - bt: lấy đối; ++ b, --b, b ++, b --: phép toán tăng, giảm b; các phép toán hai ngôi cộng +, trừ -, nhân *, chia /, lấy phần dư %, lũy thừa nguyên bậc ^; gán =; gán sau khi thao tác <thao tác>=; các phép toán so sánh <, <=, >, >=, bằng ==, khác != ...

Phép so sánh cho 1 nếu đúng, cho 0 nếu sai.

Bốn biến chuẩn là ***scale*** số lượng chữ số phần thập phân; ***last*** giá trị tính toán cuối cùng; ***ibase*** cơ số hệ đếm đối với input và ***obase*** là cơ số hệ đếm với output (ngầm định hai biến này có giá trị 10).

Các hàm chuẩn sin s (bt); cosin c (bt); arctg a (bt); lôgarit tự nhiên l (bt); mũ cơ số tự nhiên e (bt); hàm Bessel bậc nguyên n của bt là j (n, bt).

CHƯƠNG 3. HỆ THỐNG FILE

3.1 Tổng quan về hệ thống file

3.1.1. Một số khái niệm

Người dùng đã từng làm việc với hệ điều hành DOS/Windows thì rất quen biết với các khái niệm: file (tập tin), thư mục, thư mục hiện thời ... Để đảm bảo tính hệ thống và thuận tiện cho người dùng chưa từng làm việc thành thạo với một hệ điều hành nào khác, chương này vẫn giới thiệu về các khái niệm này một cách sơ bộ.

Một đối tượng điển hình trong các hệ điều hành đó là ***file***. File là một tập hợp dữ liệu có tổ chức được hệ điều hành quản lý theo yêu cầu của người dùng. Cách tổ chức dữ liệu trong file thuộc về chủ của nó là người đã tạo ra file. File có thể là một văn bản (trường hợp đặc biệt là chương trình nguồn trên C, PASCAL, shell script ...), một chương trình ngôn ngữ máy, một tập hợp dữ liệu ... Hệ điều hành tổ chức việc lưu trữ nội dung file trên các thiết bị nhớ lâu dài (chẳng hạn đĩa từ) và đảm bảo các thao tác lên file. Chính vì có hệ điều hành đảm bảo các chức năng liên quan đến file nên người dùng không cần biết file của mình lưu ở vùng nào trên đĩa từ, bằng cách nào đọc/ghi lên các vùng của đĩa từ mà vẫn thực hiện được yêu cầu tìm kiếm, xử lý lên các file.

Hệ điều hành quản lý file theo tên gọi của file (tên file) và một số thuộc tính liên quan đến file. Trước khi giới thiệu một số nội dung liên quan đến tên file và tên thư mục, chúng ta giới thiệu sơ bộ về khái niệm thư mục.

Để làm việc được với các file, hệ điều hành không chỉ quản lý nội dung file mà còn phải quản lý các thông tin liên quan đến các file. Thư mục (directory) là đối tượng được dùng để chứa thông tin về các file, hay nói theo một cách khác, thư mục chứa các file. Các thư mục cũng được hệ điều hành quản lý trên vật dẫn ngoài và vì vậy, theo nghĩa này, thư mục cũng được coi là file song trong một số trường hợp để phân biệt với "file" thư mục, chúng ta dùng thuật ngữ ***file thông thường***. Khác với file thông thường, hệ điều hành lại quan tâm đến nội dung của thư mục.

Một số nội dung sau đây liên quan đến tên file (bao gồm cả tên thư mục):

- Tên file trong Linux có thể dài tới 256 ký tự, bao gồm các chữ cái, chữ số, dấu gạch nối, gạch chân, dấu chấm. Tên thư mục/file trong Linux có thể có nhiều hơn một dấu chấm, ví dụ: **This_is.a.VERY_long.filename**. Nếu trong tên file có dấu chấm "." thì xâu con của tên file từ dấu chấm cuối cùng được gọi là phần mở rộng của tên file (hoặc file). Ví dụ, tên file trên đây có phần mở rộng là **.filename**. Chú ý rằng khái niệm phần mở rộng ở đây không mang ý nghĩa như một số hệ điều hành khác (chẳng hạn như MS-DOS).

☞ ***Lưu ý:***

- Chúng ta nên lưu ý rằng, không phải ký tự nào cũng có nghĩa. Nếu có hai file chỉ khác nhau ở ký tự cuối cùng, thì đối với Linux, đó là hai file có thể trùng tên. Bởi lẽ, Linux chỉ lấy 32 hay 64 ký tự đầu tiên trong tên file mà thôi (tùy theo phiên bản Linux), phần tên file còn lại dành cho chủ của file, Linux theo dõi thông tin, nhưng thường không xem các ký tự đứng sau ký tự thứ 33 hay 65 là quan trọng đối với nó.

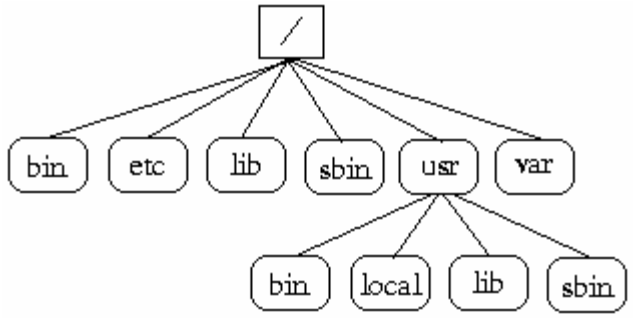
- Xin nhắc lại lưu ý về phân biệt chữ hoa và chữ thường đối với tên thư mục/file, ví dụ hai file **FILENAME.tar.gz** và **filename.tar.gz** là hai file khác nhau.

- Nếu trong tên thư mục/file có chứa khoảng trống, sẽ phải đặt tên thư mục/file vào trong cặp dấu nháy kép để sử dụng thư mục/file đó. Ví dụ, để tạo thư mục có tên là "My document" chẳng hạn, hãy đánh dòng lệnh sau:

```
# mkdir "My document"
```

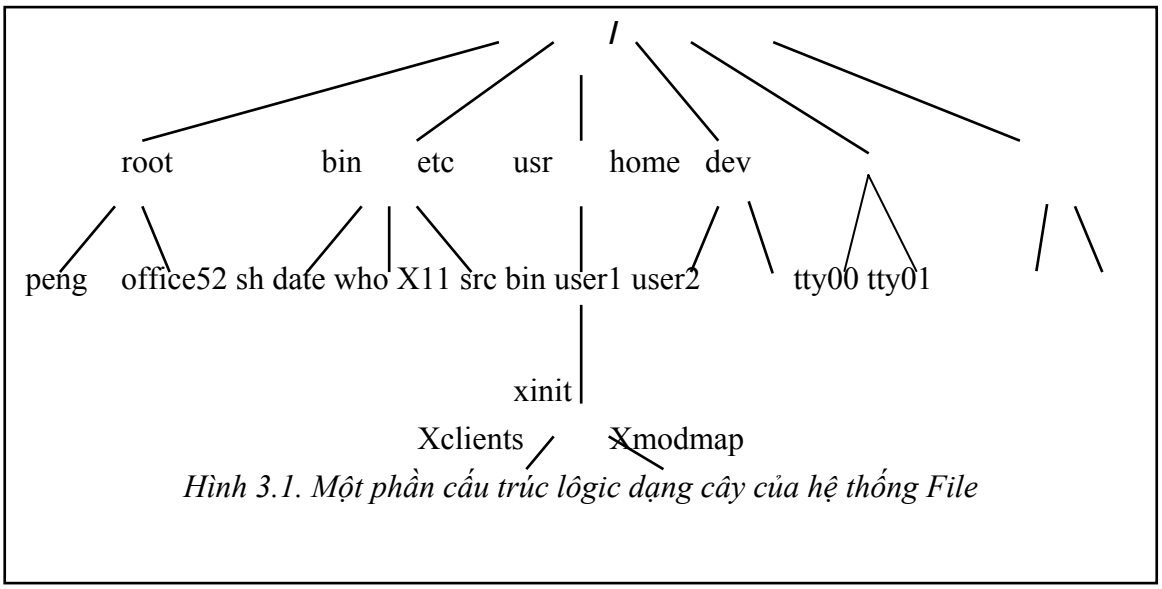
- Một số ký tự sau không được sử dụng trong tên thư mục/file: !, *, $, &, # ...
- Khi sử dụng chương trình **mc** (Midnight Commander), việc hiển thị tên file sẽ bổ sung một kí tự theo nghĩa: dấu "*" cho file khả thi trong Linux, dấu "~" cho file sao lưu, dấu "." cho file ẩn, dấu "@" cho file liên kết...

Tập hợp tất cả các file có trong hệ điều hành được gọi là ***hệ thống file*** là một hệ thống thống nhất. Bởi chính từ cách thức sử dụng thư mục, hệ thống file được tổ chức lôgic theo dạng hình cây: Hệ thống file được xuất phát từ một thư mục gốc (được kí hiệu là "**/**") và cho phép tạo ra thư mục con trong một thư mục bất kỳ. Thông thường, khi khởi tạo Linux đã có ngay hệ thống file của nó. Hình 3.1. cho minh họa một phần trong cây lôgic của hệ thống file.



Để chỉ một file hay một thư mục, chúng ta cần đưa ra một đường dẫn, ví dụ để đường dẫn xác định file **`Xclients`** trong hình 3.1. chúng ta viết như sau:

/etc/X11/xinit/Xclients



Hình 3.1. Một phần cấu trúc lôgic dạng cây của hệ thống File

Đường dẫn này cho biết **Xclients** nằm trong **xinit**, **xinit** nằm trong **X11**, **X11** nằm trong **etc** và **etc** nằm trong gốc **/**.

Tên file thường là tham số thực sự khi gõ lệnh và công việc gõ lệnh trở nên rất nặng nề đối với người dùng nếu như trong lệnh phải gõ một đường dẫn dài theo dạng trên (được biết với tên gọi là ***đường dẫn tuyệt đối***). Vì vậy, Linux (cũng như nhiều hệ điều hành khác) sử dụng khái niệm ***thư mục hiện thời*** của mỗi người dùng làm việc trong hệ thống. Thư mục hiện thời là một thư mục trong hệ thống file mà hiện thời "người dùng đang ở đó".

Qua thư mục hiện thời, Linux cho phép người dùng chỉ một file trong lệnh ngắn gọn hơn nhiều. Ví dụ, nếu thư mục hiện thời là thư mục ***xinit*** thì để chỉ file đã nói, người dùng chỉ cần viết **Xclients** hoặc **./Xclients** trong đó kí hiệu "**.**" để chỉ thư mục hiện thời. Đường dẫn được xác định qua thư mục hiện thời được gọi là ***đường dẫn tương đối***.

Khi một người dùng đăng nhập vào hệ thống, Linux luôn chuyển người dùng vào thư mục riêng, và tại thời điểm đó thư mục riêng là thư mục hiện thời của người dùng. Thư mục riêng của siêu người dùng là **/root**, thư mục riêng của người dùng có tên là ***user1*** là **/home/user1** ... Linux cho phép dùng lệnh **cd** để chuyển sang thư mục khác (lấy thư mục khác làm thư mục hiện thời). Hai dấu chấm ".." được dùng để chỉ thư mục ngay trên thư mục hiện thời (cha của thư mục hiện thời).

Linux còn cho phép ghép một hệ thống file trên một thiết bị nhớ (đĩa mềm, vùng đĩa cứng chưa được đưa vào hệ thống file) thành một thư mục con trong hệ thống file của hệ thống bằng lệnh ***mount***. Các hệ thống file được ghép thuộc vào các kiểu khác nhau.

Hai mục tiếp theo (3.1.2 và 3.1.3.) giới thiệu những nội dung sâu hơn về hệ thống file Linux.

3.1.2. Sơ bộ kiến trúc nội tại của hệ thống file

Trên đĩa từ, hệ thống file được coi là dãy tuần tự các khối lôgic mỗi khối chứa hoặc 512B hoặc 1024B hoặc bội của 512B là cố định trong một hệ thống file. Trong hệ thống file, các khối dữ liệu được địa chỉ hóa bằng cách đánh chỉ số liên tiếp, mỗi địa chỉ được chứa trong 4 byte (32 bit).

Cấu trúc nội tại của hệ thống file bao gồm 4 thành phần kế tiếp nhau: Boot block (dùng để khởi động hệ thống), Siêu khối (Super block), Danh sách inode và Vùng dữ liệu.

Dưới đây, chúng ta xem xét sơ lược nội dung các thành phần cấu trúc nội tại một hệ thống file.

Siêu khối

Siêu khối chứa nhiều thông tin liên quan đến trạng thái của hệ thống file. Trong siêu khối có các trường sau đây:

- Kích thước của danh sách inode (khái niệm inode sẽ được giải thích trong mục sau): định kích cỡ vùng không gian trên Hệ thống file quản lý các inode.
- Kích thước của hệ thống file.

Hai kích thước trên đây tính theo đơn vị dung lượng bộ nhớ ngoài,

- Một danh sách chỉ số các khối rỗi (thường trực trên siêu khối) trong hệ thống file.

Chỉ số các khối rỗi thường trực trên siêu khối được dùng để đáp ứng nhu cầu phân phối mới. Chú ý rằng, danh sách chỉ số các khối rỗi có trên siêu khối chỉ là một bộ phận của tập tất cả các khối rỗi có trên hệ thống file.

- Chỉ số của khối rỗi tiếp theo trong danh sách các khối rỗi.

Chỉ số khối rỗi tiếp theo dùng để hỗ trợ việc tìm kiếm tiếp các khối rỗi: bắt đầu tìm từ khối có chỉ số này trở đi. Điều đó có nghĩa là mọi khối có chỉ số không lớn hơn chỉ số này hoặc có trong danh sách các khối rỗi thường trực hoặc đã được cấp phát cho một file nào đó. Nhiều thao tác tạo file mới, xoá file, thay đổi nội dung file v.v. cập nhật các thông tin này.

- Một danh sách các inode rỗi (thường trực trên siêu khối) trong hệ thống file.

Danh sách này chứa chỉ số các inode rỗi được dùng để phân phối ngay được cho một file mới được khởi tạo. Thông thường, danh sách này chỉ chứa một bộ phận các inode rỗi trên hệ thống file.

- Chỉ số inode rỗi tiếp theo trong danh sách các inode rỗi.

Chỉ số inode rỗi tiếp theo định vị việc tìm kiếm tiếp thêm inode rỗi: bắt đầu tìm từ inode có chỉ số này trở đi. Điều đó có nghĩa là mọi inode có chỉ số không lớn hơn chỉ số này hoặc có trong danh sách các inode rỗi thường trực hoặc đã được tương ứng với một file nào đó.

Hai tham số trên đây tạo thành cặp xác định được danh sách các inode rỗi trên hệ thống file các thao tác tạo file mới, xoá file cập nhật thông tin này.

- Các trường khóa (lock) danh sách các khối rỗi và danh sách inode rỗi: Trong một số trường hợp, chẳng hạn khi hệ thống đang làm việc thực sự với đĩa từ để cập nhật các danh sách này, hệ thống không cho phép cập nhật tới hai danh sách nói trên.
- Cờ chỉ dẫn về việc siêu khối đã được biến đổi: Định kỳ thời gian siêu khối ở bộ nhớ trong được cập nhật lại vào siêu khối ở đĩa từ và vì vậy cần có thông tin về việc siêu khối ở bộ nhớ trong khác với nội dung ở bộ nhớ ngoài: nếu hai bản không giống nhau thì cần phải biến đổi để chúng được đồng nhất.
- Cờ chỉ dẫn rằng hệ thống file chỉ có thể đọc (cấm ghi): Trong một số trường hợp, hệ thống đang cập nhật thông tin từ bộ nhớ ngoài thì chỉ cho phép đọc đối với hệ thống file,
- Số lượng tổng cộng các khối rỗi trong hệ thống file,
- Số lượng tổng cộng các inode rỗi trong hệ thống file,
- Thông tin về thiết bị,
- Kích thước khối (đơn vị phân phối dữ liệu) của hệ thống file. Hiện tại kích thước phổ biến của khối là 1KB.

Trong thời gian máy hoạt động, theo từng giai đoạn, nhân sẽ đưa siêu khối lên đĩa nếu nó đã được biến đổi để phù hợp với dữ liệu trên hệ thống file.
Một trong khái niệm cốt lõi xuất hiện trong hệ thống file đó là inode. Các đối tượng liên quan đến khái niệm này sẽ được trình bày trong các mục tiếp theo.

Inode

Mỗi khi một quá trình khởi tạo một file mới, nhân hệ thống sẽ gán cho nó một inode chưa sử dụng. Để hiểu rõ hơn về inode, chúng ta xem xét sơ lược mối quan hệ liên quan giữa file dữ liệu và việc lưu trữ trên vật dẫn ngoài đối với Linux.

Nội dung của file được chứa trong vùng dữ liệu của hệ thống file và được phân chia các khối dữ liệu (chứa nội dung file) và hình ảnh phân bố nội dung file có trong một inode tương ứng. Liên kết đến tập hợp các khối dữ liệu này là một inode, chỉ thông qua inode mới

có thể làm việc với dữ liệu tại các khối dữ liệu: Inode chứa dựng thông tin về tập hợp các khối dữ liệu nội dung file. Có thể quan niệm rằng, tổ hợp gồm inode và tập các khối dữ liệu như vậy là một file vật lý: inode có thông tin về file vật lý, trong đó có địa chỉ của các khối nhớ chứa nội dung của file vật lý. Thuật ngữ inode là sự kết hợp của hai từ index với node và được sử dụng phổ dụng trong Linux.

Các inode được phân biệt nhau theo chỉ số của inode: đó chính là số thứ tự của inode trong danh sách inode trên hệ thống file. Thông thường, hệ thống dùng 2 bytes để lưu trữ chỉ số của inode. Với cách lưu trữ chỉ số như thế, không có nhiều hơn 65535 inode trong một hệ thống file.

Như vậy, một file chỉ có một inode song một file lại có một hoặc một số tên file. Người dùng tác động thông qua tên file và tên file lại tham chiếu đến inode (tên file và chỉ số inode là hai trường của một phần tử của một thư mục). Một inode có thể tương ứng với một hoặc nhiều tên file, mỗi tương ứng như vậy được gọi là một liên kết. Inode được lưu trữ tại vùng danh sách các inode.

Trong quá trình làm việc, Linux dùng một vùng bộ nhớ, được gọi là bảng inode (trong một số trường hợp, nó còn được gọi tường minh là bảng sao in-core inode) với chức năng tương ứng với vùng danh sách các inode có trong hệ thống file, hỗ trợ cho quá trình truy nhập dữ liệu trong hệ thống file. Nội dung của một in-core inode không chỉ chứa các thông tin trong inode tương ứng mà còn được bổ sung các thông tin mới giúp cho quá trình xử lý inode.

Chúng ta xem xét cấu trúc nội tại của một inode để thấy được sự trình bày nội tại của một file. Inode bao gồm các trường thông tin sau đây:

- Kiểu file. Trong Linux phân loại các kiểu file: file thông thường (regular), thư mục, đặc tả kí tự, đặc tả khối và ống dẫn FIFO (pipes). Linux quy định trường kiểu file có giá trị 0 tương ứng đó là inode chưa được sử dụng.
- Quyền truy nhập file. Trong Linux, file là một tài nguyên chung của hệ thống vì vậy quyền truy nhập file được đặc biệt quan tâm để tránh những trường hợp truy nhập không hợp lệ. Đối với một inode, có 3 mức quyền truy nhập liên quan đến các đối tượng:
  - ❖ mức chủ của file (đối tượng này được ký hiệu là ***u***: từ chữ user),
  - ❖ mức nhóm người dùng của chủ nhân của file (đối tượng này được ký hiệu là ***g***: từ chữ group),
  - ❖ mức người dùng khác (đối tượng này được ký hiệu là ***a***: từ chữ all).

Quyền truy nhập là đọc, ghi, thực hiện hoặc một tổ hợp nào đó từ nhóm gồm 3 quyền trên. Chú ý rằng, quyền thực hiện đối với một thư mục tương ứng với việc cho phép tìm một tên file có trong thư mục đó.

- Số lượng liên kết đối với inode: Đây chính là số lượng các tên file trên các thư mục được liên kết với inode này,
- Định danh chủ nhân của inode,
- Định danh nhóm chủ nhân: xác định tên nhóm người dùng mà chủ file là một thành viên của nhóm này,
- Độ dài của file tính theo byte,
- Thời gian truy nhập file:
  - ❖ thời gian file được sửa đổi muộn nhất,

- ❖ thời gian file được truy nhập muộn nhất,
- ❖ thời gian file được khởi tạo,

- Bảng địa chỉ chứa các địa chỉ khối nhớ chứa nội dung file. Bảng này có 13 phần tử địa chỉ, trong đó có 10 phần tử trực tiếp, 1 phần tử gián tiếp bậc 1, 1 phần tử gián tiếp bậc 2 và một phần tử gián tiếp bậc 3 (chi tiết có trong phần sau).

Nội dung của file thay đổi khi có thao tác ghi lên nó; nội dung của một inode thay đổi khi nội dung của file thay đổi hoặc thay đổi chủ hoặc thay đổi quyền hoặc thay đổi số liên kết.

Ví dụ về nội dung một inode như sau:

type regular
perms rwxr-xr-x
links 2
owner 41CT
group 41CNTT
size 5703 bytes
accessed Sep 14 1999 7:30 AM
modified Sep 10 1999 1:30 PM
inode Aug 1 1995 10:15 AM
Các phần tử địa chỉ dữ liệu

Bản sao in-core inode còn bổ sung thêm trường trạng thái của in-core inode.

- Trường trạng thái của in-core inode có các thông tin sau:
  - ❖ inode đã bị khoá,
  - ❖ một quá trình đang chờ đợi khi inode tháo khóa,
  - ❖ in-core inode khác với inode do sự thay đổi dữ liệu trong inode,
  - ❖ in-core inode khác với inode do sự thay đổi dữ liệu trong file,
  - ❖ số lượng các tên file nối với file đang được mở,
  - ❖ số hiệu thiết bị lôgic của hệ thống file chứa file nói trên
  - ❖ chỉ số inode: dùng để liên kết với inode trên đĩa,
  - ❖ các móc nối tới các in-core inode khác. Trong bộ nhớ trong, các in-core inode được liên kết theo một hàng băm và một danh sách tự do. Trong danh sách hàng băm các in-core inode hòa hợp theo số hiệu thiết bị lôgic và số hiệu inode.

Trong quá trình hệ thống làm việc, nảy sinh khái niệm inode tích cực nếu như có một quá trình đang làm việc với inode đó (như mở file).

Một inode thuộc vào danh sách các inode rỗi khi không có file vật lý nào tương ứng với inode đó.

Bảng chứa địa chỉ khối dữ liệu của File trong UNIX

Bảng chứa địa chỉ khối dữ liệu của file gồm 13 phần tử với 10 phần tử trực tiếp và 3 phần tử gián tiếp: Mỗi phần tử có độ dài 4 bytes, chứa một số hiệu của một khối nhớ trên đĩa. Mỗi phần tử trực tiếp trỏ tới 1 khối dữ liệu thực sự chứa nội dung file. Phần tử gián tiếp bậc 1 (single indirect) trỏ tới 1 khối nhớ ngoài. Khác với phần tử trực tiếp, khối nhớ ngoài này không dùng để chứa dữ liệu của file mà lại chứa danh sách chỉ số các khối nhớ ngoài

và chính các khối nhớ ngoài này mới thực sự chứa nội dung file. Như vậy, nếu khối có độ dài 1KB và một chỉ số khối ngoài có độ dài 4 bytes thì địa chỉ gián tiếp cho phép định vị không gian trên đĩa lưu trữ dữ liệu của file tới 256KB (Không gian bộ nhớ ngoài trong vùng dữ liệu phải dùng tới là 257KB). Tương tự đối với các phần tử gián tiếp mức cao hơn.

Cơ chế quản lý địa chỉ file như trên cho thấy có sự phân biệt giữa file nhỏ với file lớn. File nhỏ có độ dài bé hơn và theo cách tổ chức như trên, phương pháp truy nhập sẽ cho phép tốc độ nhanh hơn, đơn giản hơn do chỉ phải làm việc với các phần tử trực tiếp. Khi xử lý, thuật toán đọc File tiến hành theo các cách khác nhau đối với các phần tử trực tiếp và gián tiếp.

Cơ chế tổ chức lưu trữ nội dung Fle như đã trình bày cho phép độ dài file có thể lên tới $(2^{24}+2^{16} + 2^{8}+10)$ khối.

Vùng dữ liệu bao gồm các khối dữ liệu, mỗi khối dữ liệu được đánh chỉ số để phân biệt. Khối trên vùng dữ liệu được dùng để chứa nội dung các file, nội dung các thư mục và nội dung các khối định vị địa chỉ của các file. Chú ý rằng, chỉ số của khối dữ liệu được chứa trong 32 bit và thông tin này xác định dung lượng lớn nhất của hệ thống file.

3.1.3. Một số thuật toán làm việc với inode

Hệ thống lời gọi hệ thống file

Khi làm việc với file thường thông qua lời gọi hệ thống. Một số lời gọi hệ thống thường gặp như mở file open, đóng file close, đọc nội dung file read, ghi nội dung file write v.v.

Bảng dưới đây thống kê các lời gọi hệ thống làm việc với hệ thống file và phân loại theo chức năng của mỗi lời gọi hệ thống (một lời gọi có thể được nhắc tới một số lần):

Thời điểm sử dụng file	Sử dụng ***namei***		gán inode	thuộc tính file	Vào-ra file	Cấu trúc hệ thống file	Quản lý cây
open	***open***	***stat***	***creat***	***chown***	***read***	***mont***	***chdir***
creat	***creat***	***link***	***mknod***	***chmod***	***write***	***umount***	***chown***
dup	***chdir***	***unlink***	***link***	***stat***	***cseek***		
pipe	***chroot***	***mknod***	***unlink***				
close	***chown***	***mount***					
	chmod	***umount***					

Thuật toán hệ thống file mức thấp

Namei						
iget	***iput***	***Ialloc***	***ifree***	***alloc***	***free***	***bmap***

Thuật toán định vị buffer

getblk	***brelse***	***Bread***	***breada***	***bwrite***

Hình 3.2. Tổng thể về lời gọi hệ thống File

Chúng ta xem xét một số thuật toán làm việc với inode.

Thuật toán truy nhập tới inode (iget)

Nhiều tình huống đòi hỏi thuật toán ***iget***, chẳng hạn như, một quá trình mở một file mới hoặc tạo một file mới v.v.. Thuật toán ***iget*** cấp phát một bản in-core inode đối với một số hiệu inode. Tuy nhiên, trong trường hợp chưa có bản sao in-core inode thì dể có nội dung

của nó cần phải đọc được nội dung của inode đó và cần định vị khối dữ liệu chứa inode đã cho. Công thức liên quan đến khối đĩa từ chứa inode để có thể đọc vào bộ nhớ trong như sau:

Chỉ số khối chứa inode = (số hiệu inode - 1) / (số lượng inode trong một khối nhớ) + chỉ số khối nhớ đầu tiên chứa danh sách inode trên đĩa.

Sau khi đã đọc khối đĩa chứa inode vào bộ nhớ trong, để xác định chính xác vị trí của inode, chúng ta có công thức sau:

Byte vị trí đầu tiên = ((số hiệu inode - 1) mod (số lượng inode trong một khối nhớ))*độ dài một inode

Ví dụ, nếu như mỗi inode đĩa chiếm 64 bytes, mỗi khối đĩa chứa 8 inode đĩa thì inode số 8 sẽ bắt đầu từ byte thứ 448 trên khối đĩa đầu tiên trong vùng danh sách các inode.
Để ý rằng, khi làm việc với một hệ thống file thì super block của nó luôn có mặt trong bộ nhớ trong để hệ thống có những thông tin làm việc. Chú ý rằng, trong super block có một danh sách các inode rỗi (trên nó) và một danh sách các khối rỗi.

Thuật toán ***iget*** nhận một inode để cho nó tích cực và điều đó tùy thuộc vào một số tình huống sau đây:

- Nếu inode không tồn tại trong vùng đệm mà lại không thuộc danh sách các inode rỗi trên super block thì hệ thống phải thông báo một lỗi đã được gặp. Lỗi này xảy ra do yêu cầu một inode không còn đủ vùng đệm làm việc với file nữa (tương ứng với trường hợp trong MS-DOS thông báo: too many files opened),
- inode đã có trong vùng đệm các inode trên hệ thống file (đã có in-core inode). Trong trường hợp này xử lý theo hai bước:
  + inode tương ứng đã bị khóa bởi một quá trình khác: lúc đó phải đợi cho đến khi quá trình trước đây không khóa inode nữa. Sau khi được tháo khóa inode có thể trở thành tích cực hoặc rỗi,
  + Nếu inode ở danh sách các inode rỗi thì loại bỏ nó khỏi danh sách này bằng cách đặt inode sang tích cực.
- inode không tồn tại trên vùng đệm tuy nhiên danh sách các inode rỗi khác rỗng. Khi danh sách các inode này khác rỗng, có nghĩa là có những inode không có giá trị: loại bỏ nó và đặt inode mới vào thay thế.

Thuật toán iput loại bỏ inode

Thuật toán ***iput*** có chức năng đối ngẫu với thuật toán ***iget***: cần tháo bớt sự xuất hiện của một inode, chẳng hạn khi chương trình thực hiện thao tác đóng file.

Khác với trường hợp thuật toán ***iget***, thuật toán ***iput*** không nảy sinh tình huống sai sót.

Trong thuật toán này, khi một quá trình không làm việc với một file được liên kết với một inode nữa thì một số tình huống xẩy ra:

- Hệ thống giảm số lượng file tích cực đi 1,
- Nếu số lượng file tích cực là 0 thì:
  + Nếu đó là lệnh xoá file thì trước đó hệ thống đã thực hiện thao tác giảm số liên kết với inode đi 1 và vì vậy có thể số lượng liên kết trở thành 0, có nghĩa là sự tồn tại của file vật lý không còn. Khi đó, chúng ta thực hiện việc xoá thức sự file nói trên bằng một số thao tác: giải phóng các khối dữ liệu, đặt kiểu file của inode là 0 và giải phóng inode.
  + khi số liên kết >0 thi cần cập nhật sự thay đổi của inode lên đĩa từ.

- Trong trường hợp số lượng file tích cực vẫn dương thì không thực hiện thao tác gì.
  Chú ý là trong thuật toán này có sử dụng thuật toán ***ifree***.

Thuật toán ialloc gán inode cho một file mới

Khi một file mới được xuất hiện, chẳng hạn khởi tạo file ***creat***, phải cung cấp một inode cho file và thuật toán ***ialloc*** đáp ứng đòi hỏi trên.

Hoạt động của thuật toán ***ialloc*** được giải thích như sau:

- kiểm tra danh sách inode rỗi trên super block, xảy ra một trong hai trường hợp hoặc danh sách rỗng hoặc không rỗng,
- Nếu danh sách không rỗng thì lấy một inode tiếp theo cho file, khởi tạo các giá trị ban đầu của inode đó và giảm số inode rỗi trên super trên super block.
- Nếu danh sách các inode rỗi trên super block là rỗng: tìm kiếm trên hệ thống file những inode rỗi để tải vào danh sách các inode rỗi trên super block. Nếu danh sách đó đầy hoặc không tìm thấy được nữa thì gán một inode cho file. Nếu danh sách inode rỗi trên super block là rỗng và không tìm thấy inode rỗi trên đĩa thì sẽ có thông báo lỗi.

Trên danh sách các inode rỗi, nhân lưu giữ một inode được gọi là inode nhớ, chính là inode cuối cùng được tìm thấy để sau này thuận lợi cho tìm kiếm.

Thuật toán ifree tải một inode rỗi trên đĩa vào danh sách các inode rỗi trên super block

Thuật toán namei tìm chỉ số một inode theo tên file

Thuật toán ***namei*** là một thuật toán phổ dụng, nhiều thuật toán làm việc với file phải sử dụng ***namei***. Từ tên một đường dẫn file/thư mục, thuật toán ***namei*** cho inode tương ứng.

Thuật toán cấp phát dữ liệu trên đĩa

Khi nhân muốn cấp phát một khối dữ liệu, nó sẽ cấp phát khối rỗi tiếp theo đã được ghi nhận trong super block. Khi một khối dữ liệu đã được cấp cho một file thì nó chỉ được cấp phát lại khi nó trở thành rỗi. Nếu không còn khối rỗng nào trên hệ thống file mà lại có nhu cầu cung cấp khối thì nhân sẽ thông báo lỗi.

3.1.4. Hỗ trợ nhiều hệ thống File

Các phiên bản đầu tiên của Linux chỉ hỗ trợ một hệ thống file duy nhất đó là hệ thống file minix. Sau đó, với sự mở rộng nhân, cộng đồng Linux đã thêm vào nó rất nhiều kiểu hệ thống file khác nhau và Linux trở thành một hệ điều hành hỗ trợ rất nhiều hệ thống file. Dưới đây là một số hệ thống file thông dụng trong các hệ điều hành khác nhau được Linux hỗ trợ.

- Hệ thống file ADFS: ADFS viết tắt của Acorn Disc Filing System là hệ thống file chuẩn trên hệ điều hành RiscOS. Với sự hỗ trợ này, Linux có thể truy cập vào các phân vùng đĩa định dạng theo hệ thống file ADFS.
- Hệ thống file AFFS: AFFS (The Amiga Fast File System) là một hệ thống file phổ biến của hệ điều hành AmigaOS phiên bản 1.3 chạy trên các máy Amiga.

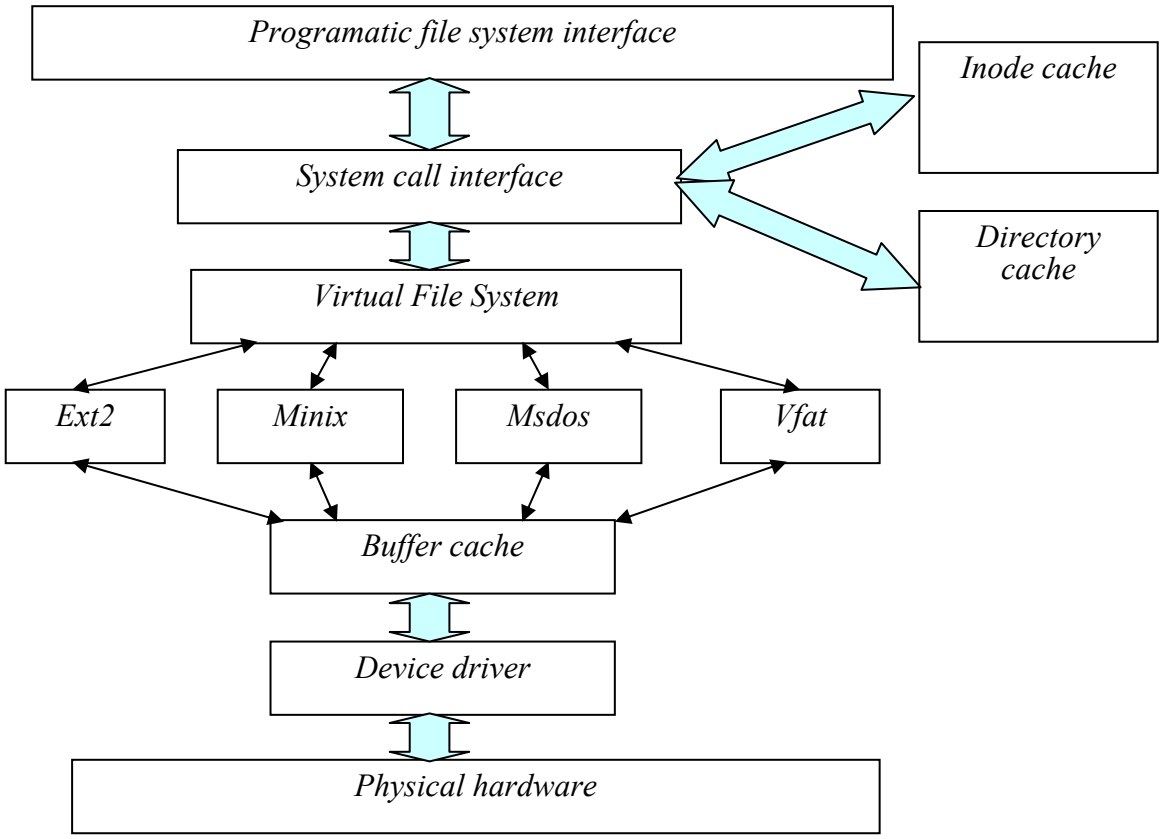
- Hệ thống file CODA: CODA là một hệ thống file mạng cho phép người dùng có thể kết gán các hệ thống file từ xa và truy cập chúng như các hệ thống file cục bộ (local).
- Hệ thống file DEVPTS: Hệ thống file cho Unix98 PTYs.
- Hệ thống file EFS: Đây là một dạng hệ thống file sử dụng cho CDROM.
- Hệ thống file EXT2: Hệ thống file EXT2 (The second extended filesystem) là hệ thống được dùng chủ yếu trên các phiên bản của hệ điều hành Linux. Chúng ta sẽ trở lại ngiên cứu hệ thống file này trong các phần sau.
- Hệ thống file HFS: Đây là hệ thống file chạy trên các máy Apple Macintosh.
- Hệ thống file HPFS: HPFS là hệ thống file được sử dụng trong hệ điều hành OS/2. Linux hỗ trợ hệ thống file này ở mức chỉ đọc (read only).
- Hệ thống file ISOFS: Đây là hệ thống file được sử dụng cho các đĩa CD. Hệ thống thông dụng nhất cho các đĩa CD hiện nay là ISO 9660. Với sự hỗ trợ này, hệ thống Linux có thể truy cập dữ liệu trên các đĩa CD.
- Hệ thống file MINIX: MINIX là hệ thống file đầu tiên mà Linux hỗ trợ. Hệ thống file này được sử dụng trong hệ điều hành Minix và một số hệ thống Linux cũ.
- Hệ thống file MSDOS: Với sự hỗ trợ này, hệ thống Linux có thể truy cập được các phân vùng của hệ điều hành MSDOS. Linux cũng có thể sử dụng kiểu MSDOS để truy cập các phân vùng của Window 95/98 tuy nhiên khi đó, các ưu điểm của hệ điều hành Window sẽ không còn giá trị ví dụ như tên file chỉ tối đa 13 ký tự (kể cả mở rộng).
- Hệ thống file NFS: NFS (Network File System) là một hệ thống file trên mạng hỗ trợ việc truy cập dữ liệu từ xa giống như hệ thống file CODA. Với NFS, các máy chạy Linux có thể chia sẻ các phân vùng đĩa trên mạng để sử dụng như là các phân vùng cục bộ của chính máy mình.
- Hệ thống file NTFS: Với sự hỗ trợ này, hệ thống Linux có thể truy cập vào các phân vùng của hệ điều hành Microsoft Window NT.
- Hệ thống file PROC: Đây là một hệ thống file đặc biệt được Linux hỗ trợ. Hệ thống file PROC không chiếm một phân vùng nào của hệ thống và cũng không quản lý các dữ liệu lưu trữ trên đĩa. PROC hiển thị nội dung của chính nhân hệ thống. Các file trong hệ thống file PROC lưu trữ các thông tin về trạng thái hiện hành của nhân. Thông tin về mỗi một tiến trình đang thực hiện trong hệ thống được lưu trong một thư mục mang tên ứng với chỉ số process ID của tiến trình đó. Người dùng có thể sử dụng hệ thống file PROC để lấy các thông tin về nhân cũng như sửa đổi một số giá trị của nhân thông qua sửa đổi nội dung của các file trong hệ thống file này. Tuy nhiên, việc sửa đổi trực tiếp như trên tương đối nguy hiểm, dễ gây đổ vỡ hệ thống.
- Hệ thống file QNX4: Đây là hệ thống file được sử dụng trong hệ điều hành QNX 4.
- Hệ thống file ROMFS: Đây là các hệ thống file chỉ đọc (read only) được sử dụng chủ yếu cho việc khởi tạo đĩa ảo (ramdisk) trong quá trình khởi động đĩa cài đặt.
- Hệ thống file SMB: SMB (Server Mesage Block) là một giao thức của Windows dùng để chia sẻ file giữa các hệ điều hành Windows 95/98, Windows NT và OS/2 Lan Manager. Với sự hỗ trợ SMB, hệ điều hành Linux có thể chia sẻ cũng như truy cập các

file nằm trên các phân vùng của một máy chạy các hệ điều hành kể trên. Nói tóm lại, SMB cũng là một dạng hỗ trợ hệ thống file mạng giúp hệ thống có thể chia sẻ với các hệ thống sử dụng chung giao thức SMB.

- Hệ thống file UMSDOS: Hệ thống file UMSDOS (Unix-like MSDOS) là hệ thống file được mở rộng từ hệ thống file MSDOS theo định hướng Unix. Hệ thống file này có một số ưu điểm so với MSDOS như là hỗ trợ tên file dài, hỗ trợ việc phân quyền, hỗ trợ các liên kết (link), hỗ trợ các file đặc biệt (device, pipe ...) và... Hệ thống file này có thể được sử dụng làm phân vùng gốc của hệ thống Linux.
- Hệ thống file VFAT: VFAT chính là hệ thống file mở rộng của hệ thống FAT. Hệ thống file này được sử dụng trong các hệ điều hành Windows 95/98.

Như vậy, ngoài khả năng hỗ trợ nhiều loại thiết bị, Linux còn có khả năng hỗ trợ nhiều kiểu hệ thống file. Bằng cách hỗ trợ nhiều kiểu hệ thống file, Linux có thể truy cập và xử lý các file của nhiều hệ điều hành khác nhau. Mặc dù có khả năng truy cập nhiều hệ thống file khác nhau, hệ thống file của Linux vẫn phải đảm bảo cung cấp cho người dùng một giao diện nhất quán đối với các file, bảo vệ các file trên các hệ thống khác nhau, tối ưu các thao tác truy cập vào thiết bị... Để thực hiện được điều này, Linux sử dụng một hệ thống file đặc biệt gọi là hệ thống file ảo VFS (Virtual File System).

Hệ thống file ảo VFS được thiết kế để cung cấp một giao diện thống nhất về các file ược lưu trữ trên các thiết bị. Hình 3.3 mô tả mối quan hệ giữa VFS với các hệ thống file thực và các thiết bị lưu trữ.



Hình 3.3. Hệ thống file ảo VFS

VFS có trách nhiệm cung cấp cho chương trình người dùng một giao diện nhất quán về hệ thống file thông qua các lệnh gọi hệ thống (system call). Mỗi khi có một yêu cầu truy cập file, VFS sẽ dựa vào các hệ thống file thực để tìm kiếm file yêu cầu trên các thiết bị vật lý. Với mỗi file tìm được, nó thực hiện thao tác mở file đó và cho tương ứng file với một cấu trúc dữ liệu gọi là i-node. VFS cung cấp rất nhiều lệnh gọi để thao tác với hệ thống file nhưng chủ yếu thuộc vào các loại sau:

- Các thao tác liên quan tới hệ thống file.
- Các thao tác liên quan tới i-node.
- Các thao tác với file đang mở.
- Các thao tác với vùng đệm dữ liệu.

3.1.5. Liên kết tượng trưng (lệnh ln)

Trong Linux có hai kiểu liên kết đó là liên kết tượng trưng (liên kết mềm) và liên kết cứng.

"Liên kết cứng" là một cách gọi khác đối với một file đang tồn tại (không có sự phân biệt giữa file gốc và file liên kết). Theo cách nói kỹ thuật, chúng cùng chia sẻ một inode và inode này chứa đựng tất cả các thông tin về file. Không thể tạo một liên kết cứng tới một thư mục.

"Liên kết tượng trưng" là một kiểu file đặc biệt, trong đó, một file liên kết thực sự tham chiếu theo tên đến một file khác. Có thể hiểu kiểu file này như là một con trỏ chỉ dẫn tới một file hoặc một thư mục, và được sử dụng để thay thế cho file hoặc thư mục được trỏ tới. Hầu hết các thao tác (như mở, đọc, ghi ...) được thực hiện trên các file liên kết, sau đó, nhân hệ thống sẽ tự động "tham chiếu" và thực hiện trên file đích của liên kết. Tuy nhiên, có một số các thao tác như xóa file, file liên kết sẽ bị xóa bỏ chứ không phải file đích của nó.

Để tạo một liên kết tượng trưng, hãy sử dụng lệnh **ln** với cú pháp như sau:

```
ln [tùy-chọn] <đích> [tên-nối]
```

Lệnh này sẽ tạo một liên kết đến thư mục/file **đích** với tên file liên kết là **tên-nối**. Nếu **tên-nối** không có, một liên kết với tên file liên kết giống như tên file **đích** sẽ được tạo ra trong thư mục hiện thời.

Các tuỳ chọn của lệnh **ln**:

- **-b, --backup[=CONTROL]** : tạo liên kết quay trở lại cho mỗi file đích đang tồn tại.
- **-f, --force** : xóa bỏ các file đích đang tồn tại.
- **-d, -F, --directory** : tạo liên kết cứng đến các thư mục (tùy chọn này chỉ dành cho người dùng có quyền quản trị hệ thống). Một số phiên bản không có tùy chọn này.
- **-n, --no-dereference** : một file bình thường được xem là đích liên kết từ một thư mục.
- **-i, interactive** : vẫn tạo liên kết dù file đích đã bị xóa bỏ.
- **-s, --symbolic** : tạo các liên kết tượng trưng.
- **--target-directory=<tên-thư-mục>** : xác định thư mục tên-thư-mục là thư mục có chứa các liên kết.
- **-v, --verbose** : hiển thị tên các file trước khi tạo liên kết.
- **--help** : hiển thị trang trợ giúp và thoát.

Ví dụ, muốn tạo liên kết đến file **/usr/doc/g77/DOC** với tên file liên kết là **g77manual.txt**, thì gõ lệnh như sau:

```
# ln -s /usr/doc/g77/DOC g77manual.txt
```

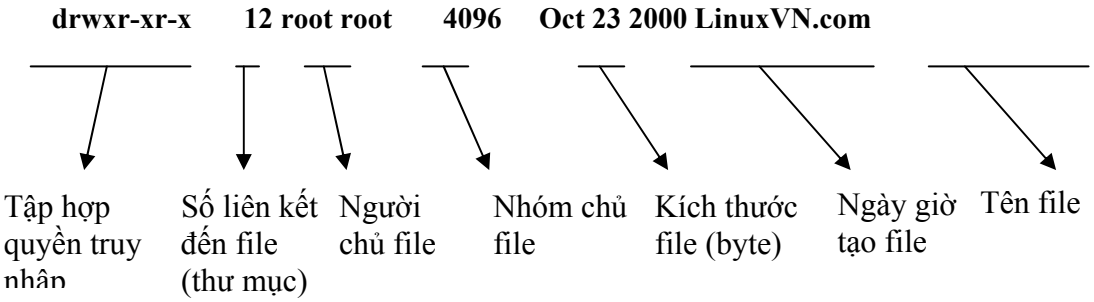
Khi chạy chương trình ***mc***, các file liên kết có tên bắt đầu bởi dấu "ừ", và khi vệt sáng di chuyển đến file liên kết thì tên file được liên kết đến sẽ hiển thị ở bên dưới.

3.2 Quyền truy nhập thư mục và file

3.2.1 Quyền truy nhập

Mỗi file và thư mục trong Linux đều có một chủ sở hữu và một nhóm sở hữu, cũng như một tập hợp các quyền truy nhập. Cho phép thay đổi các quyền truy nhập và quyền sở hữu file và thư mục nhằm cung cấp truy nhập nhiều hơn hay ít hơn.

Thông tin về một file có dạng sau (được hiện ra theo lệnh hiện danh sách file **ls -l**):



Trong đó, dãy 10 ký tự đầu tiên mô tả kiểu file và quyền truy nhập đối với tập tin đó.

Theo mặc định, người dùng tạo một file chính là người chủ (sở hữu) của file đó và là người có quyền sở hữu nó. Người chủ của file có đặc quyền thay đổi quyền truy nhập hay quyền sở hữu đối với file đó. Tất nhiên, một khi đã chuyển quyền sở hữu của mình cho người dùng khác thì người chủ cũ không được phép chuyển quyền sở hữu và quyền truy nhập được nữa.

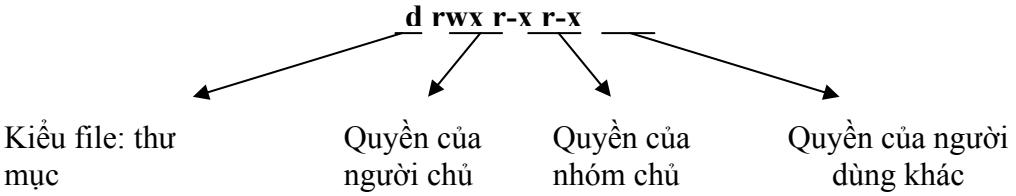
Tập hợp một chuỗi có 10 ký tự đã giới thiệu trên đây được chia ra làm 4 phần: kiểu file, các quyền truy nhập đến file của chủ sở hữu, của nhóm sở hữu và người dùng khác.

Có một số kiểu file trong Linux. Ký tự đầu tiên trong tập hợp 10 ký tự mô tả kiểu file và quyền truy nhập sẽ cho biết file thuộc kiểu nào (chữ cái đó được gọi là chữ cái biểu diễn). Bảng dưới đây sẽ liệt kê các kiểu file trong Linux:

Chữ cái biểu diễn	Kiểu file
d	Thư mục (directory)
b	File kiểu khối (block-type special file)
c	File kiểu ký tự (character-type special file)
l	Liên kết tượng trưng (symbolic link)
p	File đường ống (pipe)
s	Socket
-	File bình thường (regular file)

Chín ký tự tiếp theo trong chuỗi là quyền truy nhập được chia ra làm 3 nhóm tương ứng với quyền truy nhập của người sử hữu, nhóm sở hữu và người dùng khác.

Ví dụ, 10 ký tự đầu tiên trong dòng ví dụ ngay trước đây sẽ được phân tích thành:



Để hiểu được chính xác quyền truy nhập có ý nghĩa như thế nào đối với hệ thống máy tính, phải nhớ rằng **Linux xem mọi thứ đều là file**. Nếu cài đặt một ứng dụng, nó cũng sẽ được xem như mọi chương trình khác, trừ một điều: hệ thống nhận biết rằng một ứng dụng là một chương trình khả thi, tức là nó có thể chạy được. Một bức thư gửi cho mẹ là một dạng file văn bản bình thường, nhưng nếu thông báo cho hệ thống biết đó là một chương trình khả thi, hệ thống sẽ cố để chạy chương trình (và tất nhiên là lỗi).

Có ba loại quyền truy nhập chính đối với thư mục/file, đó là: đọc (read - r), ghi (write - w) và thực hiện (execute - x). Quyền đọc cho phép người dùng có thể xem nội dung của file với rất nhiều chương trình khác nhau, nhưng họ sẽ không thể thay đổi, sửa chữa hoặc xóa bất kỳ thông tin nào trong đó. Tuy nhiên, họ có thể sao chép file đó thành file của họ và sửa chữa file bản sao.

Quyền ghi là quyền truy nhập tiếp theo. Người sử dụng với quyền ghi khi truy nhập vào file có thể thêm thông tin vào file. Nếu có quyền ghi và quyền đọc đối với một file, có thể soạn thảo lại file đó - quyền đọc cho phép xem nội dung, và quyền ghi cho phép thay đổi nội dung file. Nếu chỉ có quyền ghi, sẽ thêm được thông tin vào file, nhưng lại không thể xem được nội dung của file.

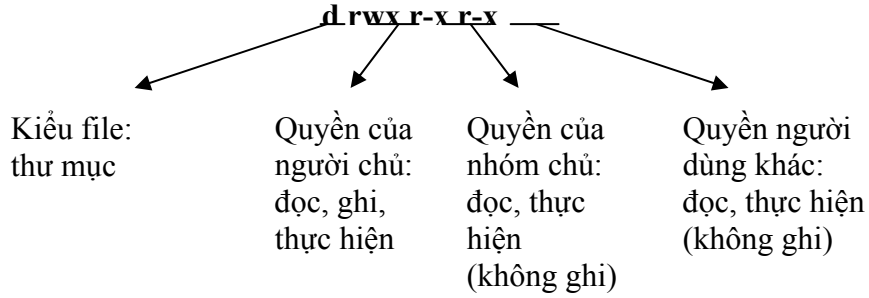
Loại quyền truy nhập thứ ba là quyền thực hiện, quyền này cho phép người dùng có thể chạy được file, nếu đó là một chương trình khả thi. Quyền thực hiện độc lập với các quyền truy nhập khác, vì thế hoàn toàn có thể có một chương trình với quyền đọc và quyền thực hiện, nhưng không có quyền ghi. Cũng có trường hợp một chương trình chỉ có quyền thực hiện, có nghĩa là người dùng có thể chạy ứng dụng, nhưng họ không thể xem được cách nó làm việc hay sao chép nó.

Bảng dưới đây giới thiệu cách ký hiệu của các quyền truy nhập:

Quyền truy nhập	Ý nghĩa
---	Không cho phép một quyền truy nhập nào
r--	Chỉ được quyền đọc
r-x	Quyền đọc và thực hiện (cho chương trình và shell script)
rw-	Quyền đọc và ghi
rwx	Cho phép tất cả các quyền truy nhập (cho chương trình)

Tuy nhiên, đối với thư mục thì chỉ có ba loại ký hiệu của các quyền truy nhập là: **---**, **r-x** và **rwx**, vì nội dung của thư mục là danh sách của các file và các thư mục con có bên trong thư mục đó. Quyền đọc một thư mục là được xem nội dung của thư mục đó và quyền thực hiện đối với một thư mục là quyền tìm được file và thư mục con có trong thư mục.

Như vậy, với ví dụ đang được xem xét, chúng ta nhận được đây là một thư mục và quyền truy nhập nó được giải thích như sau:



☞ ***Giải thích:***

Sự hạn chế trường hợp về quyền truy nhập thư mục được giải thích theo các lập luận như sau:

- Hãy hình dung, giả sử chỉ có quyền đọc trên thư mục, khi đó sẽ xem được có những file hay thư mục nào trong thư mục nhưng lại không thể xem cụ thể nội dung của một file hay thư mục có trên thư mục đó vì không tìm được nó.
- Hoặc giả sử có quyền thực hiện - quyền này sẽ cho phép tìm được file có trên thư mục - nhưng lại không có quyền đọc đối với một thư mục, vậy thì làm thế nào để biết được trong thư mục có những file nào.

3.2.2. Các lệnh cơ bản

a. Thay đổi quyền sở hữu file với lệnh *chown*

Để thay đổi quyền sở hữu đối với một file, hãy sử dụng lệnh **chown** với cú pháp như sau:

```
chown [tïy-chän] [chñ][.nhãm] <file ...>
```

Lệnh này cho phép thay chủ sở hữu **file**. Nếu chỉ có tham số về ***chủ***, thì người dùng ***chủ*** sẽ có quyền sở hữu file và nhóm sở hữu không thay đổi. Nếu theo sau tên người **chủ** là dấu "**.**" và tên của một ***nhóm*** thì nhóm đó sẽ nhóm sở hữu file. Nếu chỉ có dấu "**.**" và ***nhóm*** mà không có tên người chủ thì chỉ có quyền sở hữu nhóm của file thay đổi, lúc này, lệnh **chown** có tác dụng giống như lệnh **chgrp** (lệnh **chgrp** được trình bày dưới đây).
Các tùy chọn của lệnh **chown**:

- **-c, --changes :** hiển thị dòng thông báo chỉ với các file mà lệnh làm thay đổi sở hữu (số thông báo hiện ra có thể ít hơn trường hợp **-v, -verbosr**) **.**
- **-f, --silent, --quiet :** bỏ qua hầu hết các thông báo lỗi.
- **-R, --recursive :** thực hiện đổi quyền sở hữu đối với thư mục và file theo đệ quy**.**
- **-v, --verbose** : hiển thị dòng thông báo với mọi file liên quan mà chown tác động tới (có hoặc không thay đổi sở hữu).
- **--help :** đưa ra trang trợ giúp và thoát.

Ví dụ, thư mục **LinuxVN.com** có thông tin về các quyền truy nhập như sau:

drwxr-xr-x 12 thu root 4096 Oct 23 2000 LinuxVN.com

Người sở hữu hiện tại thư mục **LinuxVN.com** là người dùng ***thu***. Để người dùng ***lan*** là chủ sở hữu thư mục trên, hãy gõ lệnh:

```
# chown lan LinuxVN.com
```

Khi đó, nếu dùng lệnh **ls** thì thông tin về thư mục **LinuxVN.com** sẽ có dạng:

```
drwxr-xr-x 12 lan root 4096 Oct 23 2000 LinuxVN.com
```

với người sở hữu thư mục bây giờ là người dùng ***lan***.

Khi chuyển quyền sở hữu file cho một người khác, người chủ cũ mất quyền sở hữu file đó.

b. Thay đổi quyền sở hữu nhóm với lệnh *chgrp*

Các file (và người dùng) còn thuộc vào các nhóm, đây là phương thức truy nhập file thuận tiện cho nhiều người dùng nhưng không phải tất cả người dùng trên hệ thống. Khi đăng nhập, mặc định sẽ là thành viên của một nhóm được thiết lập khi siêu người dùng root tạo tài khoản người dùng. Cho phép một người dùng thuộc nhiều nhóm khác nhau, nhưng mỗi lần đăng nhập chỉ là thành viên của một nhóm.

Để thay đổi quyền sở hữu nhóm đối với một hoặc nhiều file, hãy sử dụng lệnh **chgrp** với cú pháp như sau:

```
chgrp [tùy-chọn] {nhóm|--reference=nhómR} <file...>
```

Lệnh này cho phép thay thuộc tính nhóm sở hữu của file theo tên nhóm được chỉ ra trực tiếp theo tham số **nhóm** hoặc gián tiếp qua thuộc tính nhóm của file có tên là **nhómR**.

Các tùy chọn của lệnh là (một số tương tự như ở lệnh **chown**):

- **-c, --changes :** hiển thị dòng thông báo chỉ với các file mà lệnh làm thay đổi sở hữu (số thông báo hiện ra có thể ít hơn trường hợp **-v, -verbosr**) .
- **-f, --silent, --quiet :** bỏ qua hầu hết các thông báo lỗi.
- **-R, --recursive :** thực hiện đổi quyền sở hữu đối với thư mục và file theo đệ quy.
- **-v, --verbose :** hiển thị dòng thông báo với mọi file liên quan mà chgrp tác động tới (có hoặc không thay đổi sở hữu).
- **--help :** hiển thị trang trợ giúp và thoát

Tham số --reference=nhómR cho thấy cách gián tiếp thay nhóm chủ của file theo nhóm chủ của một file khác (tên là nhómR) là cách thức được ưa chuộng hơn. Tham số này là xung khắc với tham số nhóm của lệnh.

c. Thay đổi quyền truy cập file với lệnh *chmod*

Cú pháp lệnh **chmod** có ba dạng:

```
chmod [tùy-chọn] <mod [,mod]...> <file...>
chmod [tùy-chọn] <mod-hệ-8> <file...>
chmod [tùy-chọn] --reference=nhómR <file...>
```

Lệnh **chmod** cho phép xác lập quyền truy nhập theo kiểu (**mode)** trên **file**. Dạng đầu tiên là dạng xác lập tương đối, dạng thứ hai là dạng xác lập tuyệt đối và dạng cuối cùng là dạng gián tiếp chỉ dẫn theo quyền truy nhập của file **nhómR**.

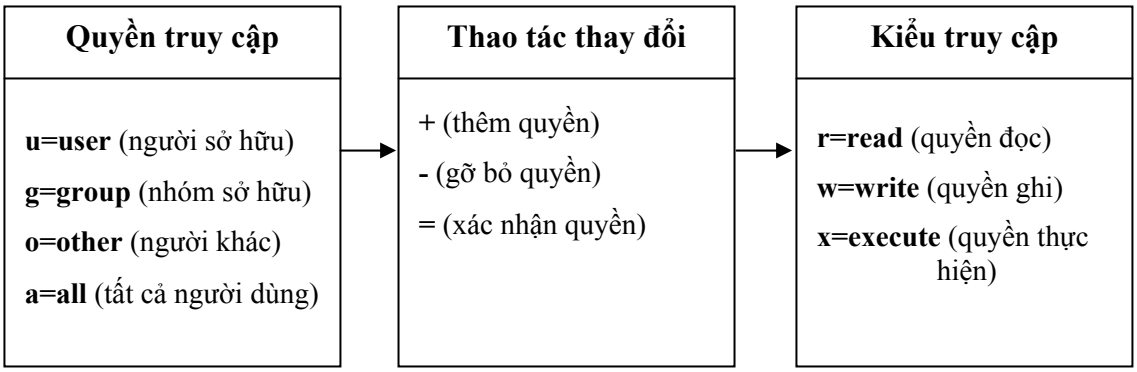
Các tùy chọn của lệnh **chmod** được liệt kê như dưới đây và có ý nghĩa tương tự các tuỳ chọn tương ứng của các lệnh **chown**, **chgrp**:

- `-c, --changes`
- `-f, --silent, --quiet`
- `-v, --verbose`
- `-R, --recursive`
- `--help`
- và tham số **--reference=RFILE** cũng ý nghĩa gián tiếp như trong lệnh **chgrp**.

Giải thích về hai cách xác lập quyền truy nhập file trong lệnh **chmod** như sau: xác lập tuyệt đối (dùng hệ thống mã số viết theo hệ cơ số 8 biểu diễn cho các quyền truy nhập) và xác lập tương đối (dùng các chữ cái để biểu diễn quyền truy nhập).

Cách xác lập tương đối

Cách xác lập tương đối là dễ nhớ theo ý nghĩa của nội dung các **mod** và chỉ những thay đổi thực sự mới được biểu diễn trong lệnh. Ba hộp sau đây sẽ giải thích các chữ cái biểu diễn **mod** theo cách xác lập tương đối.



Có thể kết hợp các mục từ hộp thứ nhất và hộp thứ ba với một mục từ hộp thứ hai để tạo ra một **mod**.

Ví dụ, nếu muốn thêm quyền ghi đối với file ***test*** cho tất cả người dùng trong nhóm sở hữu, hãy chọn **g** cho nhóm sở hữu, **+** cho thêm quyền truy nhập, và **w** cho quyền ghi. Lúc đó lệnh **chmod** sẽ có dạng sau:

```
chmod g+w test
```

Cách xác lập tương đối trong lệnh **chmod** gần giống như một menu có nhiều mục chọn khác nhau, cho phép kết hợp để có được sự lựa chọn theo yêu cầu.

Nếu quyết định gỡ bỏ quyền đọc và thực hiện trên file ***test*** cho những người không cùng nhóm, hãy chọn **o** cho người dùng khác, **-** để gỡ bỏ quyền truy nhập, và r,x cho quyền đọc và thực hiện. Lệnh **chmod** sẽ là:

```
chmod o-rx test
```

Cách xác lập tuyệt đối

Đối với người dùng hiểu sơ bộ về biểu diễn số trong hệ cơ số 8 thì cách xác lập tuyệt đối lại được ưa chuộng hơn.

Phần 3.2.1. cho biết biểu diễn quyền truy nhập file thông qua dãy gồm 9 vị trí dưới dạng **rwxrwxrwx**, trong đó từng cụm 3 vị trí theo thứ tự tương ứng với: chủ sở hữu, nhóm sở hữu và người dùng khác. Như vậy thuộc tính quyền truy nhập của một file có thể biểu

diễn thành 9 bít nhị phân trong đó bít có giá trị 1 thì quyền đó được xác định, ngược lại thì quyền đó bị tháo bỏ. Như vậy, chủ sở hữu tương ứng với 3 bít đầu tiên, nhóm sỡ hữu tương ứng với 3 bít giữa, người dùng khác tương ứng với 3 bít cuối. Mỗi cụm 3 bít như vậy cho một chữ số hệ 8 (nhận giá trị từ 0 đến 7) và thuộc tính quyền truy nhập tương ứng với 3 chữ số hệ 8.

Ví dụ, cặp 3 số hệ 8 là 755 tương ứng với dòng 9 bít 111101101 với 111 cho chủ sở hữu, 101 cho nhóm sở hữu, 101 cho người dùng khác. Ví dụ lệnh:

```
chmod 753 memo1
```

đặt thuộc tính quyền truy nhập đối với file memo1 là **rwxr-xr-x**. Để dễ xác lập 3 chữ số hệ 8 áp dụng cách tính như sau:

Quyền	*Chữ số hệ 8*	*Quyền*	*Chữ số hệ 8*
Chỉ đọc	4	Chỉ đọc và ghi	6
Chỉ ghi	2	Chỉ đọc và thực hiện	5
Chỉ thực hiện	1	Chỉ ghi và thực hiện	3
Không có quyền nào	0	Đọc, ghi và thực hiện	7

d. đăng nhập vào một nhóm người dùng mới với lệnh *newgrp*

Linux cho phép một người dùng có thể là thành viên của một hoặc nhiều nhóm người dùng khác nhau, trong đó có một nhóm được gọi là nhóm khởi động. Điều này được đảm bảo khi thực hiện lệnh **adduser** hoặc **usersdd**. Tuy nhiên, tại một thời điểm, một người dùng thuộc vào chỉ một nhóm. Khi một người dùng đăng nhập, hệ thống ngầm định người dùng đó là thành viên của nhóm khởi động, và có quyền truy nhập đối với những file thuộc quyền sở hữu của nhóm khởi động đó. Nếu muốn sử dụng quyền sở hữu theo các nhóm khác đối với những file thì người dùng phải chuyển đổi thành thành viên của một nhóm những nhóm đã được gắn với người dùng. Lệnh **newgr** cho phép người dùng chuyển sang nhóm người dùng khác đã gắn với mình với cú pháp:

```
newgrp [nhóm]
```

trong đó **nhóm** là một tên nhóm người dùng tồn tại trong hệ thống.

Ví dụ, một người dùng là thành viên của hai nhóm ***user*** và ***installer***, với ***user*** là nhóm khởi động. Khi đăng nhập, người dùng đó có tư cách là thành viên của nhóm ***user***. Khi mong muốn sử dụng một số các chương trình thuộc quyền sở hữu của nhóm ***installer***, người dùng cần gõ lệnh sau:

```
# newgrp installer
```

Nếu người dùng nói trên cố chuyển vào một nhóm mà người dùng đó không là thành viên, chẳng hạn dùng lệnh:

```
# newgrp hot2
```

thì Linux sẽ đưa ra một khuyến cáo thân thiện như sau:

```
newgrp: Sorry
```

3.3 Thao tác với thư mục

Như đã được giới thiệu (mục 3.1.1.), Linux tổ chức hệ thống file theo cách sử dụng các thư mục. Mục này bắt đầu bằng việc giới thiệu một số thư mục chính và tác dụng của chúng trong hệ thống Linux. Sau đó một số lệnh thao tác với thư mục cơ bản nhất được trình bày.

3.3.1 Một số thư mục đặc biệt

*** Thư mục gốc /**

Đây là thư mục gốc chứa đựng tất cả các thư mục con có trong hệ thống.

*** Thư mục */root***

Thư mục **/root** có thể được coi là "thư mục riêng" của siêu người dùng. Thư mục này được sử dụng để lưu trữ các file tạm thời, nhân Linux và ảnh khởi động, các file nhị phân quan trọng (những file được sử dụng đến trước khi Linux có thể gắn kết đến phân vùng **/user**), các file đăng nhập quan trọng, bộ đệm in cho việc in ấn, hay vùng lưu tạm cho việc nhận và gửi email. Nó cũng được sử dụng cho các vùng trống tạm thời khi thực hiện các thao tác quan trọng, ví dụ như khi xây dựng (**build**) một gói RPM từ các file RPM nguồn.

*** Thư mục */bin***

Trong Linux, chương trình được coi là khả thi nếu nó có thể thực hiện được. Khi một chương trình được biên dịch, nó sẽ có dạng là file nhị phân. Như vậy, chương trình ứng dụng trong Linux là một file nhị phân khả thi.

Chính vì lẽ đó, những nhà phát triển Linux đã quyết định phải tổ chức một thư mục "binaries" để lưu trữ các chương trình khả thi có trên hệ thống, đó chính là thư mục **/bin**.

Ban đầu, thư mục **/bin** (`bin` là viết tắt của từ **binary**) là nơi lưu trữ các file nhị phân khả thi. Nhưng theo thời gian, ngày càng có nhiều hơn các file khả thi có trong Linux, do đó, có thêm các thư mục như **/sbin**, **/usr/bin** được sử dụng để lưu trữ các file đó.

*** Thư mục */dev***

Một phần không thể thiếu trong bất kỳ máy tính nào đó là các trình điều khiển thiết bị. Không có chúng, sẽ không thể có được bất kỳ thông tin nào trên màn hình của (các thông tin có được do trình điều khiển thiết bị hiển thị đưa ra). Cũng không thể nhập được thông tin (những thông tin do trình điều khiển thiết bị bàn phím đọc và chuyển tới hệ thống), và cũng không thể sử dụng đĩa mềm của (được quản lý bởi trình điều khiển đĩa mềm).

Tất cả các trình điều khiển thiết bị đều được lưu trữ trong thư mục **/dev**.

*** Thư mục */etc***

Quản trị hệ thống trong Linux không phải là đơn giản, chẳng hạn như việc quản lý tài khoản người dùng, vấn đề bảo mật, trình điều khiển thiết bị, cấu hình phần cứng, v.v.. Để giảm bớt độ phức tạp, thư mục **/etc** đã được thiết kế để lưu trữ tất cả các thông tin hay các file cấu hình hệ thống.

*** Thư mục */lib***

Linux có một trung tâm lưu trữ các thư viện hàm và thủ tục, đó là thư mục **/lib**.

*** Thư mục *<u>/lost+found</u>***

Một file được khôi phục sau khi có bất kỳ một vấn đề hoặc gặp một lỗi về ghi đĩa trên hệ thống đều được lưu vào thư mục này.

*** Thư mục *<u>/mnt</u>***

Thư mục **/mnt** là nơi để kết nối các thiết bị (ví dụ đĩa cứng, đĩa mềm...) vào hệ thống file chính nhờ lệnh **mount**. Thông thường các thư mục con của **/mnt** chính là gốc của các hệ thống file được kết nối: **/mnt/floppy**: đĩa mềm, **/mnt/hda1**: vùng đầu tiên của đĩa cứng thứ nhất (**hda**), **/mnt/hdb3**: vùng thứ ba của đĩa cứng thứ 2 (**hdb**) ...

*** Thư mục *<u>/tmp</u>***

Thư mục **/tmp** được rất nhiều chương trình trong Linux sử dụng như một nơi để lưu trữ các file tạm thời.

Ví dụ, nếu đang soạn thảo một file, chương trình sẽ tạo ra một file là bản sao tạm thời (bản nháp) của file đó và lưu vào trong thư mục **/tmp**. Việc soạn thảo thực hiện trực tiếp trên file tạm thời này và sau khi soạn thảo xong, file tạm thời sẽ được ghi đè lên file gốc. Cách thức như vậy bảo đảm sự an toàn đối với file cần soạn thảo.

*** Thư mục *<u>/usr</u>***

Thông thường thì thư mục **/usr** là trung tâm lưu trữ tất cả các lệnh hướng đến người dùng (user-related commands). Tuy nhiên, ngày nay thật khó xác định trong thư mục này có những thứ gì, bởi vì hầu hết các file nhị phân cần cho Linux đều được lưu trữ ở đây, trong đó đáng chú ý là thư mục con **/usr/src** bao gồm các thư mục con chứa các chương trình nguồn của nhân Linux.

*** Thư mục *<u>/home</u>***

Thư mục này chứa các thư mục cá nhân của người dùng: mỗi người dùng tương ứng với một thư mục con ở đây, tên người dùng được lấy làm tên của thư mục con.

*** Thư mục *<u>/var</u>***

Thư mục **/var** được sử dụng để lưu trữ các file chứa các thông tin luôn luôn thay đổi, bao gồm bộ đệm in, vùng lưu tạm thời cho việc nhận và gửi thư (mail), các khóa quá trình, v.v..

*** Thư mục *<u>/boot</u>***

Là thư mục chứa nhân của hệ thống (**Linux-*.*.**), **System.map** (file ánh xạ đến các ***driver*** để nạp các hệ thống file khác), ảnh (image) của hệ thống file dùng cho **initrd** (**ramdisk**), trình điều khiển cho các thiết bị RAID (một thiết bị gồm một mảng các ổ đĩa cứng để tăng tốc độ và độ an toàn khi ghi dữ liệu), các bản sao lưu **`boot record`** của các phân vùng đĩa khác. Thư mục này cho phép khởi động và nạp lại bất kỳ trình điều khiển nào được yêu cầu để đọc các hệ thống file khác.

*** Thư mục *<u>/proc</u>***

Đây là thư mục dành cho nhân (**kernel**) của hệ điều hành và thực tế đây là một hệ thống file độc lập do nhân khởi tạo.

*** Thư mục */misc* và thư mục */opt***

Cho phép lưu trữ mọi đối tượng vào hai thư mục này.

*** Thư mục */sbin***

Thư mục lưu giữ các file hệ thống thường tự động chạy.

3.3.2 Các lệnh cơ bản về thư mục

*** Xác định thư mục hiện thời với lệnh *pwd***

Cú pháp lệnh:

```
pwd
```

Lệnh này cho biết hiện người dùng đang ở trong thư mục nào và hiện ra theo dạng một đường dẫn tuyệt đối.

Ví dụ, gõ lệnh **pwd** tại dấu nhắc lệnh sau khi người dùng **lan** vừa đăng nhập thì màn hình hiển thị như sau:

```
# pwd
/home/lan
#
```

*** Xem thông tin về thư mục với lệnh *ls***

Sử dụng lệnh **ls** và một số các tùy chọn của nó là có thể biết được mọi thông tin về một thư mục. Cú pháp lệnh:

```
ls [tùy-chọn] [file]...
```

Lệnh này đưa ra danh sách các file liên quan đến tham số ***file*** trong lệnh. Trường hợp phổ biến tham số file là một thư mục, tuy nhiên trong một số trường hợp khác, tham số ***file*** xác định nhóm (khi sử dụng các mô tả nhóm *, ? và cặp [và]); nếu không có tham số ***file***, mặc định danh sách các file có trong thư mục hiện thời sẽ được hiển thị.

Các tùy chọn của lệnh:

- **-a :** liệt kê tất cả các file, bao gồm cả file ẩn.
- **-l :** đưa ra thông tin đầy đủ nhất về các file và thư mục.
- **-s :** chỉ ra kích thước của file, tính theo khối (1 khối = 1204 byte).
- **-F :** xác định kiểu file (/ = thư mục, * = chương trình khả thi).
- **-m :** liệt kê các file được ngăn cách nhau bởi dấu ",".
- **-C :** đưa ra danh sách các file và thư mục theo dạng cột (hai thư mục gần nhau được xếp vào một cột).
- **-1** : hiển thị mỗi file hoặc thư mục trên một dòng.
- **-t** : sắp xếp các file và thư mục trong danh sách theo thứ tự về thời gian được sửa đổi gần đây nhất.
- **-x :** đưa ra danh sách các file và thư mục theo dạng cột (hai thư mục gần nhau được xếp trên hai dòng đầu của hai cột kề nhau).
- **-r :** sắp xếp danh sách hiển thị theo thứ tự ngược lại.
- **-R :** liệt kê lần lượt các thư mục và nội dung của các thư mục.

Ví dụ, lệnh

```
# ls -l
```

sẽ hiển thị danh sách đầy đủ nhất về các file và thư mục có trong thư mục hiện thời.

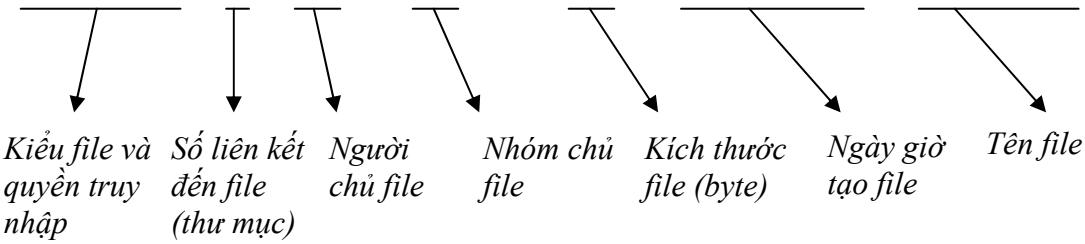
```
total 108
drwxr-xr-x 12 thu root 4096 Oct 23 2000 LinuxVN.com
drwxr-xr-x 2 root root 4096 Oct 31 2000 bin
drwxr-xr-x 2 root root 4096 Dec 11 16:54 boot
drwxr-xr-x 7 root root 36864 Dec 11 16:54 dev
drwxr-xr-x 43 root root 4096 Dec 11 16:55 etc
drwxr-xr-x 5 root root 4096 Dec 11 16:57 home
drwxr-xr-x 4 root root 4096 Oct 31 2000 lib
drwxr-xr-x 2 root root 16384 Oct 31 2000 lost+found
drwxr-xr-x 2 root root 0 Dec 11 16:54 misc
drwxr-xr-x 5 root root 4096 Oct 31 2000 mnt
drwxr-xr-x 2 root root 4096 Aug 23 12:03 opt
dr-xr-xr-x 56 root root 0 Dec 11 11:54 proc
drwxr-x--- 12 root root 4096 Dec 11 16:55 root
drwxr-xr-x 3 root root 4096 Oct 31 2000 sbin
drwxr-xr-x 3 root root 4096 Oct 31 2000 tftpboot
drwxrwxrwx 8 root root 4096 Dec 11 16:58 tmp
drwxr-xr-x 22 root root 4096 Oct 31 2000 usr
drwxr-xr-x 22 root root 4096 Oct 31 2000 var
```

Dòng đầu tiên "**total 108**" cho biết tổng số khối (1024 byte) trên đĩa lưu trữ các file trong danh sách (14*4+36+16=108).

Mỗi dòng tiếp theo trình bày thông tin về mỗi file hay thư mục con. Các thông tin này đã được giới thiệu trước đây.

Chẳng hạn,

drwxr-xr-x 22 root root 4096 Oct 31 2000 var



ý nghĩa của mỗi trường trên đây đã được giải thích trong mục 3.2.1.

Khi gõ lệnh:

```
# ls [is]*
```

cho danh sách các file và thư mục con có tên bắt đầu bằng hoặc chữ cái **i** hoặc chữ cái **s** có trong thư mục hiện thời:

```
info-dir        initlog.conf    inittab         services
shadow          shadow-         shells          smb.conf
```

sysctl.conf syslog.conf

*** Lệnh tạo thư mục *mkdir***

Lệnh **mkdir** tạo một thư mục với cú pháp:

```
mkdir [tùy-chọn] <thư-mục>
```

Lệnh này cho phép tạo một thư mục mới nếu thư mục đó chưa thực sự tồn tại. Để tạo một thư mục, cần đặc tả tên và vị trí của nó trên hệ thống file (vị trí mặc định là thư mục hiện thời). Nếu thư mục đã tồn tại, hệ thống sẽ thông báo cho biết.

Các tùy chọn:

- **-m, --mode=Mod :** thiết lập quyền truy nhập **Mod** như trong lệnh **chmod** nhưng không cho quyền **rwxrwxrwx.**
- **-p, --parents :** tạo các thư mục cần thiết mà không thông báo lỗi khi nó đã tồn tại.
- **--verbose :** hiển thị các thông báo cho mỗi thư mục được tạo.
- **--help :** đưa ra trang trợ giúp và thoát.

Ví dụ, nếu muốn tạo thư mục **test** trong thư mục **home**, hãy gõ lệnh sau:

```
# mkdir /home/test
```

*** Lệnh xóa bỏ thư mục *rmdir***

Như đã biết, lệnh **mkdir** để tạo ra một thư mục mới, và về đối ngẫu thì lệnh **rmdir** được dùng để xóa bỏ một thư mục. Cú pháp lệnh:

```
rmdir [tùy-chọn[ <thư-mục>
```

Có thể xóa bỏ bất kỳ thư mục nào nếu có quyền đó. Lưu ý rằng, thư mục chỉ bị xóa khi nó "rỗng", tức là không tồn tại file hay thư mục con nào trong đó.

Không có cách gì khôi phục lại các thư mục đã bị xóa, vì thế hãy suy nghĩ cẩn thận trước khi quyết định xóa một thư mục.

Các tùy chọn của lệnh:

- **--ignore-fail-on-non-empty :** bỏ qua các lỗi nếu xóa một thư mục không rỗng.
- **-p, --parents :** xóa bỏ một thư mục, sau đó lần lượt xóa bỏ tiếp các thư mục có trên đường dẫn chứa thư mục vừa xóa. Ví dụ, dòng lệnh **rmdir -p /a/b/c** sẽ tương đương với ba dòng lệnh **rmdir /a/b/c, rmdir /a/b, rmdir /a** (với điều kiện các thư mục là rỗng).
- **--verbose :** đưa ra thông báo khi xóa một thư mục.
- **--help :** hiển thị trang trợ giúp và thoát.

Ví dụ:

```
# rmdir -p /test/test1/test2
rmdir: /: No such file or directory
#
```

Dòng lệnh trên sẽ lần lượt xóa ba thư mục **test2**, **test1**, **test** và hiển thị thông báo trên màn hình kết quả của lệnh.

*** Lệnh đổi tên thư mục *mv***

Cú pháp lệnh:

```
mv <tên-cũ> <tên-mới>
```

Lệnh này cho phép đổi tên một thư mục từ **tên-cũ** thành **tên-mới**.

Ví dụ, lệnh

```
# mv LinuxVN.com LinuxVN
```

sẽ đổi tên thư mục **LinuxVN.com** thành **LinuxVN**.

Nếu sử dụng lệnh **mv** để đổi tên một thư mục với một cái tên đã được đặt cho một file thì lệnh sẽ gặp lỗi.

Nếu tên mới trùng với tên một thư mục đang tồn tại thì nội dung của thư mục được đổi tên sẽ ghi đè lên nội dung của thư mục trùng tên.

3.4. Các lệnh làm việc với file

3.4.1 Các kiểu file có trong Linux

Mục 3.1.2. đã trình bày sơ lược về kiểu của các file. Như đã được giới thiệu, có rất nhiều file khác nhau trong Linux, nhưng bao giờ cũng tồn tại một số kiểu file cần thiết cho hệ điều hành và người dùng, dưới đây giới thiệu lại một số các kiểu file cơ bản.

- ***File người dùng (user data file):*** là các file tạo ra do hoạt động của người dùng khi kích hoạt các chương trình ứng dụng tương ứng. Ví dụ như các file thuần văn bản, các file cơ sở dữ liệu hay các file bảng tính.
- ***File hệ thống (system data file):*** là các file lưu trữ thông tin của hệ thống như: cấu hình cho khởi động, tài khoản của người dùng, thông tin thiết bị ... thường được cất trong các tệp dạng văn bản để người dùng có thể can thiệp, sửa đổi theo ý mình.
- ***File thực hiện (executable file):*** là các file chứa mã lệnh hay chỉ thị cho máy tính thực hiện. File thực hiện lưu trữ dưới dạng mã máy mà ta khó có thể tìm hiểu được ý nghĩa của nó, nhưng tồn tại một số công cụ để "hiểu" được các file đó. Khi dùng trình ứng dụng **mc** (Midnight Commander, chương 8), file thực hiện được bắt đầu bởi dấu (*) và thường có màu xanh lục.
- ***Thư mục hay còn gọi là file bao hàm (directory):*** là file bao hàm các file khác và có cấu tạo hoàn toàn tương tự như file thông thường khác nên có thể gọi là file. Trong **mc**, file bao hàm thường có màu trắng và bắt đầu bằng dấu ngã (**~**) hoặc dấu chia (**/**). Ví dụ: **/**, **/home**, **/bin**, **/usr**, **/usr/man**, **/dev** ...
- ***File thiết bị (device file):*** là file mô tả thiết bị, dùng như là định danh để chỉ ra thiết bị cần thao tác. Theo quy ước, file thiết bị được lưu trữ trong thư mục **/dev**. Các file thiết bị hay gặp trong thư mục này là **tty** (teletype - thiết bị truyền thông), **ttyS** (teletype serial - thiết bị truyền thông nối tiếp), **fd0**, **fd1**, ... (floppy disk- thiết bị ổ đĩa mềm), **hda1**, **hda2**, ... **hdb1**, **hdb2**, ... (hardisk - thiết bị ổ cứng theo chuẩn IDE; a, b,... đánh số ổ đĩa vật lý; 1, 2, 3... đánh số ổ logic). Trong **mc**, file thiết bị có màu tím và bắt đầu bằng dấu cộng (+).
- ***File liên kết (linked file):*** là những file chứa tham chiếu đến các file khác trong hệ thống tệp tin của Linux. Tham chiếu này cho phép người dùng tìm nhanh tới file thay vì tới vị trí nguyên thủy của nó. Hơn nữa, người ta có thể gắn vào đó các thông tin phụ trợ làm cho file này có tính năng trội hơn so với tính năng nguyên thủy của nó. Ta thấy loại file này giống như khái niệm ***shortcut*** trong MS-Windows98.

Không giống một số hệ điều hành khác (như MS-DOS chẳng hạn), Linux quản lý thời gian của tệp tin qua các thông số thời gian truy nhập (accesed time), thời gian kiến tạo (created time) và thời gian sửa đổi (modified time).

3.4.2. Các lệnh tạo file

Trong Linux có rất nhiều cách để tạo file, sau đây là các cách hay được dùng.

*** Tạo file với lệnh *touch***

Lệnh **touch** có nhiều chức năng, trong đó một chức năng là giúp tạo file mới trên hệ thống: **touch** rất hữu ích cho việc tổ chức một tập hợp các file mới. Cú pháp lệnh:

```
touch <file>
```

Thực chất lệnh này có tác dụng dùng để cập nhật thời gian truy nhập và sửa chữa lần cuối của một file. Vì lý do này, các file được tạo bằng lệnh **touch** đều được sắp xếp theo thời gian sửa đổi. Nếu sử dụng lệnh **touch** đối với một file chưa tồn tại, chương trình sẽ tạo ra file đó. Sử dụng bất kỳ trình soạn thảo nào để soạn thảo file mới.

Ví dụ, dùng lệnh **touch** để tạo file **newfile:**

```
# touch newfile
```

*** Tạo file bằng cách đổi hướng đầu ra của lệnh (>)**

Cách này rất hữu ích nếu muốn lưu kết quả của một lệnh đã thực hiện.

Để gửi kết quả của một lệnh vào một file, dùng dấu ">" theo nghĩa chuyển hướng lối ra chuẩn. Ví dụ, đưa kết quả của lệnh **ls -l /bin** vào file **/home/thu/lenhls** bằng cách gõ:

```
# ls -l /bin > /home/thu/lenhls
```

Linux tự động tạo nếu file **lenhls** chưa có, trong trường hợp ngược lại, nội dung file cũ sẽ bị thế chỗ bởi kết quả của lệnh.

```
# ls -l /bin >/home/thu/lenhls
```

Nếu muốn bổ sung kết quả vào cuối file thay vì thay thế nội dung file, hãy sử dụng dấu ">>".

Ví dụ, lệnh

```
# ls -l /bin >> /home/thu/lenhls
```

đưa các dòng danh sách file trong thư mục **/bin** vào cuối nội dung của file **/home/thu/lenhls**.

*** Tạo file với lệnh *cat***

Lệnh **cat** tuy đơn giản nhưng rất hữu dụng trong Linux. Chúng ta có thể sử dụng lệnh này để lấy thông tin từ đầu vào (bàn phím...) rồi kết xuất ra file hoặc các nguồn khác (màn hình ...), hay để xem nội dung của một file ... Phần này trình bày tác dụng của lệnh **cat** đối với việc tạo file. Cú pháp lệnh:

```
cat > <file>
```

Theo ngầm định, lệnh này cho phép lấy thông tin đầu vào từ bàn phím rồi xuất ra màn hình. Soạn thảo nội dung của một file bằng lệnh cat tức là đã đổi hướng đầu ra của lệnh từ màn hình vào một file. Người dùng gõ nội dung của file ngay tại dấu nhắc màn hình và gõ **CTRL+d** để kết thúc việc soạn thảo. Nhược điểm của cách tạo file này là nó không cho phép sửa lỗi, ví dụ nếu muốn sửa một lỗi chính tả trên một dòng, chỉ có cách là xóa đến vị trí của lỗi và gõ lại nội dung vừa bị xóa.

Ví dụ. tạo file **newfile** trong thư mục **/home/vd** bằng lệnh **cat.**

```
# cat > /home/vd/newfile
This is a example of cat command
#
```

Sau khi soạn thảo xong, gõ **Enter** và **CTRL+d** để trở về dấu nhắc lệnh, nếu không gõ **Enter** thì phải gõ **CTRL+d** hai lần. Có thể sử dụng luôn lệnh **cat** để xem nội dung của file vừa soạn thảo:

```
# cat /home/vd/newfile
This is a example of cat command
#
```

3.4.3 Các lệnh thao tác trên file

*** Sao chép file với lệnh *cp***

Lệnh **cp** có hai dạng như sau:

```
cp [tùy-chọn] <file-nguồn> ... <file-đích>
cp [tùy-chọn] --target-directory=<thư-mục> <file-nguồn>...
```

Lệnh này cho phép sao **file-nguồn** thành **file-đích** hoặc sao chép từ nhiều file-nguồn vào một thư mục đích (tham số <file-đích> hay <thư-mục>). Dạng thứ hai là một cách viết khác đổi thứ tự hai tham số vị trí.

Các tùy chọn:

- **-a, --archive** : giống như **-dpR** (tổ hợp ba tham số **-d**, **-p**, **-R**, như dưới đây).
- **-b, --backup[=CONTROL]** : tạo file lưu cho mỗi file đích nếu như nó đang tồn tại.
- **-d, --no-dereference** : duy trì các liên kết.
- **-f, --force** : ghi đè file đích đang tồn tại mà không nhắc nhở.
- **-i, --interactive** : có thông báo nhắc nhở trước khi ghi đè.
- **-l, --link** : chỉ tạo liên kết giữa file-đích từ file-nguồn mà không sao chép.
- **-p, --preserve** : duy trì các thuộc tính của file-nguồn sang file-đích.
- **-r** : cho phép sao chép một cách đệ quy file thông thường.
- **-R** : cho phép sao chép một cách đệ quy thư mục.
- **-s, --symbolic-link** : tạo liên kết tượng trưng thay cho việc sao chép các file.
- **-S, --suffix=<hậu-tố>** : bỏ qua các hậu tố thông thường (hoặc được chỉ ra).
- **-u, --update** : chỉ sao chép khi file nguồn mới hơn file đích hoặc khi file đích chưa có.
- **-v, --verbose** : đưa ra thông báo về quá trình sao chép.
- **--help** : hiển thị trang trợ giúp và thoát.

File đích được tạo ra có cùng kích thước và các quyền truy nhập như file nguồn, tuy nhiên file đích có thời gian tạo lập là thời điểm thực hiện lệnh nên các thuộc tính thời gian sẽ khác.

Ví dụ, lệnh

```
# cp /home/ftp/vd /home/test/vd1
```

Nếu ở vị trí đích, mô tả đầy đủ tên file đích thì nội dung file nguồn sẽ được sao chép sang file đích. Trong trường hợp chỉ đưa ra vị trí file đích được đặt trong thư mục nào thì tên của file nguồn sẽ là tên của file đích.

```
# cp /home/ftp/vd /home/test/
```

Trong ví dụ này, tên file đích sẽ là **vd** nghĩa là tạo một file mới **/home/test/vd**.
Nếu sử dụng lệnh này để sao một thư mục, sẽ có một thông báo được đưa ra cho biết nguồn là một thư mục và vì vậy không thể dùng lệnh **cp** để sao chép.

```
# cp . newdir
cp: .: omitting directory
```

Ví dụ về việc lệnh **cp** cho phép sao nhiều file cùng một lúc vào một thư mục.

```
# cp vd vd1 newdir
# pwd
/newdir
# ls -l
total 8
-rw-r--r-- 1 root ftp 15 Nov 14 11:00 vd
-rw-r--r-- 1 root ftp 12 Nov 14 11:00 vd1
```

☞ ***Lưu ý:***

- Đối với nhiều lệnh làm việc với file, khi gõ lệnh có thể sử dụng kí hiệu mô tả nhóm để xác định một nhóm file làm cho tăng hiệu lực của các lệnh đó. Ví dụ, lệnh:

  ```
  # cp * bak
  ```

  thực hiện việc sao chép mọi file có trong thư mục hiện thời sang thư mục con của nó có tên là **bak**.

Dùng lệnh

```
# cp /usr/src/linux-2.2.14/include/linux/*.h bak
```

cho phép sao chép mọi file với tên có hai kí hiệu cuối cùng là "**.h**" sang thư mục con **bak**.

Chính vì lí do nói trên, dù trong nhiều lệnh tuy không nói đến việc sử dụng kí hiệu mô tả nhóm file nhưng chúng ta có thể áp dụng chúng nếu điều đó không trái với suy luận thông thường. Do những tình huống như thế là quá phong phú cho nên không thể giới thiệu hết trong tài liệu. Chúng ta chú ý một giải pháp là mỗi khi sử dụng một lệnh nào đó, nên thử nghiệm cách thức hiệu quả này.

*** Đổi tên file với lệnh *mv***

Cú pháp lệnh đổi tên file:

```
mv <tên-cũ> <tên-mới>
```

Lệnh này cho phép đổi tên file từ **tên cũ** thành **tên mới**.
Ví dụ:

```
# mv vd newfile
```

Lệnh này sẽ đổi tên file **vd** thành **newfile**. Trong trường hợp file **newfile** đã tồn tại, nội dung của file **vd** sẽ ghi đè lên nội dung của file **newfile**.

*** Xóa file với lệnh *rm***

Lệnh **rm** là lệnh rất "nguy hiểm" vì trong Linux không có lệnh khôi phục lại những gì đã xóa, vì thế hãy cẩn trọng khi sử dụng lệnh này. Cú pháp lệnh:

```
rm [tùy-chọn] <file> ...
```

Lệnh **rm** cho phép xóa bỏ một file hoặc nhiều file.

Các tùy chọn:

- **-d, --directory :** loại bỏ liên kết của thư mục, kể cả thư mục không rỗng. Chỉ có siêu người dùng mới được phép dùng tùy chọn này.
- **-f, --force :** bỏ qua các file (xác định qua tham số **file**) không tồn tại mà không cần nhắc nhở.
- **-i, --interactive :** nhắc nhở trước khi xóa bỏ một file.
- **-r, -R, --recursive :** xóa bỏ nội dung của thư mục một cách đệ quy.
- **-v, --verbose :** đưa ra các thông báo về quá trình xóa file.
- **--help :** hiển thị trang trợ giúp và thoát.

Lệnh **rm** cho phép xóa nhiều file cùng một lúc bằng cách chỉ ra tên của các file cần xóa trong dòng lệnh (hoặc dùng kí kiệu mô tả nhóm).

Ví dụ, dùng lệnh **ls** để xem danh sách các file trong thư mục hiện thời:

```
# ls
  ld-Linux.so.1              libnss_dns-2.1.3.so
  ld-Linux.so.1.9.5          libnss_dns.so.1
  ld-Linux.so.2              libnss_dns.so.2
  ld.so                      libnss_files-2.1.3.so
  ld.so.1.9.5                libnss_files.so.1
  libBrokenLocale-2.1.3.so   libnss_files.so.2
  libBrokenLocale.so.1       libnss_hesiod-1.3.so
  libNoVersion-2.1.3.so      telex.o
  vd2.txt
```

Sử dụng lệnh xóa file **vd2.txt** sau đây:

```
# rm vd2.txt telex.o
```

và sau đó dùng lệnh **ls** để xem lại danh sách file:

```
# ls
  ld-Linux.so.1              Libnss_dns-2.1.3.so
  ld-Linux.so.1.9.5          Libnss_dns.so.1
  ld-Linux.so.2              Libnss_dns.so.2
  ld.so                      Libnss_files-2.1.3.so
  ld.so.1.9.5                Libnss_files.so.1
  libBrokenLocale-2.1.3.so   Libnss_files.so.2
  libBrokenLocale.so.1       Libnss_hesiod-1.3.so
  libNoVersion-2.1.3.so      telex.o
```

Dùng lệnh

```
# rm bak/*.h
```

xóa mọi file với tên có hai kí hiệu cuối cùng là "**.h**" trong thư mục con **bak.**

*** Lệnh đếm từ và dòng trong file *wc***

Linux có lệnh **wc** dùng để đếm số ký tự, số từ, hay số dòng trong một file. Cú pháp lệnh:

```
wc [tùy-chọn] [file]...
```

Lệnh hiện ra số lượng dòng, số lượng từ, số lượng ký tự có trong mỗi file, và một dòng tính tổng nếu có nhiều hơn một file được chỉ ra. Nếu không có tùy chọn nào thì mặc định đưa ra cả số dòng, số từ và số ký tự. Ngầm định khi không có tên file trong lệnh thì sẽ đọc và đếm trên thiết bị vào chuẩn.

Các tuỳ chọn:

- **-c, --byte, --chars :** đưa ra số ký tự trong file.
- **-l, --lines :** đưa ra số dòng trong file.
- **-L, --max-line-length :** đưa ra chiều dài của dòng dài nhất trong file.
- **-w, --words :** đưa ra số từ trong file.
- **--help :** hiển thị trang trợ giúp và thoát.

Ví dụ, sau khi gõ lệnh:

```
# wc /home/lan/mau/mau1
```

xuất hiện dòng thông báo:

```
11 64 293 /home/lan/mau/mau1
```

Dòng thông báo trên cho biết file **mau1** có 293 ký tự, số 64 từ và có 11 dòng.

Ví dụ sau khi gõ lệnh:

```
# wc
```

người dùng gõ tiếp các dòng như sau:

```
This is a example of wc command without
[namefile]
```

sau đó người dùng gõ cặp phím **Ctrl-d** để kết thúc thì thấy dòng thông báo hiện ra:

```
2       9       49
```

Khi gõ lệnh **wc** mà không có một tham số nào, mặc định sẽ soạn thảo trực tiếp nội dung trên thiết bị vào chuẩn. Dùng **CTRL+d** để kết thúc việc soạn thảo, kết quả sẽ hiển thị lên màn hình như ví dụ trên.

```
# wc /home/lan/vd/vdcalj /home/lan/vd/vdwc
8       41      192     /home/lan/vd/vdcalj
24      209     1473    /home/lan/vd/vdwc
32      250     1665    total
```

Lệnh trên đếm số ký tự, số từ, số dòng trên mỗi file được chỉ ra, và dòng cuối cùng hiển thị tổng số dòng, số từ, số ký tự đếm được.

Bằng cách kết hợp lệnh **wc** với một số lệnh khác, có thể có nhiều cách để biết được những thông tin cần thiết. Chẳng hạn:

- kết hợp với lệnh **ls** để xác định số file có trong một thư mục:

```
# ls | wc -l
```

```
37
```

dòng lệnh trên cho biết trong thư mục chủ của có 36 file (do dòng đầu tiên kết quả thông báo của lệnh **ls** không xác định một file).

- kết hợp với lệnh **cat** để biết số tài khoản cá nhân có trên máy của người dùng:

```
# cat /etc/passwd | wc -l
 324
```

*** <u>Lệnh loại bỏ những dòng không quan trọng *uniq*</u>**

Trong một số trường hợp khi xem nội dung một file, chúng ta thấy có một số các thông tin bị trùng lặp, ví dụ các dòng trống hoặc các dòng chứa nội dung giống nhau. Để đồng thời làm gọn và thu nhỏ kích thước của file, có thể sử dụng lệnh **uniq** để liệt kê ra nội dung file sau khi đã loại bỏ các dòng trùng lặp. Cú pháp lệnh:

```
uniq [tùy-chọn] [input] [output]
```

Lệnh **uniq** sẽ loại bỏ các dòng trùng lặp kề nhau từ **input** (thiết bị vào chuẩn) và chỉ giữ lại một dòng duy nhất trong số các dòng trùng lặp rồi đưa ra **output** (thiết bị ra chuẩn). Các tuỳ chọn:

- `-c, --count :` đếm và hiển thị số lần xuất hiện của các dòng trong file.
- `-d :` hiển thị lên màn hình dòng bị trùng lặp.
- `-u :` hiển thị nội dung file sau khi xóa bỏ toàn bộ các dòng bị trùng lặp không giữ lại một dòng nào.
- `-i :` hiển thị nội dung file sau khi xóa bỏ các dòng trùng lặp và chỉ giữ lại duy nhất một dòng có nội dung bị trùng lặp.
- `-D :` hiển thị tất cả các dòng trùng lặp trên màn hình.

Nếu sử dụng lệnh **uniq** trên một file không có các dòng trùng lặp thì lệnh không có tác dụng.

Ví dụ, người dùng sử dụng lệnh **cat** để xem nội dung file **vduniq**

```
# cat vduniq
Gnome có hai phương pháp để thoát ra ngoài.
Gnome có hai phương pháp để thoát ra ngoài.
Để thoát bằng cách sử dụng menu chính, hãy mở
menu chính, chọn mục Logout ở đáy menu.
Chọn YES/ NO để kết thúc phiên làm việc với Gnome.
Chọn YES/ NO để kết thúc phiên làm việc với Gnome.
Nếu muốn thoát bằng cách sử dụng nút Logout trên Panel,
trước hết phải thêm nút này vào Panel.
Chọn YES/ NO để kết thúc phiên làm việc với Gnome.
```

Trong file **vduniq** có hai dòng bị trùng lặp và kề nhau là dòng thứ 1 và 2.

```
Gnome có hai phương pháp để thoát ra ngoài.
Gnome có hai phương pháp để thoát ra ngoài.
```

và dòng thứ 5 và 6

```
Chọn YES/ NO để kết thúc phiên làm việc với Gnome.
```

```
Chọn YES/ NO để kết thúc phiên làm việc với Gnome.
```

Dùng lệnh **uniq** để loại bỏ dòng trùng lặp:

```
# uniq vduniq
Gnome có hai phương pháp để thoát ra ngoài.
Để thoát bằng cách sử dụng menu chính, hãy mở
menu chính, chọn mục Logout ở đáy menu.
Chọn YES/ NO để kết thúc phiên làm việc với Gnome.
Nếu muốn thoát bằng cách sử dụng nút Logout trên Panel,
trước hết phải thêm nút này vào Panel.
Chọn YES/ NO để kết thúc phiên làm việc với Gnome.
```

Dòng cuối cùng trong file **vduniq** có nội dung trùng với dòng thứ 5, nhưng sau lệnh **uniq**, nó không bị xóa vì không kề với dòng có nội dung trùng lặp.

*** Sắp xếp nội dung file với lệnh *sort***

sort là lệnh đọc các thông tin và sắp xếp chúng theo thứ tự trong bảng chữ cái hoặc theo thứ tự được quy định theo các tùy chọn của lệnh. Cú pháp lệnh:

```
sort [tùy-chọn] [file] ...
```

Hiển thị nội dung sau khi sắp xếp của một hoặc nhiều file ra thiết bị ra chuẩn là tác dụng của lệnh **sort**. Ngầm định sắp xếp theo thứ tự từ điển của các dòng có trong các file (từng chữ cái theo bảng chữ hệ thống (chẳng hạn ASCII) và kể từ vị trí đầu tiên trong các dòng).

Các tùy chọn:

- **+<số1> [-<số2>]** : Hai giá trị **số1** và **số2** xác định "khóa" sắp xếp của các dòng, thực chất lấy xâu con từ vị trí **số1** tới vị trí **số2** của các dòng để so sánh lấy thứ tự sắp xếp các dòng. Nếu **số2** không có thì coi là hết các dòng; nếu **số2** nhỏ hơn **số1** thì bỏ qua lựa chọn này. Chú ý, nếu có **số2** thì phải cách **số1** ít nhất một dấu cách.
- **-b** : bỏ qua các dấu cách đứng trước trong phạm vi sắp xếp.
- **-c** : kiểm tra nếu file đã sắp xếp thì thôi không sắp xếp nữa.
- **-d** : xem như chỉ có các ký tự [a-zA-Z0-9] trong khóa sắp xếp, các dòng có các kí tự đặc biệt (dấu cách, ? ...) được đưa lên đầu.
- **-f** : sắp xếp không phân biệt chữ hoa chữ thường.
- **-n** : sắp xếp theo kích thước của file.
- **-r** : chuyển đổi thứ tự sắp xếp hiện thời.

Ví dụ, muốn sắp xếp file **vdsort**

```
# cat vdsort
trước hết phải thêm nút này vào Panel.
21434
bạn xác nhận là có thực sự muốn thoát hay không.
menu chính, chọn mục Logout ở đáy menu.
Bạn có thể sử dụng mục Logout từ menu chính
Gnome có hai phương pháp để thoát ra ngoài.
```

```
hoặc nút Logout trên Panel chính để thoát ra ngoài.
Khi đó một hộp thoại Logout sẽ xuất hiện yêu cầu
57879
Lựa chọn YES hoặc NO để kết thúc phiên làm việc với
Gnome.
Nó không cung cấp chức năng hoạt động nào khác ngoài chức
năng này.
Nó không cung cấp chức năng hoạt động nào khác ngoài chức
năng này.
Nếu muốn thoát bằng cách sử dụng nút Logout trên Panel,
# sort -f vdsort
21434
57879
Bạn có thể sử dụng mục Logout từ menu chính
bạn xác nhận là có thực sự muốn thoát hay không.
Gnome có hai phương pháp để thoát ra ngoài.
hoặc nút Logout trên Panel chính để thoát ra ngoài.
Khi đó một hộp thoại Logout sẽ xuất hiện yêu cầu
Lựa chọn YES hoặc NO để kết thúc phiên làm việc với
Gnome.
menu chính, chọn mục Logout ở đáy menu.
Nếu muốn thoát bằng cách sử dụng nút Logout trên Panel,
Nó không cung cấp chức năng hoạt động nào khác ngoài chức
năng này.
Nó không cung cấp chức năng hoạt động nào khác ngoài chức
năng này.
trước hết phải thêm nút này vào Panel.
```

Có thể kết hợp lệnh **sort** với các lệnh khác, ví dụ:

```
# ls -s | sort -n
total 127
1       Archive/
1       infoWorld/
13      keylime.pie
46      drop.text.hqx
64      bitnet.mailing-lists.Z
```

Lệnh trên cho thứ tự sắp xếp của các file theo kích thước trong thư mục hiện thời.

3.4.4 Các lệnh thao tác theo nội dung file

*** Sử dụng lệnh *file* để xác định kiểu file**

Cú pháp lệnh **file**:

file [tùy-chọn] [-f file] [-m <file-ảnh>...] <file>...

Lệnh **file** cho phép xác định và in ra kiểu thông tin chứa trong **file**. Lệnh **file** sẽ lần lượt kiểm tra từ kiểu file hệ thống, kiểu file **magic** (ví dụ file mô tả thiết bị) rồi đến kiểu file văn bản thông thường. Nếu file được kiểm tra thỏa mãn một trong ba kiểu file trên thì kiểu file sẽ được in ra theo các dạng cơ bản sau:

- **text**: dạng file văn bản thông thường, chỉ chứa các mã ký tự ASCII.
- **executable**: dạng file nhị phân khả thi.
- **data**: thường là dạng file chứa mã nhị phân và không thể in ra được.

Một số tuỳ chọn sau đây:

- **-b :** cho phép chỉ đưa ra kiểu file mà không đưa kèm theo tên file.
- **-f tên-file** : cho phép hiển thị kiểu của các file có tên trùng với nội dung trên mỗi dòng trong file **tên-file**. Để kiểm tra trên thiết bị vào chuẩn, sử dụng dấu "**-**".
- **-z :** xem kiểu của file nén.

Ví dụ:

```
# file file.c file /dev/hda
file.c: C program text
file:   ELF   32-bit   LSB   executable,   Intel   80386,version
1,dynamically linked, not stripped
/dev/hda:   block special
```

Lệnh trên cho xem kiểu của hai file **file.c**, **file** và thư mục **/dev/hda**.
Nhớ rằng kết quả của lệnh **file** không phải lúc nào cũng chính xác tuyệt đối.

*** Xem nội dung file với lệnh *cat***

Ở đoạn trước, chúng ta đã có dịp làm quen với lệnh **cat** thông qua tác dụng tạo file của lệnh. Phần này giới thiệu tác dụng chủ yếu của lệnh **cat**: đó là tác dụng xem nội dung của một file. Cú pháp lệnh:

cat [tùy-chọn] <tên file>

Các tùy chọn:

- **-A, --show-all :** giống như tùy chọn **-vET.**
- **-b, --number-nonblank :** hiển thị thêm số thứ tự trên mỗi dòng (bỏ qua dòng trống).
- **-e :** giống như tùy chọn **-vE.**
- **-E, --show-ends :** hiển thị dấu "$" tại cuối mỗi dòng.
- **-n, --number :** hiển thị số thứ tự của mỗi dòng (kể cả dòng trống).
- **-s :** nếu trong nội dung file có nhiều dòng trống thì sẽ loại bỏ bớt để chỉ hiển thị một dòng trống.
- **-t :** giống như **-vT.**
- **-T, --show-tabs :** hiển thị dấu TAB dưới dạng ^I.
- **-v, --show-nonprinting :** hiển thị các ký tự không in ra được ngoại trừ LFD và TAB.
- **--help :** hiển thị trang trợ giúp và thoát.

Ví dụ:

```
# cat vdcat
```

chúng ta thấy xuất hiện các dòng sau đây:

```
Gnome có hai phương pháp để thoát ra ngoài.
có thể sử dụng mục Logout từ menu chính
hoặc nút Logout trên Panel chính để thoát ra ngoài.
Để thoát bằng cách sử dụng menu chính, hãy mở
menu chính, chọn mục Logout ở đáy menu.

Khi đó một hộp thoại Logout sẽ xuất hiện yêu cầu
xác nhận là có thực sự muốn thoát hay không.

hoặc nút Logout trên Panel chính để thoát ra ngoài.
Để thoát bằng cách sử dụng menu chính, hãy mở
menu chính, chọn mục Logout ở đáy menu.
# cat -bEs vdcat
```

thì nội dung file hiện ra như sau:

```
1 Gnome có hai phương pháp để thoát ra ngoài. $
2 có thể sử dụng mục Logout từ menu chính $
3 hoặc nút Logout trên Panel chính để thoát ra ngoài.$
4 Để thoát bằng cách sử dụng menu chính, hãy mở $
5 menu chính, chọn mục Logout ở đáy menu. $
$
6 Khi đó một hộp thoại Logout sẽ xuất hiện yêu cầu $
7 xác nhận là có thực sự muốn thoát hay không.$
$
8 hoặc nút Logout trên Panel chính để thoát ra ngoài.$
9 Để thoát bằng cách sử dụng menu chính, hãy mở $
10 menu chính, chọn mục Logout ở đáy menu.
```

*** Xem nội dung các file lớn với lệnh *more***

Lệnh **cat** cho phép xem nội dung của một file, nhưng nếu file quá lớn, nội dung file sẽ trôi trên màn hình và chỉ có thể nhìn thấy phần cuối của file. Linux có một lệnh cho phép có thể xem nội dung của một file lớn, đó là lệnh **more**. Cú pháp lệnh:

```
more [-dlfpcsu] [-số] [+/xâumẫu] [+dòng-số] [file ...]
```

Lệnh **more** hiển thị nội dung của file theo từng trang màn hình.

Các lựa chọn:

- **-số** : xác định số dòng nội dung của file được hiển thị (**số**).
- **-d** : trên màn hình sẽ hiển thị các thông báo giúp người dùng cách sử dụng đối với lệnh more, ví như **[Press space to continue, "q" to quit .]**, hay hiển thị [**Press "h" for instructions .]** thay thế cho tiếng chuông cảnh báo khi bấm sai một phím.

- **-l : more** thường xem ^L là một ký tự đặc biệt, nếu không có tùy chọn này, lệnh sẽ dừng tại dòng đầu tiên có chứa ^L và hiển thị % nội dung đã xem được (^L không bị mất), nhấn phím **space** (hoặc **enter**) để tiếp tục. Nếu có tùy chọn **-l**, nội dung của file sẽ được hiển thị như bình thường nhưng ở một khuôn dạng khác, tức là dấu ^L sẽ mất và trước dòng có chứa ^L sẽ có thêm một dòng trống.
- **-p :** không cuộn màn hình, thay vào đó là xóa những gì có trên màn hình và hiển thị tiếp nội dung file.
- **-c :** không cuộn màn hình, thay vào đó xóa màn hình và hiển thị nội dung file bắt đầu từ đỉnh màn hình.
- **-s :** xóa bớt các dòng trống liền nhau trong nội dung file chỉ giữ lại một dòng.
- **-u :** bỏ qua dấu gạch chân.
- **+/xâumẫu** : tùy chọn **+/xâumẫu** chỉ ra một chuỗi sẽ được tìm kiếm trước khi hiển thị mỗi file.
- **+dòng-số** : bắt đầu hiển thị từ dòng thứ **dòng-số.**

Ví dụ:

```
# more -d vdmore
total 1424
drwxr-xr-x 6 root root 4096 Oct 31 2000 AfterStep-1.8.0
drwxr-xr-x 2 root root 4096 Oct 31 2000 AnotherLevel
drwxr-xr-x 2 root root 4096 Oct 31 2000 ElectricFence
drwxr-xr-x 2 root root 4096 Oct 31 2000 GXedit-1.23
drwxr-xr-x 3 root root 4096 Oct 31 2000 HTML
drwxr-xr-x 3 root root 4096 Oct 31 2000 ImageMagick
drwxr-xr-x 6 root root 4096 Oct 31 2000 LDP
drwxr-xr-x 3 root root 4096 Oct 31 2000 ORBit-0.5.0
drwxr-xr-x 2 root root 4096 Oct 31 2000 SVGATextMode
drwxr-xr-x 2 root root 4096 Oct 31 2000 SysVinit-2.78
drwxr-xr-x 2 root root 4096 Oct 31 2000 WindowMaker
--More--(9%) [ Press space to continue, "q" to quit .]
```

Đối với lệnh **more**, có thể sử dụng một số các phím tắt để thực hiện một số các thao tác đơn giản trong khi đang thực hiện lệnh. Bảng dưới đây liệt kê các phím tắt đó:

Phím tắt	Chức năng
[Space]	Nhấn phím space để hiển thị màn hình tiếp theo
n	Hiển thị n dòng tiếp theo
[Enter]	Hiển thị dòng tiếp theo
h	Hiển thị danh sách các phím tắt
d hoặc CTRL+D	Cuộn màn hình (mặc định là 11 dòng)
q hoặc CTRL+Q	Thoát khỏi lệnh **more**
s	Bỏ qua n dòng (mặc định là 1)
f	Bỏ qua k màn hình tiếp theo (mặc định là 1)
b hoặc CTRL+B	Trở lại k màn hình trước (mặc định là 1)

=	Hiển thị số dòng hiện thời
:n	xem k file tiếp theo
:p	Trở lại k file trước
v	Chạy chương trình soạn thảo vi tại dòng hiện thời
CTRL+L	Vẽ lại màn hình
:f	Hiển thị tên file hiện thời và số dòng
.	Lặp lại lệnh trước

*** Thêm số thứ tự của các dòng trong file với lệnh *nl***

Như đã biết lệnh **cat** với tham số **-n** sẽ đánh số thứ tự của các dòng trong file, tuy nhiên Linux còn cho phép dùng lệnh **nl** để thực hiện công việc như vậy. Cú pháp lệnh:

```
nl [tùy-chọn] <file>
```

Lệnh này sẽ đưa nội dung file ra thiết bị ra chuẩn, với số thứ tự của dòng được thêm vào. Nếu không có **file** (**tên file)**, hoặc khi **file** là dấu "**-**", thì đọc nội dung từ thiết bị vào chuẩn.

Các tuỳ chọn:

- **-b, --body-numbering=STYLE :** sử dụng kiểu STYLE cho việc đánh thứ tự các dòng trong nội dung file. Có các kiểu STYLE sau:
  - **a** : đánh số tất cả các dòng kể cả dòng trống;
  - **t** : chỉ đánh số các dòng không trống;
  - **n** : không đánh số dòng.
- **-d, --section-delimiter=CC :** sử dụng CC để đánh số trang logic (CC là hai ký tự xác định phạm vi cho việc phân trang logic).
- **-f, --footer-numbering=STYLE :** sử dụng kiểu STYLE để đánh số các dòng trong nội dung file (một câu có thể có hai dòng ...).
- **-h, --header-numbering=STYLE :** sử dụng kiểu STYLE để đánh số các dòng trong nội dung file.
- **-i, --page-increment=số** : đánh số thứ tự của dòng theo cấp số cộng có công sai là **số**.
- **-l, --join-blank-lines=số** :nhóm **số** dòng trống vào thành một dòng trống.
- **-n, --number-format=khuôn** : chèn số dòng theo **khuôn** (khuôn: **ln** - căn trái, không có số 0 ở đầu; **rn** - căn phải, không có số 0 ở đầu; **rz** - căn phải và có số 0 ở đầu)
- **-p, --no-renumber :** không thiết lập lại số dòng tại mỗi trang logic.
- **-s, --number-separator=xâu** : thêm chuỗi **xâu** vào sau số thứ tự của dòng.
- **-v, --first-page=số** : số dòng đầu tiên trên mỗi trang logic.
- **-w, --number-width=số** : hiển thị số thứ tự của dòng trên cột thứ **số**.
- **--help** : hiển thị trang trợ giúp và thoát.

Ví dụ:

```
  # nl --body-numbering=a --number-format=rz vdnl
000001 1) New configuration mode
000002
000003
```

```
000004 1-1) Directories
000005
000006 Now, everything goes to ~/GNUstep/Library/AfterStep or
000007 /usr/local/share/afterstep !
000008
000009 You can use your old .steprc config file with afterstep -
f myoldsteprc,
000010 however, this isn't recommended at all.
000011
000012 New versions of asapps will also put their config. file
here in a near
000013 future, like modules currently do.
000014
```

Lệnh trong ví dụ trên cho thêm số thứ tự của các câu trong file **vdnl** theo dạng: đánh số thứ tự tất cả các dòng, kể cả dòng trống, các số thứ tự được căn phải và có số 0 ở đầu (lưu ý rằng có dòng trong file được hiện ra thành hai dòng trên giấy).

* Xem qua nội dung file với lệnh **head**

Các đoạn trước cho biết cách thức xem nội dung của một file nhờ lệnh **cat** hay **more**. Trong Linux cũng có các lệnh khác cho nhiều cách thức để xem nội dung của một file. Trước hết, chúng ta hãy làm quen với lệnh **head**. Cú pháp lệnh

```
head [tùy-chọn] [file]...
```

Lệnh này mặc định sẽ đưa ra màn hình 10 dòng đầu tiên của mỗi file. Nếu có nhiều hơn một file, thì lần lượt tên của file và 10 dòng nội dung đầu tiên sẽ được hiển thị. Nếu không có tham số **file**, hoặc **file** là dấu "**-**", thì ngầm định sẽ đọc từ thiết bị vào chuẩn.

Các tuỳ chọn:

- **-c, --bytes=cỡ** : hiển thị **cỡ** (số nguyên) ký tự đầu tiên trong nội dung file (**cỡ** có thể nhận giá trị là **b** cho 512, **k** cho 1K, **m** cho 1 Meg)
- **-n, --lines=n** : hiển thị **n** (số nguyên) dòng thay cho 10 dòng ngầm định.
- **-q, --quiet, --silent** : không đưa ra tên file ở dòng đầu.
- **-v, --verbose** : luôn đưa ra tên file ở dòng đầu.
- **--help** : hiển thị trang trợ giúp và thoát.

Ví dụ:

```
# head -6 vdhead1 vdhead2
==> vdhead1 <==
1) New configuration mode
1-1) Directories

Now, everything goes to Ú/GNUstep/Library/AfterStep or

==> vdhead2 <==
```

```
1.7.164 patch 3

$HOME/GNUstep/Library/AfterStep/start/Desktop/Theme/.include
changed from shell script call to perl script call
```

Lệnh này cho xem qua 6 dòng đầu tiên trong nội dung hai file **vdhead1** và **vdhead2**.

*** Xem qua nội dung file với lệnh *tail***

Lệnh thứ hai cho phép xem qua nội dung của file là lệnh **tail** với cú pháp:

```
tail [tùy-chọn] [file]...
```

Lệnh **tail** ngầm định đưa ra màn hình 10 dòng cuối trong nội dung của các file. Nếu có nhiều hơn một file, thì lần lượt tên của file và 10 dòng cuối sẽ được hiển thị. Nếu không có tham số **file,** hoặc **file** là dấu "**-**" thì ngầm định sẽ đọc từ thiết bị vào chuẩn.
Các tùy chọn:

- **--retry** : cố gắng mở một file khó truy nhập khi bắt đầu thực hiện lệnh **tail.**
- **-c, --bytes=n** : hiển thị **n** (số) ký tự sau cùng.
- **-f, --follow[={name | descritptor}]** : sau khi hiện nội dung file sẽ hiện thông tin về file: **-f**, **--follow**, và **--follow=descriptor** là như nhau.
- **-n, --lines=n** : hiển thị **n** (số) dòng cuối cùng của file thay cho 10 dòng ngầm định.
- **--max-unchanged-stats=n** : hiển thị tài liệu về file (ngầm định **n** là 5).
- **--max-consecutive-size-changes=n** : hiển thị tài liệu về file (ngầm định **n** là 200).
- **--pid=PID** : kết hợp với tùy chọn **-f**, chấm dứt sau khi quá trình có chỉ số = **PID** lỗi.
- **-q, --quiet, --silent** : không đưa ra tên file ở dòng đầu trong nội dung được hiển thị.
- **-s, --sleep-interval=k** : kết hợp với tùy chọn -f, dừng **k** giây giữa các hoạt động.
- **-v, --verbose** : luôn hiển thị tên của file.
- **--help** : hiển thị trang trợ giúp và thoát.

Ví dụ:

```
# tail -2 vdtail1 vdtail2

==> vdtail1 <==
Now, everything goes to ~/GNUstep/Library/AfterStep or
/usr/local/share/afterstep !

==> vdtail2 <==
changed from shell script call to perl script call
```

Lệnh trên cho xem hai dòng cuối của hai file **vdtail1** và **vdtail2**.

*** Tìm sự khác nhau giữa hai file (lệnh *diff*)**

Việc tìm ra sự khác nhau giữa hai file đôi khi là rất cần thiết. Linux có một lệnh có tác dụng như vậy, đó là lệnh **diff** với cú pháp:

```
diff [tuỳ-chọn] <file1> <file2>
```

Trong trường hợp đơn giản, lệnh **diff** sẽ so sánh nội dung của hai file. Nếu **file1** là một thư mục còn **file2** là một file bình thường, **diff** sẽ so sánh file có tên trùng với **file2** trong thư mục **file1** với **file2**.

Nếu cả **file1** và **file2** đều là thư mục, **diff** sẽ thực hiện sự so sánh lần lượt các file trong cả hai thư mục theo thứ tự từ a-z (sự so sánh này sẽ không đệ qui nếu tuỳ chọn **-r** hoặc **--recursive** không được đưa ra). Tất nhiên so sánh giữa hai thư mục không thể chính xác như khi so sánh hai file.

Các tuỳ chọn:

- **-a**: xem tất cả các file ở dạng văn bản và so sánh theo từng dòng.
- **-b**: bỏ qua sự thay đổi về số lượng của ký tự trống.
- **-B**: bỏ qua mọi sự thay đổi mà chỉ chèn hoặc xoá các dòng trống.
- **--brief**: chỉ thông báo khi có sự khác nhau mà không đưa ra chi tiết nội dung khác nhau.
- **-d**: tìm ra sự khác biệt nhỏ (tuỳ chọn này có thể làm chậm tốc độ làm việc của lệnh **diff**).
- **--exclude-from=file**: khi so sánh thư mục, bỏ qua các file và các thư mục con có tên phù hợp với mẫu có trong **file.**
- **-i**: so sánh không biệt chữ hoa chữ thường.
- **-r**: thực hiện so sánh đệ qui trên thư mục.
- **-s**: thông báo khi hai file là giống nhau.
- **-y**: hiển thị hai file cạnh nhau để dễ phân biệt sự khác nhau.

3.4.5 Các lệnh tìm file

*** Tìm theo nội dung file bằng lệnh *grep***

Lệnh **grep** cũng như lệnh **ls** là hai lệnh rất quan trọng trong Linux. Lệnh này có hai tác dụng cơ bản như sau:

- tác dụng thứ nhất là lọc đầu ra của một lệnh khác với cú pháp là

  ```
  <lệnh> | grep <mẫu lọc>
  ```

- tác dụng thứ hai, và cũng là tác dụng cơ bản được giới thiệu trong phần này, là tìm dòng chứa mẫu đã định trong file được chỉ ra.

Cú pháp lệnh **grep**:

```
grep [tùy-chọn] <mẫu-lọc> [file]
```

Lệnh **grep** hiển thị tất cả các dòng có chứa **mẫu-lọc** trong **file** được chỉ ra (hoặc từ thiết bị vào chuẩn nếu không có **file** hoặc **file** có dạng là dấu "**-**")

Các tùy chọn:

- **-G, --basic-regexp** : xem mẫu lọc như một biểu thức thông thường. Điều này là ngầm định.
- **-E, --extended-regexp** : xem mẫu lọc như một biểu thức mở rộng.
- **-F, --fixed-strings** : xem mẫu như một danh sách các xâu cố định, được phân ra bởi các dòng mới. Ngoài lệnh grep còn có hai lệnh là egrep và fgrep. egrep tương tự như lệnh grep -E, fgrep tương tự với lệnh grep -F .

Lệnh **grep** còn có các tùy chọn sau:

- **-A NUM, --after-context=NUM** : đưa ra NUM dòng nội dung tiếp theo sau dòng có chứa mẫu.

- **-B NUM, --before-context=NUM** : đưa ra NUM dòng nội dung trước dòng có chứa mẫu.
- **-C [NUM], --context[=NUM]** : hiển thị NUM dòng (mặc định là 2 dòng) nội dung.
- **-NUM** : giống **--context=NUM** đưa ra các dòng nội dung trước và sau dòng có chứa mẫu. Tuy nhiên, **grep** sẽ không đưa ra dòng nào nhiều hơn một lần.
- **-b, --byte-offset** : hiển thị địa chỉ tương đối trong file đầu vào trước mỗi dòng được đưa ra
- **-c, --count** : đếm số dòng tương ứng chứa mẫu trong file đầu vào thay cho việc hiển thị các dòng chứa mẫu.
- **-d ACTION, --directories=ACTION** : nếu đầu vào là một thư mục, sử dụng ACTION để xử lý nó. Mặc định, ACTION là **read**, tức là sẽ đọc nội dung thư mục như một file thông thường. Nếu ACTION là **skip**, thư mục sẽ bị bỏ qua. Nếu ACTION là **recurse**, **grep** sẽ đọc nội dung của tất cả các file bên trong thư mục (đệ quy); tùy chọn này tương đương với tùy chọn **-r**.
- **-f file, --file=file** : lấy các mẫu từ **file**, một mẫu trên một dòng. File trống chứa đựng các mẫu rỗng, và các dòng đưa ra cũng là các dòng trống.
- **-H, --with-file** : đưa ra tên file trên mỗi dòng chứa mẫu tương ứng.
- **-h, --no-filename** : không hiển thị tên file kèm theo dòng chứa mẫu trong trường hợp tìm nhiều file.
- **-i** : hiển thị các dòng chứa mẫu không phân biệt chữ hoa chữ thường.
- **-l** : đưa ra tên các file trùng với mẫu lọc.
- **-n, --line-number** : thêm số thứ tự của dòng chứa mẫu trong file.
- **-r, --recursive** : đọc tất cả các file có trong thư mục (đệ quy).
- **-s, --no-messages** : bỏ qua các thông báo lỗi file không đọc được hoặc không tồn tại.
- **-v, --invert-match** : hiển thị các dòng không chứa mẫu.
- **-w, --word-regexp** : chỉ hiển thị những dòng có chứa mẫu lọc là một từ trọn vẹn.
- **-x, --line-regexp** : chỉ hiển thị những dòng mà nội dung trùng hoàn toàn với mẫu lọc.

Ví dụ, người dùng gõ lệnh **cat** để xem nội dung file **text**:

```
# cat -n text
```

thì hiện ra nội dung file đó như sau:

```
1$ file file.c file /dev/hda
2file.c: C program text
3file:ELF 32-bit LSB executable, Intel 80386, version 1,
4dynamically linked, not stripped
5/dev/hda: block special
6
7$ file -s /dev/hdaỳ,1,2,3,4,5,6,7,8,9,10Ỷ
8/dev/hda: x86 boot sector
9/dev/hda1: Linux/i386 ext2 filesystem
10     /dev/hda2: x86 boot sector
11     /dev/hda3: x86 boot sector, extended partition table
12     /dev/hda4: Linux/i386 ext2 filesystem
13     /dev/hda5: Linux/i386 swap file
14     /dev/hda6: Linux/i386 swap file
```

```
15      /dev/hda7: Linux/i386 swap file
16      /dev/hda8: Linux/i386 swap file
17      thutest
18      toithutest
```

Sau đó, dùng lệnh **grep** để lọc các dòng có cụm **filesystem**

```
# grep -n filesystem text
9: /dev/hda1: Linux/i386 ext2 filesystem
12: /dev/hda4: Linux/i386 ext2 filesystem
```

Cũng có thể sử dụng các ký hiệu biểu diễn thông thường (regular - expression) trong mẫu lọc để đưa ra được nhiều cách tìm kiếm file khác nhau.

Bảng dưới đây liệt kê một số ký hiệu hay dùng:

Ký hiệu	Ý nghĩa
C	- thay thế cho ký tự c
\c	- hiển thị c như là một ký tự bình thường nếu c là một ký tự điều khiển
^	- bắt đầu một dòng
$	- kết thúc dòng
.	- thay cho một ký tự đơn
[xy]	- chọn một ký tự trong tập hợp các ký tự được đưa ra
[^xy]	- chọn một ký tự không thuộc tập hợp các ký tự được đưa ra
c*	- thay cho một mẫu có hoặc không chứa ký tự c

```
# grep - H thutest text
text: thutest
text: toithutest
# grep -H "^thutest" text
text: thutest
```

Ngoài các tùy chọn khác nhau, lệnh **grep** còn có hai dạng nữa trên Linux. Hai dạng đó là **egrep** - sử dụng với các mẫu lọc phức tạp, và **fgrep** - sử dụng để tìm nhiều mẫu lọc cùng một lúc.

- Thỉnh thoảng một biểu thức đơn giản không thể xác định được đối tượng cần tìm, ví dụ, như đang cần tìm các dòng có một hoặc hai mẫu lọc. Những lúc đó, lệnh **egrep** tỏ ra rất có ích. **egrep - expression grep -** có rất nhiều các ký hiệu biểu diễn mạnh hơn **grep**. Dưới đây là các ký hiệu hay dùng:

Ký hiệu	Ý nghĩa
c	- thay thế cho ký tự c
\c	- hiển thị c như là một ký tự bình thường nếu c là một ký tự điều khiển
^	- bắt đầu một dòng
$	- kết thúc dòng
.	- thay cho một ký tự đơn
[xy]	- chọn một ký tự trong tập hợp các ký tự được đưa ra
[^xy]	- chọn một ký tự không thuộc tập hợp các ký tự được đưa ra
c*	- thay cho một mẫu có hoặc không chứa ký tự c
c+	- thay cho một mẫu có chứa một hoặc nhiều hơn ký tự c
c?	- thay cho một mẫu không có hoặc chỉ có chứa duy nhất một ký tự c
a | b	- hoặc là a hoặc là b
(a)	- a một biểu thức

Ví dụ, giả sử bây giờ muốn tìm các dòng có chứa một hoặc nhiều hơn ký tự ***b*** trên file **passwk** với lệnh **egrep**.

```
# egrep 'b+' /etc/passwd | head
```

cho ra các dòng kết quả sau:

```
root : x : 0 : 0 : root : /root : /bin/bash
bin : x : 1 : 1 : bin : /bin :
daemon : x : 2 : 2 : daemon : /sbin :
sync : x : 5 : 0 : sync : /sbin : /bin/sync
shutdown : x : 6 : 0 : shutdown : /sbin : /sbin/shutdown
halt : x : 7 : 0 : halt : /sbin : /sbin/halt
gopher : x : 13 : 30 : gopher : /usr/lib/gopher-data :
nobody : x : 99 : 99 : Nobody : / :
xfs : x : 43 : 43 : X Font Server : /etc/X11/fs : /bin/false
named : x : 25 : 25 : Named : /var/named : /bin/false
```

Khi gõ lệnh:

```
# egrep '([^a-zA-Z] | :wi)' /etc/printcap | head
```

thì nhận được thông báo kết quả:

```
aglw:U\
      :wi=AG 23 : wk=multiple Apple LaserWrite IINT:
 aglw1:\
      :wi=AG 23 : wk=Apple LaserWrite IINT:
 aglw2:\
      :wi=AG 23 : wk=Apple LaserWrite IINT:
 aglw3:\
      :wi=AG 23 : wk=Apple LaserWrite IINT:
```

Lệnh trên cho phép tìm các dòng được bắt đầu bởi (^) một chữ cái không phân biệt chữ hoa chữ thường ([a-zA-Z]) hoặc (|) dòng có chứa mẫu **:wi**.

Bất kỳ lúc nào muốn tìm các dòng có chứa nhiều hơn một mẫu lọc, **egrep** là lệnh tốt nhất để sử dụng.

❖ Có những lúc cần phải tìm nhiều mẫu lọc trong một lúc. Ví dụ, có một file chứa rất nhiều mẫu lọc và muốn sử dụng một lệnh trong Linux để tìm các dòng có chứa các mẫu đó. Lệnh **fgrep** sẽ làm được điều này.

Ví dụ, file **thu** có nội dung như sau:

```
# cat thu
/dev/hda4: Linux/i386 ext2 filesystem
/dev/hda5: Linux/i386 swap file
/dev/hda8: Linux/i386 swap file
/dev/hda9: empty
/dev/hda10: empty
thutest
toithutest
```

và file **mauloc** có nội dung là:

```
# cat mauloc
empty
test
```

Bây giờ muốn sử dụng nội dung file **mauloc** làm mẫu lọc để tìm các câu trong file **thu**, hãy gõ lệnh:

```
# fgrep -i -f mauloc thu
/dev/hda9: empty
/dev/hda10: empty
thutest
toithutest
```

*** Tìm theo các đặc tính của file với lệnh *find***

Các đoạn trên đây đã giới thiệu cách thức tìm file theo nội dung với các lệnh **grep**, **egrep** và **fgrep**. Linux còn cho phép người dùng sử dụng một cách thức khác đầy năng lực, đó là sử dụng lệnh **find**, lệnh tìm file theo các thuộc tính của file. Lệnh này có một sự khác biệt so với các lệnh khác, đó là các tùy chọn của lệnh là một từ chứ không phải một ký tự. Điều kiện cần đối với lệnh này là chỉ ra được điểm bắt đầu của việc tìm kiếm trong hệ thống file và những quy tắc cần tuân theo của việc tìm kiếm. Cú pháp của lệnh **find**:

```
find [đường-dẫn] [biểu-thức]
```

Lệnh **find** thực hiện việc tìm kiếm file trên cây thư mục theo **biểu thức** được đưa ra. Mặc định **đường dẫn** là thư mục hiện thời, **biểu thức** là **-print**.

Biểu thức có thể có những dạng sau:

- Các toán tử:

(EXPR); ! EXPR hoặc -not EXPR; EXPR1 -a EXPR2 hoặc EXPR1 -and EXPR2; EXPR1 -o EXPR2 hoặc EXPR1 -or EXPR2; và EXPR1, EXPR2

- Các tùy chọn lệnh: tất cả các tùy chọn này luôn trả về giá trị **true** và được đặt ở đầu biểu thức

- **-daystart** : đo thời gian (-amin, -atime, -cmin, -ctime, -mmin, -mtime).
- **-depth** : thực hiện tìm kiếm từ nội dung bên trong thư mục trước (mặc định việc tìm kiếm được thực hiện bắt đầu tại gốc cây thư mục có chứa file cần tìm).
- **-follow** : (tùy chọn này chỉ áp dụng cho thư mục) nếu có tùy chọn này thì các liên kết tượng trưng có trong một thư mục liên kết sẽ được chỉ ra.
- **-help**, **--help** : hiển thị kết quả của lệnh find và thoát.
  - các **test**
- **-amin n** : tìm file được truy nhập n phút trước.
- **-atime n** : tìm file được truy nhập n*24 giờ trước.
- **-cmin n** : trạng thái của file được thay đổi n phút trước đây.
- **-ctime n** : trạng thái của file được thay đổi n*24 giờ trước đây.
- **-empty** : file rỗng và hoặc là thư mục hoặc là file bình thường.
- **-fstype kiểu** : file thuộc hệ thống file với kiểu.
- **-gid n** : chỉ số nhóm của file là n.
- **-group nhóm** : file thuộc quyền sở hữu của nhóm.
- **-links n** : file có n liên kết.
- **-mmin n** : dữ liệu của file được sửa lần cuối vào n phút trước đây.
- **-mtime n** : dữ liệu của file được sửa vào n*24 giờ trước đây.
- **-name mẫu** : tìm kiếm file có tên là mẫu. Trong tên file có thể chứa cả các ký tự đại diện như dấu "*", "?"...
- **-type kiểu** : tìm các file thuộc kiểu với kiểu nhận các giá trị:
  - b: đặc biệt theo khối
  - c: đặc biệt theo ký tự
  - d: thư mục
  - p: pipe
  - f: file bình thường
  - l: liên kết tượng trưng
  - s: socket
- **-uid n**: chỉ số người sở hữu file là n.
- **-user tên-người**: file được sở hữu bởi người dùng tên-người.

- Các hành động
  - **-exec lệnh :** tùy chọn này cho phép kết hợp lệnh **find** với một lệnh khác để có được thông tin nhiều hơn về các thư mục có chứa file cần tìm. Tùy chọn **exec** phải sử dụng dấu {} - nó sẽ thay thế cho tên file tương ứng, và dấu '\' tại cuối dòng lệnh, (phải có khoảng trống giữa {} và '\'). Kết thúc lệnh là dấu '**;**'
  - **-fprint file :** hiển thị đầy đủ tên file vào trong **file**. Nếu **file** không tồn tại thì sẽ được tạo ra, nếu đã tồn tại thì sẽ bị thay thế nội dung.
  - **-print :** hiển thị đầy đủ tên file trên thiết bị ra chuẩn.

- **-ls :** hiển thị file hiện thời theo khuôn dạng: liệt kê danh sách đầy đủ kèm cả số thư mục, chỉ số của mỗi file, với kích thước file được tính theo khối (block).

Ví dụ:

```
# find -name 'what*'
./usr/bin/whatis
./usr/bin/whatnow
./usr/doc/AfterStep-1.8.0/TODO/1.0ỵtoỵ1.5/whatsnew
./usr/doc/gnome-libs-devel-1.0.55/devel-docs/gnome-dev-
info/gnome-dev-info/what.html
./usr/doc/gnome-libs-devel-1.0.55/devel-docs/gnome-dev-
info/gnome-dev-info/whatis.html

# find . -type f -exec grep -l -i mapping {} \ ;
./OWL/WordMap/msw-to-txt.c
./.elm/aliases.text
./Mail/mark
./News/usenet.alt
./bin/my.new.cmd: Permission denied
./src/fixit.c
./temp/attach.msg
```

3.5 Nén và sao lưu các file

3.5.1 Sao lưu các file (lệnh tar)

Dữ liệu rất có giá trị, sẽ mất nhiều thời gian và công sức nếu phải tạo lại, thậm chí có lúc cũng không thể nào tạo lại được. Vì vậy, Linux đưa ra các cách thức để người dùng bảo vệ dữ liệu của mình.

Có bốn nguyên nhân cơ bản khiến dữ liệu có thể bị mất: lỗi phần cứng, lỗi phần mềm, lỗi do con người hoặc do thiên tai.

Sao lưu là cách để bảo vệ dữ liệu một cách kinh tế nhất. Bằng cách sao lưu dữ liệu, sẽ không có vấn đề gì xảy ra nếu dữ liệu trên hệ thống bị mất.

Một vấn đề rất quan trọng trong việc sao lưu đó là lựa chọn phương tiện sao lưu. cần phải quan tâm đến giá cả, độ tin cậy, tốc độ, ích lợi cũng như tính khả dụng của các phương tiện sao lưu.

Có rất nhiều các công cụ có thể được sử dụng để sao lưu. Các công cụ truyền thống là **tar**, **cpio** và **dump** (công cụ trong tài liệu này là **tar**). Ngoài ra còn rất nhiều các công cụ khác có thể lựa chọn tùy theo phương tiện sao lưu có trong hệ thống.

Có hai kiểu sao lưu là sao lưu theo kiểu toàn bộ (***full backup***) và sao lưu theo kiểu tăng dần (***incremental backup***). Sao lưu toàn bộ thực hiện việc sao mọi thứ trên hệ thống file, bao gồm tất cả các file. Sao lưu tăng dần chỉ sao lưu những file được thay đổi hoặc được tạo ra kể từ đợt sao lưu cuối cùng.

Việc sao lưu toàn bộ có thể được thực hiện dễ dàng với lệnh **tar** với cú pháp:

```
tar [tùy-chọn] [<file>, ...] [<thư-mục>, ...]
```

Lệnh (chương trình) **tar** được thiết kế để tạo lập một file lưu trữ duy nhất. Với **tar**, có thể kết hợp nhiều file thành một file duy nhất có kích thước lớn hơn, điều này sẽ giúp cho việc di chuyển file hoặc sao lưu băng từ trở nên dễ dàng hơn nhiều.

Lệnh **tar** có các lựa chọn:

- **-c, --create** : tạo file lưu trữ mới.
- **-d, --diff, --compare** : tìm ra sự khác nhau giữa file lưu trữ và file hệ thống được lưu trữ.
- **--delete** : xóa từ file lưu trữ (không sử dụng cho băng từ).
- **-r, --append** : chèn thêm file vào cuối file lưu trữ.
- **-t, --list** : liệt kê nội dung của một file lưu trữ.
- **-u, --update** : chỉ thêm vào file lưu trữ các file mới hơn các file đã có.
- **-x, --extract, --get** : tách các file ra khỏi file lưu trữ.
- **-C, --directory tên-thư-mục** : thay đổi đến thư mục có tên là **tên-thư-mục.**
- **--checkpoint** : đưa ra tên thư mục khi đọc file lưu trữ.
- **-f, --file [HOSTNAME:]file** : tùy chọn này xác định tên file lưu trữ hoặc thiết bị lưu trữ là **file** (nếu không có tùy chọn này, mặc định nơi lưu trữ là **/dev/rmt0**).
- **-h, --dereference :** không hiện các file liên kết mà hiện các file mà chúng trỏ tới.
- **-k, --keep-old-files** : giữ nguyên các file lưu trữ đang tồn tại mà không ghi đè file lưu trữ mới lên chúng.
- **-K, --starting-file file** : bắt đầu tại **file** trong file lưu trữ.
- **-l, --one-file-system** : tạo file lưu trữ trên hệ thống file cục bộ.
- **-M, --multi-volume** : tùy chọn này được sử dụng khi dung lượng của file cần sao lưu là lớn và không chứa hết trong một đơn vị lưu trữ vật lý.
- **-N, --after-date DATE, --newer DATE** : chỉ lưu trữ các file mới hơn các file được lưu trữ trong ngày DATE.
- **--remove-files** : xóa file gốc sau khi đã sao lưu chúng vào trong file lưu trữ.
- **--totals** : đưa ra tổng số byte được tạo bởi tùy chọn **--create**.
- **-v, --verbose** : hiển thị danh sách các file đã được xử lý.

Ví dụ:

```
# tar --create --file /dev/ftape /usr/src
tar: Removing leading / from absolute path names in the archive
#
```

Lệnh trên tạo một file sao lưu của thư mục **/usr/src** trong thư mục **/dev/ftape**, (dòng thông báo ở trên cho biết rằng **tar** sẽ chuyển cả dấu **/** vào trong file sao lưu).

Nếu việc sao lưu không thể thực hiện gọn vào trong một băng từ, lúc đó hãy sử dụng tùy chọn **-M**:

```
# tar -cMf /dev/fd0H1440 /usr/src
tar: Removing leading / from absolute path names in the archive
Prepare volume #2 for /dev/fd0H1440 and hit return:
#
```

Chú ý rằng phải định dạng đĩa mềm trước khi thực hiện việc sao lưu, có thể sử dụng một thiết bị đầu cuối khác để thực hiện việc định dạng đĩa khi **tar** yêu cầu một đĩa mềm mới.

Sau khi thực hiện việc sao lưu, có thể kiểm tra kết quả của công việc bằng tùy chọn **--compare**:

```
# tar --compare --verbose -f /dev/ftape
usr/src/
usr/src/Linux
usr/src/Linux-1.2.10-includes/
...
#
```

Để sử dụng kiểu sao lưu tăng dần, hãy sử dụng tùy chọn **-N**:

```
# tar --create --newer '8 Sep 1995' --file /dev/ftape /usr/src
--verbose
tar: Removing leading / from absolute path names in the
archive
usr/src/
usr/src/Linux-1.2.10-includes/
usr/src/Linux-1.2.10-includes/include/
usr/src/Linux-1.2.10-includes/include/Linux/
usr/src/Linux-1.2.10-includes/include/Linux/modules/
usr/src/Linux-1.2.10-includes/include/asm-generic/
usr/src/Linux-1.2.10-includes/include/asm-i386/
usr/src/Linux-1.2.10-includes/include/asm-mips/
usr/src/Linux-1.2.10-includes/include/asm-alpha/
usr/src/Linux-1.2.10-includes/include/asm-m68k/
usr/src/Linux-1.2.10-includes/include/asm-sparc/
usr/src/patch-1.2.11.gz
#
```

Lưu ý rằng, **tar** không thể thông báo được khi các thông tin trong ***inode*** của một file bị thay đổi, ví dụ như thay đổi quyền truy nhập của file, hay thay đổi tên file chẳng hạn. Để biết được những thông tin thay đổi sẽ cần dùng đến lệnh **find** và so sánh với trạng thái hiện thời của file hệ thống với danh sách các file được sao lưu từ trước.

3.5.2 Nén dữ liệu

Việc sao lưu rất có ích nhưng đồng thời nó cũng chiếm rất nhiều không gian cần thiết để sao lưu. Để giảm không gian lưu trữ cần thiết, có thể thực hiện việc nén dữ liệu trước khi sao lưu, sau đó thực hiện việc giải nén (dãn) để nhận lại nội dung trước khi nén.

Trong Linux có khá nhiều cách để nén dữ liệu, tài liệu này giới thiệu hai phương cách phổ biến là **gzip** và **compress**.

*** Nén, giải nén và xem nội dung các file với lệnh *gzip, gunzip* và *zcat***

Cú pháp các lệnh này như sau:

```
gzip [tùy-chọn] [ -S suffix ] [ < file> ]
gunzip [tùy-chọn] [ -S suffix ] [ <file> ]
zcat [tùy-chọn] [ <file> ]
```

Lệnh **gzip** sẽ làm giảm kích thước của file và khi sử dụng lệnh này, file gốc sẽ bị thay thế bởi file nén với phần mở rộng là .gz, các thông tin khác liên quan đến file không thay đổi. Nếu không có tên file nào được chỉ ra thì thông tin từ thiết bị vào chuẩn sẽ được nén và gửi ra thiết bị ra chuẩn. Trong một vài trường hợp, lệnh này sẽ bỏ qua liên kết tượng trưng.

Nếu tên file nén quá dài so với tên file gốc, **gzip** sẽ cắt bỏ bớt. **gzip** sẽ chỉ cắt phần tên file vượt quá 3 ký tự (các phần được ngăn cách với nhau bởi dấu chấm). Nếu tên file gồm nhiều phần nhỏ thì phần dài nhất sẽ bị cắt bỏ. Ví dụ, tên file là **gzip.msdos.exe**, khi được nén sẽ có tên là **gzip.msd.exe.gz**.

File được nén có thể được khôi phục trở lại dạng nguyên thể với lệnh **gzip -d** hoặc **gunzip**.

Với lệnh **gzip** có thể giải nén một hoặc nhiều file có phần mở rộng là **.gz**, **-gz**, **.z**, **-z**, **_z** hoặc **.Z** ... **gunzip** dùng để giải nén các file nén bằng lệnh **gzip**, **zip**, **compress**, **compress -H**.

Lệnh **zcat** được sử dụng khi muốn xem nội dung một file nén trên thiết bị ra chuẩn.

Các tùy chọn:

- **-c, --stdout --to-stdout** : đưa ra trên thiết bị ra chuẩn; giữ nguyên file gốc không có sự thay đổi. Nếu có nhiều hơn một file đầu vào, đầu ra sẽ tuần tự là các file được nén một cách độc lập.
- **-d, --decompress --uncompress** : giải nén.
- **-f, --force** : thực hiện nén hoặc giải nén thậm chí file có nhiều liên kết hoặc file tương ứng thực sự đã tồn tại, hay dữ liệu nén được đọc hoặc ghi trên thiết bị đầu cuối.
- **-h, --help** : hiển thị màn hình trợ giúp và thoát.
- **-l, --list** : hiển thị những thông tin sau đối với một file được nén:
  - **compressed size:** kích thước của file nén
  - **uncompressed size**: kích thước của file được giải nén
  - **ratio**: tỷ lệ nén (0.0% nếu không biết)
  - **uncompressed_name**: tên của file được giải nén

Nếu kết hợp với tùy chọn **--verbose**, các thông tin sau sẽ được hiển thị:

- **method**: phương thức nén
- **crc**: CRC 32-bit cho dữ liệu được giải nén
- **date & time**: thời gian các file được giải nén

Nếu kết hợp với tùy chọn **--name**, tên file được giải nén, thời gian giải nén được lưu trữ trong file nén

Nếu kết hợp với tùy chọn **--verbose**, tổng kích thước và tỷ lệ nén của tất cả các file sẽ được hiển thị

Nếu kết hợp với tùy chọn **--quiet**, tiêu đề và tổng số dòng của các file nén không được hiển thị.

- **-n, --no-name** : khi nén, tùy chọn này sẽ không lưu trữ tên file gốc và thời gian nén, (tên file gốc sẽ luôn được lưu nếu khi nén tên của nó bị cắt bỏ). Khi giải nén, tùy chọn này sẽ không khôi phục lại tên file gốc cũng như thời gian thực hiện việc nén. Tùy chọn này được ngầm định.

- **-N, --name** : tùy chọn này ngược với tùy chọn trên (**-n**), nó hữu ích trên hệ thống có sự giới hạn về độ dài tên file hay khi thời điểm nén bị mất sau khi chuyển đổi file.
- **-q, --quiet** : bỏ qua mọi cảnh báo.
- **-r, --recursive** : nén thư mục.
- **-S .suf, --suffix .suf** : sử dụng phần mở rộng **.suf** thay cho **.gz**. Bất kỳ phần mở rộng nào cũng có thể được đưa ra, nhưng các phần mở rộng khác **.z** và **.gz** sẽ bị ngăn chặn để tránh sự lộn xộn khi các file được chuyển đến hệ thống khác.
- **-t, --test** : tùy chọn này được sử dụng để kiểm tra tính toàn vẹn của file được nén
- **-v, --verbose** : hiển thị phần trăm thu gọn đối với mỗi file được nén hoặc giải nén
- **-#, --fast, --best** : điều chỉnh tốc độ của việc nén bằng cách sử dụng dấu #,
    - nếu -# là **-1** hoặc **--fast** thì sử dụng phương thức nén nhanh nhất (less compression),
    - nếu là **-9** hoặc **--best** thì sẽ dùng phương thức nén chậm nhất (best compression).
    - Ngầm định mức nén là **-6** (đây là phương thức nén theo tốc độ nén cao).

Ví dụ:

```
# ls /home/test
Desktop data dictionary newt-0.50.8 rpm save vd1
# gzip /home/test/vd1
# ls /home/test
Desktop data dictionary newt-0.50.8 rpm save vd1.gz
# zcat /home/test/vd1
PID TTY TIME CMD
973 pts/0 00:00:00 bash
996 pts/0 00:00:00 man
1008 pts/0 00:00:00 sh
1010 pts/0 00:00:00 less
1142 pts/0 00:00:00 cat
1152 pts/0 00:00:00 cat
1181 pts/0 00:00:00 man
1183 pts/0 00:00:00 sh
1185 pts/0 00:00:00 less
#
```

*** Nén, giải nén và xem file với các lệnh *compress, uncompress, zcat***

Cú pháp các lệnh như sau:

```
compress [tùy-chọn] [<file>]
uncompress [tùy-chọn] [<file>]
zcat [tùy-chọn] [<file>]
```

Lệnh **compress** làm giảm kích thước của file và khi sử dụng lệnh này, file gốc sẽ bị thay thế bởi file nén với phần mở rộng là **.Z**, các thông tin khác liên quan đến file không

Các tùy chọn:

- **-f** : nếu tùy chọn này không được đưa ra và **compress** chạy trong chế độ nền trước, người dùng sẽ được nhắc khi các file đã thực sự tồn tại và có thể bị ghi đè. Các file được nén có thể được khôi phục lại nhờ việc sử dụng lệnh **uncompress**.
- **-c** : tùy chọn này sẽ thực hiện việc nén hoặc giải nén rồi đưa ra thiết bị ra chuẩn, không có file nào bị thay đổi.

Lệnh **zcat** tương đương với **uncompress -c**. **zcat** thực hiện việc giải nén hoặc là các file được liệt kê trong dòng lệnh hoặc từ thiết bị vào chuẩn để đưa ra dữ liệu được giải nén trên thiết bị ra chuẩn.

- **-r** : nếu tùy chọn này được đưa ra, **compress** sẽ thực hiện việc nén các thư mục.
- **-v** : hiển thị tỷ lệ giảm kích thước cho mỗi file được nén.

CHƯƠNG 4. QUẢN TRỊ QUÁ TRÌNH

4.1 Quá trình trong UNIX

4.1.1. Sơ bộ về quá trình

Quá trình là đối tượng trong hệ thống tương ứng với một phiên thực hiện của một chương trình. Quá trình bao gồm ba thành phần là text, data, stack. Text là thành phần câu lệnh thực hiện, data là thành phần dữ liệu còn stack là thành phần thông tin tạm thời hoạt động theo cơ chế LIFO. Các câu lệnh trong text chỉ thao tác tới vùng data, stack tương ứng của quá trình, không truy nhập được tới data và stack của các quá trình khác, ngoại trừ các vùng dữ liệu dùng chung.

Các quá trình được hệ thống phân biệt bằng số hiệu của quá trình, viết tắt là PID (Proccess Index). Quá trình được tạo khi khởi động hệ điều hành là quá trình 0. Mọi quá trình khác đều được tạo ra từ một quá trình khác thông qua lời gọi hệ thống ***fork***: quá trình thực hiện lời gọi hệ thống ***fork*** được gọi là quá trình cha, còn quá trình được tạo ra theo lời gọi ***fork*** được gọi là quá trình con. Trừ quá trình 0 không có cha, mọi quá trình có trong hệ thống đều có một cha và một cha có thể có nhiều con.

Kết quả dịch chương trình nguồn sẽ tạo ra file chương trình đích gồm một số phần như sau (lưu trữ trên vật dẫn ngoài):

- Phần đầu file mô tả một số đặc tính của file chương trình (tương tự File header của file chương trình trong MS-DOS),
- Phần text của chương trình,
- Các giá trị mở đầu về việc phân phối bộ nhớ đối với vùng data của chương trình,
- Một số bảng thông tin liên quan đến đặt file.
- Khi có lời gọi ***fork***, thông qua lời gọi hệ thống ***exec***, nhân sẽ tải nội dung của file chương trình vào bộ nhớ trong theo các vùng text, data và stack:
- Vùng text của quá trình tương ứng với file chương trình,
- Vùng data của quá trình tương ứng với các giá trị được quy định trong file chương trình,
- Vùng stack được nhân tự động tạo với kích thước theo sự linh hoạt của nhân.

Phần stach bao gồm các stack frame (khung) lôgic: mỗi satck frame được đặt vào khi ***gọi một hàm*** và lấy ra khi quay về. Mỗi stack frame chứa tham sô của hàm, các biến địa phương v.v. Tương ứng trong stack có một stack pointer liên quan đến chiều sâu của stack. Trong mã chương trình có các dòng lệnh quản lý hình trạng của stack, và nhân sẽ định vị không gian đối với stack theo yêu cầu.

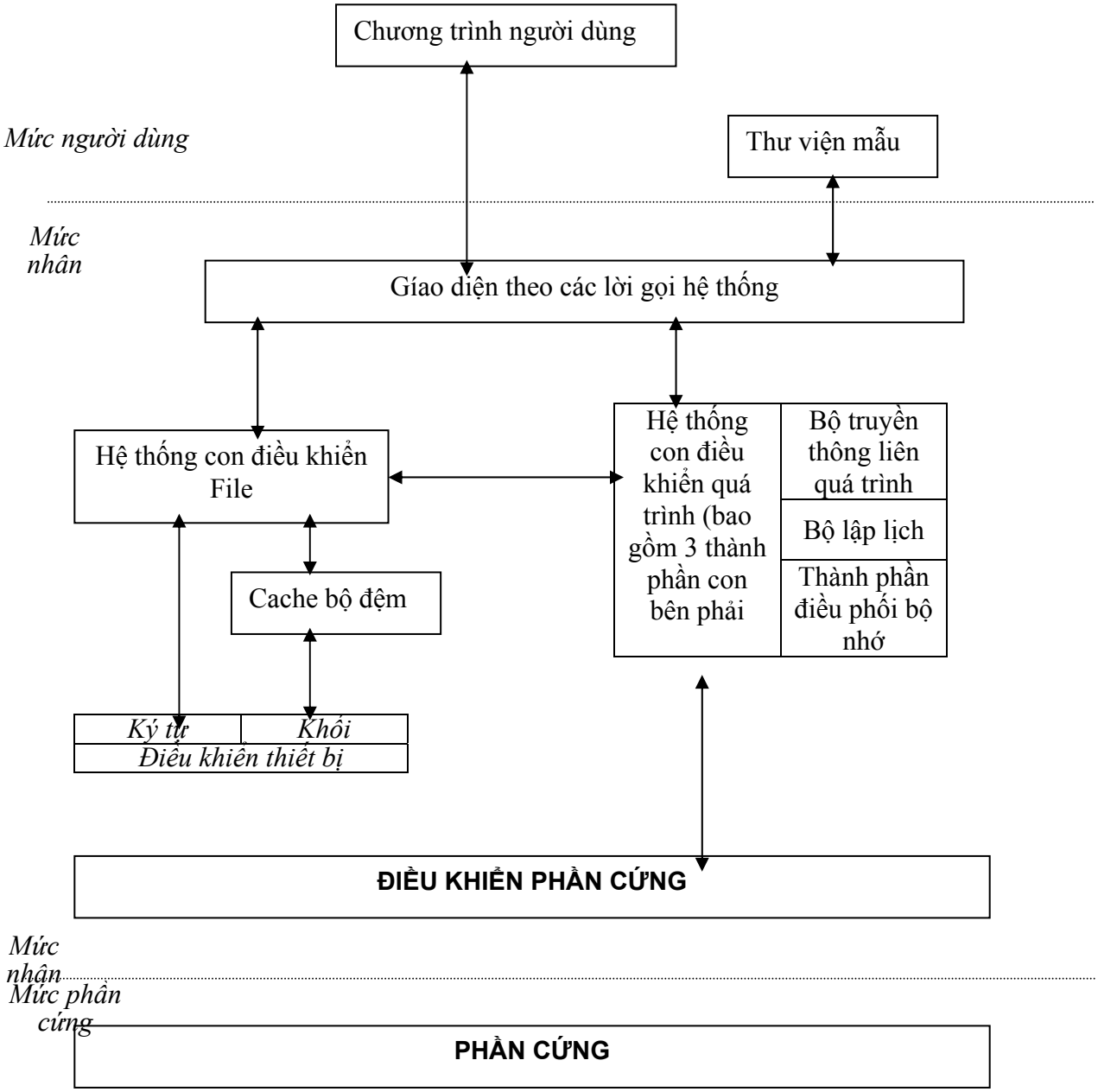
4.1.2. Sơ bộ cấu trúc điều khiển của UNIX

Theo phân cấp, hệ thống thực hiện theo ba mức: mức người dùng, mức nhân và mức phần cứng.

- Mức người dùng (user level): gồm có chương trình người dùng và chương trình trong các thư viện. Các chương trình này chạy (phần lệnh của chúng thực hiện) trong trạng thái người dùng của quá trình. Chương trình người dùng thao tác với nhân hoặc trực tiếp hoặc gián tiếp nhờ gọi thư viện nhờ các ***lời gọi hệ thống***.

- Mức nhân là mức trọng tâm nhất của hệ điều hành Linux-UNIX. Chạy ở mức nhân là những chương trình của hệ điều hành thuộc hệ thống con điều khiển File (hệ thống con làm việc với File - File Subsystem), hệ thống con điều khiển quá trình (Proccess Control System), các lời gọi hệ thống (system calls), các chương trình điều khiển thiết bị (Device Drivers), Cache bộ đệm (Buffer cache) và các chương trình điều khiển phần cứng (Hard Control). Hai thành phần cơ bản nhất là Hệ thống điều khiển File và Hệ thống con điều khiển quá trình.

Hình vẽ dưới đây cho sơ bộ cấu trúc điều khiển trong UNIX:



Cấu trúc của Nhân và các mức quá trình

4.1.3. Các hệ thống con trong nhân

★ Hệ thống con điều khiển File có nhiệm vụ quản lý hệ thống File, cung cấp vùng nhớ rỗi ở đĩa cho File, điều khiển truy cập File và tìm kiếm dữ liệu v.v. Đa số các thuật toán về các lời gọi hệ thống liên quan đến File và các hàm chương trình con mức thấp đã được trình bày trong chương 2. Các quá trình tương tác với Hệ thống con điều khiển File nhờ các lời gọi hệ thống (các lời gọi hệ thống File mức cao). Việc truy nhập tới File nhờ hai cách thức: truy nhập trực tiếp với File hoặc thông qua buffer cache.

- Các buffer cache lưu trữ dữ liệu tạm thời theo từng khối. Nhân vào-ra dữ liệu thông qua các khối trung gian và nhờ thiết bị nhớ thứ cấp: truy nhập dữ liệu theo khối,
- Nhân thao tác trực tiếp với khối điều khiển thiết bị để truy nhập trực tiếp dữ liệu trong File không qua thiết bị phụ: truy nhập theo ký tự.

★ Hệ thống con điều khiển quá trình chịu trách nhiệm đồng bộ hóa sự tương tác liên quá trình, quản lý bộ nhớ và lập lịch thực hiện đối với các quá trình đang tồn tại. Hệ thống con điều khiển File và Hệ thống con điều khiển quá trình tương tác với nhau khi file được tải vào bộ nhớ trong và cho thực hiện. Một số lời gọi hệ thống cho khối điều khiển quá trình:

- fork: Tạo quá trình mới. Lời gọi hàm này có dạng pid=fork()
- exec: Cho thực hiện quá trình đang tồn tại; exec(pid)
- exit: Cho kết thúc quá trình đang tồn tại,
- brk: điều khiển kích thước bộ nhớ cấp phát cho quá trình,
- signal: Điều khiển các hiện tượng bất thường trong quá trình

Hệ thống con điều khiển quá trình bao gồm 3 thành phần sau đây:

- Thành phần điều phối bộ nhớ có nhiệm vụ quản lý, điều khiển cấp phát bộ nhớ. Một số trang bị loại bỏ khi cấp phát bộ nhớ cho quá trình.
- Bộ lập lịch (schelduler) có nhiệm vụ điều phối CPU cho các quá trình. Các quá trình có độ ưu tiên và bộ lập lịch chọn quá trình có độ ưu tiên cao nhất.
- Bộ truyền thông liên quá trình thực hiện việc đồng bộ hóa các quá trình liên quan nhau.

★ Bộ điều khiển phần cứng (hardware control) có chức năng cho phép ngắt và tương tác thông tin với máy. Các thiết bị như đĩa, thiết bị đầu cuối có thể ngắt CPU khi đang thực hiện quá trình. Các chương trình xử lý ngắt là hàm riêng biệt trong nhân mà không phải là một quá trình.

Stack trong quá trình

Mỗi quá trình thực hiện được mode nhân và mode người dùng vì vậy phân chia hai loại stack nhân và stack người dùng.

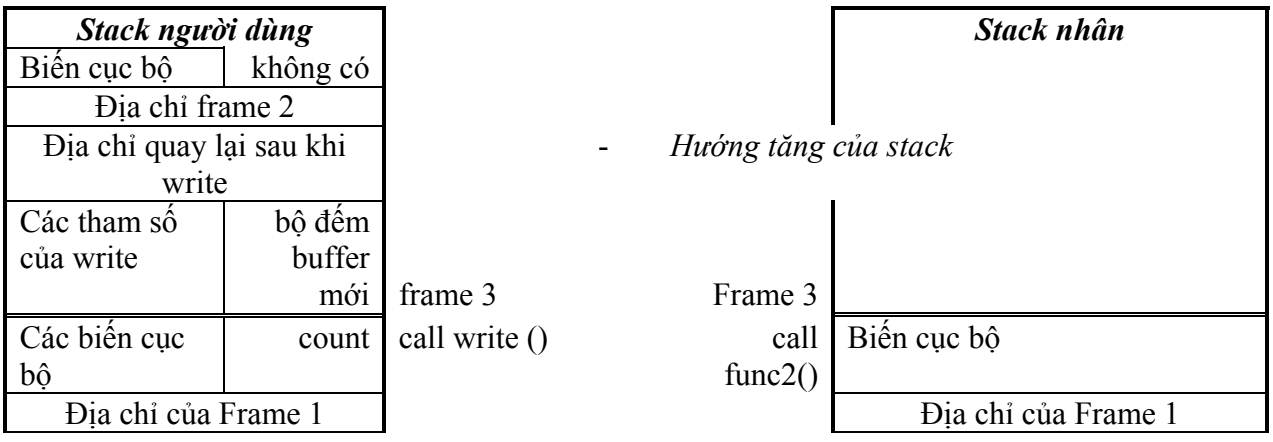
Chúng ta xem xét ví dụ sau:

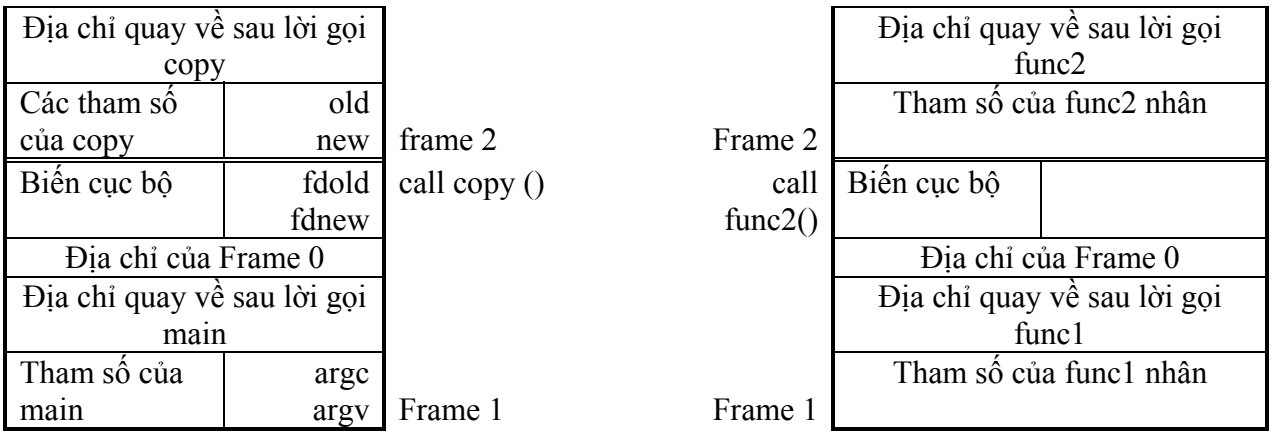
```
#include <fenth.h>
char buffer[2048];
int version;
main (argc, argv);
        int argc;
        char *argv[];
|
        int fdold, fdnew;
```

```
        if (argc != 3)
        |
                printf(' cần 2 đối số đối với chương trình sao file!');
                exit(1)
        |
        fdold = open (argv[1], O_RDONLY);          /* mở file nguồn chỉ đọc */
        if (fdold == -1)
        |
                printf (' Không thể mở file &cs\n',argv[1]);
                exit(1);
        |
        fdnew =creat (argv[2],0666); /*mở File đích rw cho mọi người */
        if (fdnew ==-1)
        |
                printf('Không thể khởi tạo file &cs\n',argv[2];
                exit(1);
        |
        copy(fdold,fdnew);
        exit(0);
|
copy (old, new)
        int old, new;
|
        int count;
        while (count = read(old,buffer,sizeof(buffer))>0)
                write(buffer,count);
|
```

Trong chương trình trên, mã lệnh (gọi là phần text) của file được sinh ra từ các hàm main và copy. Khởi tạo giá trị ban đầu cho biến version và dành vùng nhớ cho biến mảng buffer.

Trong ví dụ trên, các tham số argc, argv và các biến fdold, fdnew trong chương trình main trong stack khi main được gọi (một lần đối với mọi chương trình), còn các tham số old và new và biến count trong hàm copy xuất hiện mỗi khi copy được gọi.





Các stack cho một quá trình

Quá trình trong UNIX được thực hiện theo một trong hai mode: mode nhân hay mode người dùng và tương ứng với 2 mode này, quá trình sử dụng stack riêng biệt đối với mỗi mode.

Stack người dùng chứa các đối số, biến cục bộ, và các dữ liệu khác đối với việc thực hiện hàm trong mode người dùng. Stach nhân chứa các đối số, biến cục bộ, các tham số, các địa chỉ liên kết v.v. liên quan đến thực hiện các hàm theo mode nhân.

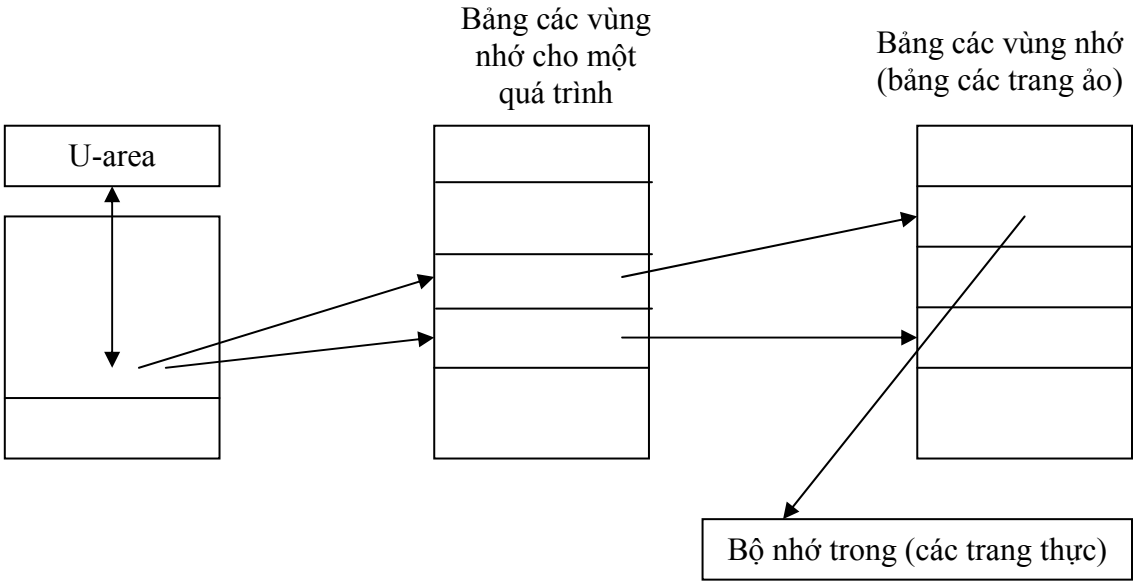
4.1.4. Sơ bộ về điều khiển quá trình

Nhân sử dụng 4 cấu trúc dữ liệu sau đây để truy nhập đến quá trình:

- Bảng các quá trình, tương ứng với mỗi quá trình đang tồn tại trong hệ thống là một thành phần. Mỗi thành phần bao gồm một số trường sau đây (mỗi thành phần ở đây chính là một PCB):
  - Trạng thái của quá trình,
  - Chủ sở hữu của quá trình,
  - Trường liên quan đến trạng thái ngưng của quá trình (theo lời gọi hàm sleep)
  - Địa chỉ của vùng sử dụng tương ứng với quá trình,
  - Các thông tin tương ứng được trình bày trong PCB.
- Vùng sử dụng (U-area) chứa các thông tin riêng, có tác dụng khi quá trình đang thực hiện:
  - Chỉ số thành phần tương ứng với quá trình trong bảng các quá trình: địa chỉ của khối PCB tương ứng,
  - Bộ đếm thời gian chạy mức nhân và mức người dùng,
  - Các giá trị trả về và mã lỗi (nếu có) đối với lời gọi hệ thống hiện tại,
  - Mô tả về các file đang mở ứng với quá trình,
  - Tham số lưu trữ dung lượng dữ liệu di chuyển trong vào - ra.
  - Thư mục hiện tại và thư mục gốc hiện tại: môi trường của quá trình,
  - Các giới hạn kích thước file và quá trình,
  - Các mức cho phép thực hiện đối với quá trình,
  - Một số thông tin khác
- Các bảng định vị địa chỉ bộ nhớ đối với mỗi quá trình,
- Bảng chứa vùng bộ nhớ chung: phân hoạch bộ nhớ, đặc tính mỗi vùng theo phân hoạch: chứa text, data hoặc vùng bộ nhớ dùng chung v.v.

Sơ bộ về mối liên kết của các cấu trúc dữ liệu trên được mô tả như hình vẽ phía sau. Nhân xử lý với các lời gọi hệ thống như sau:

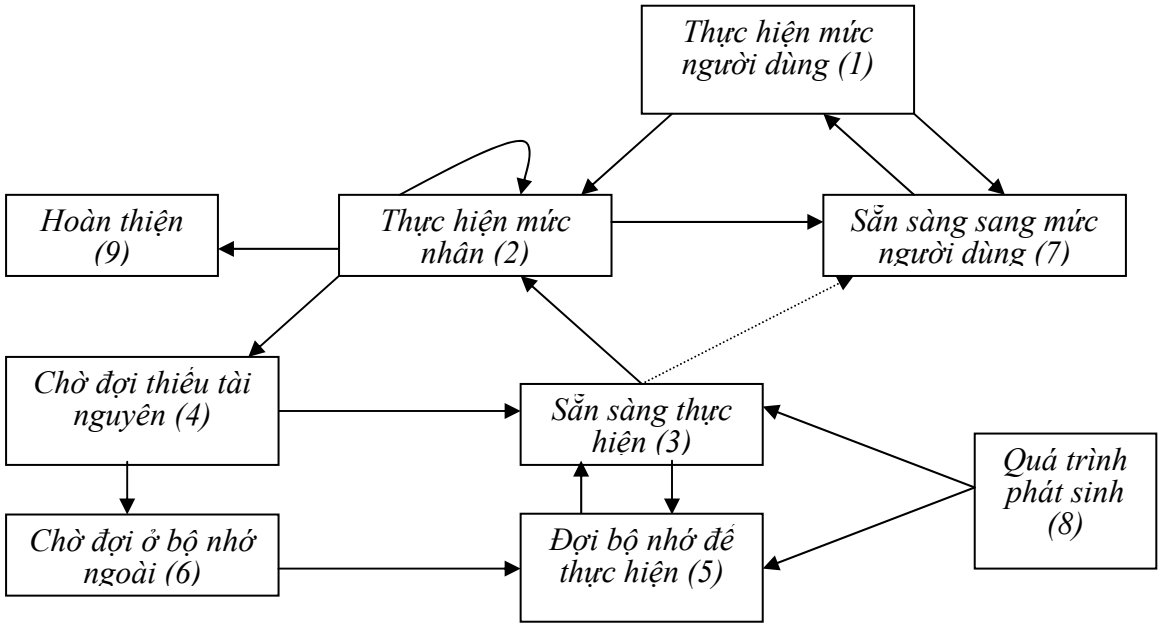
- Với lời gọi fork: Nhân sao vùng địa chỉ của quá trình cũ, cho phép các quá trình chia xẻ vùng bộ nhớ,
- Với lời gọi exec: Nhân cấp phát các vùng bộ nhớ thực cho các vùng text, data và stack,
- Với lời gọi exit: Nhân sẽ giải phóng các vùng bộ nhớ liên quan đến quá trình.



Các cấu trúc dữ liệu điều khiển quá trình

4.1.5. Trạng thái và chuyển dịch trạng thái

Sơ đồ biểu diễn các trạng thái và việc chuyển trạng thái trong UNIX được trình bày trong hình dưới đây (Số hiệu trạng thái quá trình xem trong hình vẽ).



Sơ đồ chuyển trạng thái quá trình

Khi quá trình được phát sinh nó ở trạng thái (8), tùy thuộc vào tình trạng bộ nhớ quá trình được phân phối bộ nhớ trong (3) hay bộ nhớ ngoài (5). Trạng thái (3) thể hiện quá trình đã sẵn sáng thực hiện, các thành phần của nó đã ở bộ nhớ trong chờ đợi CPU để thực hiện. Việc thực hiện tiếp theo tùy thuộc vào trạng thái trước đó của nó. Nếu lần đầu phát sinh, nó cần đi tới thực hiện mức nhân để hoàn thiện công việc lời gọi fork sẽ từ trạng thái (3) sang trạng thái (1), trong trường hợp khác, từ trạng thái (3) nó đi tới trạng thái chờ dợi CPU ở mức người dùng (7).

Trong trạng thái thực hiện ở mức người dùng (1), quá trình đi tới trạng thái (2) khi gặp lời gọi hệ thống hoặc hiện tượng ngắt xảy ra. Từ trạng thái (1) tới trạng thái (7) khi hết lượng tử thời gian.

Trạng thái (4) là trạng thái chờ đợi trong bộ nhớ còn trạng thái (6) thể hiện việc chờ đợi trong bộ nhớ ngoài.

Cung chuyển từ trạng thái (2) vào ngay trạng thái (2) xảy ra khi ở quá trình ở trạng thái thực hiện mức nhân, nhân hệ thống gọi các hàm xử lý ngắt tương ứng.

4.1.6. Sự ngưng hoạt động và hoạt động trở lại của quá trình

Một quá trình trong trạng thái thực hiện mức nhân có khả năng chuyển sang trạng thái *ngưng* theo lời gọi hàm *sleep*. Trạng thái ngưng xảy ra trong một số tình huống chờ đợi một sự kiện: hoàn thành việc vào-ra, quá trình khác thực hiện lời gọi exit v.v.

Sau khi sự kiện xảy ra, quá trình từ trạng thái ngưng chuyển sang trạng thái sẵn sàng để có thể được cấp phát CPU chạy.

4.1.7. Sơ bộ về lệnh đối với quá trình

Khi mở một trang **man**, liệt kê các file với lệnh **ls**, chạy trình soạn thảo **vi** hay chạy bất kỳ một lệnh nào trong Linux thì điều đó có nghĩa là đang khởi tạo một hoặc nhiều quá trình. Trong Linux, bất cứ chương trình nào đang chạy đều được coi là một quá trình. Có thể có nhiều quá trình cùng chạy một lúc. Ví dụ dòng lệnh **ls -l | sort | more** sẽ khởi tạo ba quá trình: **ls**, **sort** và **more**.

Quá trình có thể trải qua nhiều trạng thái khác nhau và tại một thời điểm một quá trình rơi vào một trong các trạng thái đó. Bảng dưới đây giới thiệu các trạng thái cơ bản của quá trình trong Linux.

Ký hiệu	Ý nghĩa
D	(**uninterruptible sleep**) ở trạng thái này quá trình bị treo và không thể chạy lại nó bằng một tín hiệu.
R	(**runnable**) trạng thái sẵn sàng thực hiện, tức là quá trình có thể thực hiện được nhưng chờ đến lượt thực hiện vì một quá trình khác đang có CPU.
S	(**sleeping**) trạng thái tạm dừng, tức là quá trình tạm dừng không hoạt động (20 giây hoặc ít hơn)
T	(**traced or stopped**) trạng thái dừng, quá trình có thể bị treo bởi một quá trình ngoài
Z	

W	(**zombie process**) quá trình đã kết thúc thực hiện, nhưng nó vẫn được tham chiếu trong hệ thống
<	không có các trang thường trú
N	quá trình có mức ưu tiên cao hơn
L	quá trình có mức ưu tiên thấp hơn
	có các trang khóa bên trong bộ nhớ

4.2. Các lệnh cơ bản

4.2.1. Lệnh fg và lệnh bg

Linux cho phép người dùng sử dụng tổ hợp phím **CTRL+z** để dừng một quá trình và khởi động lại quá trình đó bằng cách gõ lệnh **fg**. Lệnh **fg** (**foreground**) tham chiếu đến các chương trình mà màn hình cũng như bàn phím đang làm việc với chúng.

Ví dụ, người dùng đang xem trang **man** của lệnh **sort**, nhìn xuống cuối thấy có tùy chọn **-b**, muốn thử tùy chọn này đồng thời vẫn muốn xem trang **man**. Thay cho việc đánh **q** để thoát và sau đó chạy lại lệnh **man**, cho phép người dùng gõ **CTRL+z** để tạm dừng lệnh **man** và gõ lệnh thử tùy chọn **-b**. Sau khi thử xong, hãy gõ **fg** để tiếp tục xem trang **man** của lệnh **sort**. Kết quả của quá trình trên hiển thị như sau:

```
# man sort | more
SORT(1) FSF SORT(1)
NAME
sort - sort lines of text Files
SYNOPSIS
 ../src/sort [OPTION] ... [Files]...
DESCRIPTION
Write sorted concatenation of all FILE(s) to standard out-put.
 +POS1 [-POS2]
start a key at POS1,end it *before* POS2 obsoles-cent)field numbers and character offsets are num-bered starting with zero(contrast with the -k option)
-b ignore leading blanks in sort fields or keys
          --More--
(CTRL+z)
[1]+ Stopped    man sort | more
# ls -s | sort -b | head -4
  1 Archives/
  1 InfoWorld/
  1 Mail/
  1 News/
  1 OWL/
# fg
 man sort | more
 --More--
```

Trong phần trước, cách thức gõ phím **CTRL+z** để tạm dừng một quá trình đã được giới thiệu. Linux còn người dùng cách thức để chạy một chương trình dưới chế độ nền (***background***) - sử dụng lệnh **bg** - trong khi các chương trình khác đang chạy, và để chuyển một chương trình vào trong chế độ nền - dùng ký hiệu **&**.

Nếu một quá trình hoạt động mà không đưa ra thông tin nào trên màn hình và không cần nhận bất kỳ thông tin đầu vào nào, thì có thể sử dụng lệnh **bg** để đưa nó vào trong chế độ nền (ở chế độ này nó sẽ tiếp tục chạy cho đến khi kết thúc). Khi chương trình cần đưa thông tin ra màn hình hoặc nhận thông tin từ bàn phím, hệ thống sẽ tự động dừng chương trình và thông báo cho người dùng. Cũng có thể sử dụng chỉ số điều khiển công việc (***job control***) để làm việc với chương trình nào muốn. Khi chạy một chương trình trong chế độ nền, chương trình đó được đánh số thứ tự (được bao bởi dấu ngoặc vuông []), theo sau là chỉ số của quá trình.

Sau đó có thể sử dụng lệnh **fg + số thứ tự của chương trình** để đưa chương trình trở lại chế độ nổi và tiếp tục chạy.

Để có một chương trình (hoặc một lệnh ống) tự động chạy trong chế độ nền, chỉ cần thêm ký hiệu '**&**' vào cuối lệnh.

Trong một số hệ thống, khi quá trình nền kết thúc thì hệ thống sẽ gửi thông báo tới người dùng, nhưng trên hầu hết các hệ thống, khi quá trình trên nền hoàn thành thì hệ thống sẽ chờ cho đến khi người dùng gõ phím **Enter** thì mới hiển thị dấu nhắc lệnh mới kèm theo thông báo hoàn thành quá trình (thường thì một quá trình hoàn thành sau khoảng 20 giây).

Nếu cố để chuyển một chương trình vào chế độ nền mặc dù nó có các thông tin cần xuất hoặc nhập từ các thiết bị vào ra chuẩn thì hệ thống sẽ đưa ra thông báo lỗi dưới dạng sau: **Stopped (tty input/output) tên chương trình**.

Ví dụ, lệnh sau đây thực hiện việc tìm kiếm file **thu1** trong chế độ nền:

```
# find -name thu1 &
[5] 918
```

trong chế độ này, số thứ tự của chương trình là **[5]**, chỉ số quá trình tương ứng với lệnh **find** là **918**. Vì gõ Enter khi quá trình chưa thực hiện xong nên trên màn hình chỉ hiển thị số thứ tự của chương trình và chỉ số quá trình, nếu chờ khoảng 30 hoặc 40 giây sau rồi gõ Enter lần nữa, màn hình hiển thị thông báo hoàn thành chương trình như sau:

```
#
[5] Done          find -name thu1
#
```

Giả sử chương trình chưa hoàn thành và muốn chuyển nó lên chế độ nổi, hãy gõ lệnh sau:

```
# fg 5
find -name thu1
./thu1
```

chương trình đã hoàn thành và hiển thị thông báo rằng file **thu1** nằm ở thư mục gốc.

Thông thường sẽ đưa ra một thông báo lỗi nếu người dùng cố chuyển một chương trình vào chế độ nền khi mà chương trình đó cần phải xuất hoặc nhập thông tin từ thiết bị vào ra chuẩn. Ví dụ, lệnh:

```
# vi &
[6] 920
```

```
#
```

nhấn Enter

```
#
[6] + Stopped (tty output) vi
#
```

Lệnh trên chạy chương trình **vi** trong chế độ nền, tuy nhiên lệnh gặp phải lỗi vì đây là chương trình đòi hỏi hiển thị các thông tin ra màn hình (**output**). Dòng thông báo lỗi **Stopped (tty intput) vi** cũng xảy ra khi chương trình **vi** cần nhận thông tin.

4.2.2. Hiển thị các quá trình đang chạy với lệnh ps

Linux cung cấp cho người dùng hai cách thức nhận biết có những chương trình nào đang chạy trong hệ thống. Cách dễ hơn, đó là lệnh **jobs** sẽ cho biết các quá trình nào đã dừng hoặc là được chạy trong chế độ nền.

Cách phức tạp hơn là sử dụng lệnh **ps**. Lệnh này cho biết thông tin đầy đủ nhất về các quá trình đang chạy trên hệ thống.

Ví dụ:

```
# ps
PID    TTY   TIME   CMD
7813   pts/0      00:00:00 bash
7908   pts/0      00:00:00 ps
#
```

(PID - chỉ số của quá trình, TTY - tên thiết bị đầu cuối trên đó quá trình được thực hiện, TIME - thời gian để chạy quá trình, CMD - lệnh khởi tạo quá trình).

Cú pháp lệnh **ps**:

```
ps [tùy-chọn]
```

Lệnh **ps** có một lượng quá phong phú các tùy chọn được chia ra làm nhiều loại. Dưới đây là một số các tùy chọn hay dùng. Các tùy chọn đơn giản:

- **-A, -e** : chọn để hiển thị tất cả các quá trình.
- **-T** : chọn để hiển thị các quá trình trên trạm cuối đang chạy.
- **-a** : chọn để hiển thị tất cả các quá trình trên một trạm cuối, bao gồm cả các quá trình của những người dùng khác.
- **-r** : chỉ hiển thị quá trình đang được chạy.
  - ***Chọn theo danh sách***
- **-C** : chọn hiển thị các quá trình theo tên lệnh.
- **-G** : hiển thị các quá trình theo chỉ số nhóm người dùng.
- **-U** : hiển thị các quá trình theo tên hoặc chỉ số của người dùng thực sự (người dùng khởi động quá trình).
- **-p** : hiển thị các quá trình theo chỉ số của quá trình.
- **-s** : hiển thị các quá trình thuộc về một phiên làm việc.
- **-t** : hiển thị các quá trình thuộc một trạm cuối.
- **-u** : hiển thị các quá trình theo tên và chỉ số của người dùng hiệu quả.

- ***Thiết đặt khuôn dạng được đưa ra của các quá trình***

- **-f** : hiển thị thông tin về quá trình với các trường sau UID - chỉ số người dùng, PID - chỉ số quá trình, PPID - chỉ số quá trình khởi tạo ra quá trình, C - , STIME - thời gian khởi tạo quá trình, TTY - tên thiết bị đầu cuối trên đó quá trình được chạy, TIME - thời gian để thực hiện quá trình, CMD - lệnh khởi tạo quá trình
- **-l** : hiển thị đầy đủ các thông tin về quá trình với các trường F, S, UID, PID, PPID, C, PRI, NI, ADDR, SZ, WCHAN, TTY, TIME, CMD
- **-o xâu-chọn** : hiển thị các thông tin về quá trình theo dạng do người dùng tự chọn thông qua xâu-chọn các kí hiệu điều khiển hiển thị có các dạng như sau:

%C, %cpu	% CPU được sử dụng cho quá trình
%mem	% bộ nhớ được sử dụng để chạy quá trình
%G	tên nhóm người dùng
%P	chỉ số của quá trình cha khởi động ra quá trình con
%U	định danh người dùng
%c	lệnh tạo ra quá trình
%p	chỉ số của quá trình
%x	thời gian để chạy quá trình
%y	thiết bị đầu cuối trên đó quá trình được thực hiện

Ví dụ, muốn xem các thông tin như tên người dùng, tên nhóm, chỉ số quá trình, chỉ số quá trình khởi tạo ra quá trình, tên thiết bị đầu cuối, thời gian chạy quá trình, lệnh khởi tạo quá trình, hãy gõ lệnh:

```
# ps -o '%U %G %p %P %y %x %c'

USER GROUP PID PPID TTY TIME COMMAND
root root 1929 1927 pts/1 00:00:00 bash
root root 2279 1929 pts/1 00:00:00 ps
```

4.2.3. Hủy quá trình với lệnh kill

Trong một số trường hợp, sử dụng lệnh **kill** để hủy bỏ một quá trình. Điều quan trọng nhất khi sử dụng lệnh **kill** là phải xác định được chỉ số của quá trình mà chúng ta muốn hủy. Cú pháp lệnh:

```
kill [tùy-chọn] <chỉ-số-của-tiến-trình>
kill -l [tín hiệu]
```

Lệnh **kill** sẽ gửi một **tín hiệu** đến quá trình được chỉ ra. Nếu không chỉ ra một tín hiệu nào thì ngầm định là tín hiệu **TERM** sẽ được gửi.

- **-s** : xác định tín hiệu được gửi. Tín hiệu có thể là số hoặc tên của tín hiệu. Dưới đây là một số tín hiệu hay dùng:

Số	Tên	Ý nghĩa
1	SIGHUP	(**hang up**) đây là tín hiệu được gửi đến tất cả các quá trình đang chạy trước khi **logout** khỏi hệ thống
2	SIGINT	(**interrupt**) đây là tín hiệu được gửi khi nhấn

		CTRL+c
9	SIGKILL	(**kill**) tín hiệu này sẽ dừng quá trình ngay lập tức
15	SIGTERM	tín hiệu này yêu cầu dừng quá trình ngay lập tức, nhưng cho phép chương trình xóa các file tạm.

- **-p** : lệnh **kill** sẽ chỉ đưa ra chỉ số của quá trình mà không gửi một tín hiệu nào.
- **-l** : hiển thị danh sách các tín hiệu mà lệnh **kill** có thể gửi đến các quá trình (các tín hiệu này có trong file **/usr/include/Linux/signal.h**)

Ví dụ,

```
# ps
PID TTY TIME CMD

 2240 pts/2 00:00:00 bash
 2276 pts/2 00:00:00 man
 2277 pts/2 00:00:00 more
 2280 pts/2 00:00:00 sh
 2281 pts/2 00:00:00 sh
 2285 pts/2 00:00:00 less
 2289 pts/2 00:00:00 man
 2291 pts/2 00:00:00 sh
 2292 pts/2 00:00:00 gunzip
 2293 pts/2 00:00:00 less
 2298 pts/2 00:00:00 ps
# kill 2277

PID TTY TIME CMD
2240 pts/2 00:00:00 bash
2276 pts/2 00:00:00 man
2280 pts/2 00:00:00 sh
2281 pts/2 00:00:00 sh
2285 pts/2 00:00:00 less
2289 pts/2 00:00:00 man
2291 pts/2 00:00:00 sh
2292 pts/2 00:00:00 gunzip
2293 pts/2 00:00:00 less
2298 pts/2 00:00:00 ps
```

4.2.4. Cho máy ngừng hoạt động một thời gian với lệnh sleep

Nếu muốn cho máy nghỉ một thời gian mà không muốn tắt vì ngại khởi động lại thì cần dùng lệnh **sleep**. Cú pháp:

```
sleep [tùy-chọn] NUMBER[SUFFIXƯ]
```

- NUMBER: số giây(s) ngừng hoạt động.
- SUFFIX : có thể là giây(s) hoặc phút(m) hoặc giờ hoặc ngày(d)

Các tùy chọn:

- **--help** : hiện thị trợ giúp và thoát
- **--version** : hiển thị thông tin về phiên bản và thoát

4.2.5. Xem cây quá trình với lệnh pstree

Đã biết lệnh để xem các quá trình đang chạy trên hệ thống, tuy nhiên trong Linux còn có một lệnh cho phép có thể nhìn thấy mức độ phân cấp của các quá trình, đó là lệnh **pstree**. Cú pháp lệnh:

```
pstree [tùy-chọn] [pid | người-dùng]
```

Lệnh **pstree** sẽ hiển thị các quá trình đang chạy dưới dạng cây quá trình. Gốc của cây quá trình thường là init. Nếu đưa ra tên của một người dùng thì cây của các quá trình do người dùng đó sở hữu sẽ được đưa ra.

pstree thường gộp các nhánh quá trình trùng nhau vào trong dấu ngoặc vuông, ví dụ:

```
init -+-getty
      |-getty
      |-getty
      |-getty
```

thành

```
init ---4*[getty]
```

- **-a** : chỉ ra tham số dòng lệnh. Nếu dòng lệnh của một quá trình được tráo đổi ra bên ngoài, nó được đưa vào trong dấu ngoặc đơn.
- **-c** : không thể thu gọn các cây con đồng nhất. Mặc định, các cây con sẽ được thu gọn khi có thể
- **-h** : hiển thị quá trình hiện thời và "tổ tiên" của nó với màu sáng trắng
- **-H** : giống như tùy chọn -h, nhưng quá trình con của quá trình hiện thời không có màu sáng trắng
- **-l** : hiển thị dòng dài.
- **-n** : sắp xếp các quá trình cùng một tổ tiên theo chỉ số quá trình thay cho sắp xếp theo tên

Ví dụ,

```
# pstree
init-+-apmd
 |-atd
 |-automount
 |-crond
 |-enlightenment
 |-gdm-+-X
 |  `-gdm---gnome-session
 |-gen_util_applet
 |-gmc
 |-gnome-name-serv
 |-gnome-smproxy
 |-gnomepager_appl
 |-gpm
```

```
|-identd---identd---3*[identd]
|-inetd
|-kflushd
|-klogd
|-kpiod
|-kswapd
|-kupdate
|-lockd---rpciod
|-login---bash---mc-+-bash-+-cat
| | |-passwd
| | `-pstree
| `-cons.saver
|-lpd
|-mdrecoveryd
|-5*[mingetty]
|-panel
|-portmap
|-rpc.statd
|-sendmail
|-syslogd
`-xfs
```

4.2.6. Lệnh thiết đặt lại độ ưu tiên của quá trình nice và lệnh renice

Ngoài các lệnh xem và hủy bỏ quá trình, trong Linux còn có hai lệnh liên quan đến độ ưu tiên của quá trình, đó là lệnh **nice** và lệnh **renice**.

Để chạy một chương trình với độ ưu tiên định trước, hãy sử dụng lệnh **nice**.

Cú pháp lệnh:

```
nice [tùy-chọn] [lệnh [tham-số ]... ]
```

Lệnh **nice** sẽ chạy một chương trình (lệnh) theo độ ưu tiên đã sắp xếp. Nếu không có **lệnh**, mức độ ưu tiên hiện tại sẽ hiển thị. Độ ưu tiên được sắp xếp từ -20 (mức ưu tiên cao nhất) đến 19 (mức ưu tiên thấp nhất).

- **-ADJUST** : tăng độ ưu tiên theo ADJUST đầu tiên
- **--help** : hiển thị trang trợ giúp và thoát

Để thay đổi độ ưu tiên của một quá trình đang chạy, hãy sử dụng lệnh **renice**.

Cú pháp lệnh:

```
renice <độ-ưu-tiên> [tùy-chọn]
```

Lệnh **renice** sẽ thay đổi mức độ ưu tiên của một hoặc nhiều quá trình đang chạy.

- **-g** : thay đổi quyền ưu tiên theo nhóm người dùng
- **-p** : thay đổi quyền ưu tiên theo chỉ số của quá trình
- **-u** : thay đổi quyền ưu tiên theo tên người dùng

Ví dụ:

```
# renice +1 987 -u daemon root -p 32
```

lệnh trên sẽ thay đổi mức độ ưu tiên của quá trình có chỉ số là **987** và **32**, và tất cả các quá trình do người dùng **daemon** và **root** sở hữu.

CHƯƠNG 5. QUẢN LÝ TÀI KHOẢN NGƯỜI DÙNG

Chương này cung cấp một số công cụ hữu ích trong Linux để quản lý các tài khoản người dùng trên hệ thống.

5.1 Tài khoản người dùng

Như đã biết, trong hệ điều hành đa người dùng, cần phân biệt người dùng khác nhau do quyền sở hữu các tài nguyên trong hệ thống, chẳng hạn như, mỗi người dùng có quyền hạn với file, quá trình của riêng họ. Điều này vẫn rất quan trọng thậm chí cả khi máy tính chỉ có một người sử dụng tại một thời điểm. Mọi truy cập hệ thống Linux đều thông qua tài khoản người dùng. Vì thế, mỗi người sử dụng được gắn với tên duy nhất (đã được đăng ký) và tên đó được sử dụng để đăng nhập. Tuy nhiên một người dùng thực sự có thể có nhiều tên đăng nhập khác nhau. Tài khoản người dùng có thể hiểu là tất cả các file, các tài nguyên, và các thông tin thuộc về người dùng đó.

Khi cài đặt hệ điều hành Linux, đăng nhập ***root*** sẽ được tự động tạo ra. Đăng nhập này được xem là thuộc về siêu người dùng (người dùng cấp cao, người quản trị), vì khi đăng nhập với tư cách người dùng root, có thể làm bất cứ điều gì muốn trên hệ thống. Tốt nhất chỉ nên đăng nhập **root** khi thực sự cần thiết, và hãy đăng nhập vào hệ thống với tư cách là một người dùng bình thường.

Nội dung chương này giới thiệu các lệnh để tạo một người dùng mới, thay đổi thuộc tính của một người dùng cũng như xóa bỏ một người dùng. Lưu ý, chỉ có thể thực hiện được các lệnh trên nếu có quyền của một siêu người dùng.

5.2 Các lệnh cơ bản quản lý người dùng

Người dùng được quản lý thông qua tên người dùng (thực ra là chỉ số người dùng). Nhân hệ thống quản lý người dùng theo chỉ số, vì việc quản lý theo chỉ số sẽ dễ dàng và nhanh thông qua một cơ sở dữ liệu lưu trữ các thông tin về người dùng. Việc thêm một người dùng mới chỉ có thể thực hiện được nếu đăng nhập với tư cách là siêu người dùng.

Để tạo một người dùng mới, cần phải thêm thông tin về người dùng đó vào trong cơ sở dữ liệu người dùng, và tạo một thư mục cá nhân cho riêng người dùng đó. Điều này rất cần thiết để thiết lập các biến môi trường phù hợp cho người dùng.

Lệnh chính để thêm người dùng trong hệ thống Linux là **useradd** (hoặc **adduser**).

5.2.1 File /etc/passwd

Danh sách người dùng cũng như các thông tin tương ứng được lưu trữ trong file **/etc/passwd**.

Ví dụ dưới đây là nội dung của file **/etc/passwd**:

```
mail:x:8:12:mail:/var/spool/mail:
games:x:12:100:games:/usr/games:
gopher:x:13:30:gopher:/usr/lib/gopher-data:
bien:x:500:0:Nguyen Thanh Bien:/home/bien:/bin/bash
sangnm:x:17:100:Nguyen Minh Sang:/home/sangnm:/bin/bash
lan:x:501:0:Lan GNU:/home/lan:/bin/bash
```

Mỗi dòng trong file tương ứng với bảy trường thông tin của một người dùng, và các trường này được ngăn cách nhau bởi dấu '**:**'. ý nghĩa của các trường thông tin đó lần lượt như sau:

- Tên người dùng (**username**)
- Mật khẩu người dùng (**passwd** - được mã hóa)
- Chỉ số người dùng (**user id**)
- Các chỉ số nhóm người dùng (**group id**)
- Tên đầy đủ hoặc các thông tin khác về tài khoản người dùng (**comment**)
- Thư mục để người dùng đăng nhập
- Shell đăng nhập (chương trình chạy lúc đăng nhập)

Bất kỳ người dùng nào trên hệ thống đều có thể đọc được nội dung file **/etc/passwd**, và có thể đăng nhập với tư cách người dùng khác nếu họ biết được mật khẩu, đây chính là lý do vì sao mật khẩu đăng nhập của người dùng không hiển thị trong nội dung file.

5.2.2 Thêm người dùng với lệnh useradd

Siêu người dùng sử dụng lệnh **useradd** để tạo một người dùng mới hoặc cập nhật ngầm định các thông tin về người dùng.

Cú pháp lệnh:

```
useradd [tùy-chọn] <tên-người-dùng>
useradd -D [tùy-chọn]
```

Nếu không có tùy chọn **-D**, lệnh **useradd** sẽ tạo một tài khoản người dùng mới sử dụng các giá trị được chỉ ra trên dòng lệnh và các giá trị mặc định của hệ thống. Tài khoản người dùng mới sẽ được nhập vào trong các file hệ thống, thư mục cá nhân sẽ được tạo, hay các file khởi tạo được sao chép, điều này tùy thuộc vào các tùy chọn được đưa ra.

Các tùy chọn như sau:

- **-c, comment** : soạn thảo trường thông tin về người dùng.
- **-d, home_dir** : tạo thư mục đăng nhập cho người dùng.
- **-e, expire_date** : thiết đặt thời gian (YYYY-MM-DD) tài khoản người dùng sẽ bị hủy bỏ.
- **-f, inactive_days** : tùy chọn này xác định số ngày trước khi mật khẩu của người dùng hết hiệu lực khi tài khoản bị hủy bỏ. Nếu =0 thì hủy bỏ tài khoản người dùng ngay sau khi mật khẩu hết hiệu lực, =-1 thì ngược lại (mặc định là -1).
- **-g, initial_group** : tùy chọn này xác định tên hoặc số khởi tạo đăng nhập nhóm người dùng. Tên nhóm phải tồn tại, và số của nhóm phải tham chiếu đến một nhóm đã tồn tại. Số nhóm ngầm định là 1.
- **-G, group** : danh sách các nhóm phụ mà người dùng cũng là thành viên thuộc các nhóm đó. Mỗi nhóm sẽ được ngăn cách với nhóm khác bởi dấu **','**, mặc định người dùng sẽ thuộc vào nhóm khởi tạo.
- **-m** : với tùy chọn này, thư mục cá nhân của người dùng sẽ được tạo nếu nó chưa tồn tại.
- **-M** : không tạo thư mục người dùng.
- **-n** : ngầm định khi thêm người dùng, một nhóm cùng tên với người dùng sẽ được tạo. Tùy chọn này sẽ loại bỏ sự ngầm định trên.
- **-p, passwd** : tạo mật khẩu đăng nhập cho người dùng.
- **-s, shell** : thiết lập shell đăng nhập cho người dùng.
- **-u, uid** : thiết đặt chỉ số người dùng, giá trị này phải là duy nhất.
  - Thay đổi các giá trị ngầm định

Khi tùy chọn **-D** được sử dụng, lệnh **useradd** sẽ bỏ qua các giá trị ngầm định và cập nhật các giá trị mới.

- **-b, default_home** : thêm tên người dùng vào cuối thư mục cá nhân để tạo tên thư mục cá nhân mới.
- **-e, default_expire_date** : thay đổi thời hạn hết giá trị của tài khoản người dùng.
- **-f, default_inactive** : xác định thời điểm hết hiệu lực của mật khẩu đăng nhập khi tài khoản người dùng bị xóa bỏ.
- **-g, default_group** : thay đổi chỉ số nhóm người dùng.
- **-s, default_shell** : thay đổi shell đăng nhập.

Ngoài lệnh **useradd**, có thể tạo người dùng mới bằng cách sau:

Soạn thảo file **/etc/passwd** bằng **vipw**. Lệnh **vipw** mở trình soạn thảo trên hệ thống và hiệu chỉnh bản sao tạm của file **/etc/passwd**. Việc sử dụng file tạm và khóa file sẽ có tác dụng như một cơ chế khóa để ngăn việc hai người dùng cùng soạn thảo file một lúc. Lúc đó sẽ thêm dòng thông tin mới về người dùng cần tạo. Hãy cẩn thận trong việc soạn thảo tránh nhầm lẫn. Riêng trường mật khẩu nên để trống và tạo mật khẩu sau. Khi file này được lưu, **vipw** sẽ kiểm tra sự đồng nhất trên file bị thay đổi. Nếu tất cả mọi thứ dường như thích hợp thì có nghĩa là file **/etc/passwd** đã được cập nhật.

Ví dụ: thêm người dùng có tên là **new**, chỉ số người dùng **503**, chỉ số nhóm là **100**, thư mục cá nhân là **/home/new** và shell đăng nhập là shell bash:

```
# vipw
mail:x:8:12:mail:/var/spool/mail:
games:x:12:100:games:/usr/games:
gopher:x:13:30:gopher:/usr/lib/gopher-data:
bien:x:500:0:Nguyen Thanh Bien:/home/bien:/bin/bash
sang:x:17:100:Nguyen Minh Sang:/home/sangnm:/bin/bash
lan:x:501:0:Lan GNU:/home/lan:/bin/bash
new::503:100:them mot nguoi moi:/home/new:/bin/bash
```

Tạo thư mục cá nhân của người dùng mới với lệnh *mkdir*

```
# mkdir /home/new
```

- Sao chép các file từ thư mục **/etc/skel/** (đây là thư mục lưu trữ các file cần thiết cho người dùng) vào file cá nhân vừa tạo
- Thay đổi quyền sở hữu và các quyền truy nhập file **/home/new** với các lệnh **chown** và **chmod**

```
# chown new /home/new
# chmod go=u,go-w /home/new
```

Thiết lập mật khẩu của người dùng với lệnh *passwd*

```
# passwd new
passwd:
```

Sau khi thiết lập mật khẩu cho người dùng ở bước cuối cùng, tài khoản người dùng sẽ làm việc. Nên thiết lập mật khẩu người dùng ở bước cuối cùng, nếu không họ có thể vô tình đăng nhập trong khi đang sao chép các file.

5.2.3 Thay đổi thuộc tính người dùng

Trong Linux có rất nhiều lệnh cho phép thay đổi một số các thuộc tính của tài khoản người dùng như:

- **chfn**: thay đổi thông tin cá nhân của người dùng.
- **chsh**: thay đổi shell đăng nhập.
- **passwd**: thay đổi mật khẩu.

Một số các thuộc tính khác sẽ phải thay đổi bằng tay. Ví dụ, để thay đổi tên người dùng, cần soạn thảo lại trực tiếp trên file **/etc/passwd** (với lệnh **vipw**).

Nhưng có một lệnh tổng quát cho phép có thể thay đổi bất kỳ thông tin nào về tài khoản người dùng, đó là lệnh **usermod**.

Cú pháp lệnh:

```
usermod [tùy-chọn] <tên-đăng-nhập>
```

Lệnh **usermod** sửa đổi các file tài khoản hệ thống theo các thuộc tính được xác định trên dòng lệnh.

Các tùy chọn của lệnh:

- **-c, comment** : thay đổi thông tin cá nhân của tài khoản người dùng.
- **-d, home_dir** : thay đổi thư mục cá nhân của tài khoản người dùng.
- **-e, expire_date** : thay đổi thời điểm hết hạn của tài khoản người dùng (YYYY-MM-DD).
- **-f, inactive_days** : thiết đặt số ngày hết hiệu lực của mật khẩu trước khi tài khoản người dùng hết hạn sử dụng.
- **-g, initial_group** : tùy chọn này thay đổi tên hoặc số khởi tạo đăng nhập nhóm người dùng. Tên nhóm phải tồn tại, và số của nhóm phải tham chiếu đến một nhóm đã tồn tại. Số nhóm ngầm định là 1.
- **-G, group** : thay đổi danh sách các nhóm phụ mà người dùng cũng là thành viên thuộc các nhóm đó. Mỗi nhóm sẽ được ngăn cách với nhóm khác bởi dấu ',' mặc định người dùng sẽ thuộc vào nhóm khởi tạo.
- **-l, login_name** : thay đổi tên đăng nhập của người dùng. Trong một số trường hợp, tên thư mục riêng của người dùng có thể sẽ thay đổi để tham chiếu đến tên đăng nhập mới.
- **-p, passwd** : thay đổi mật khẩu đăng nhập của tài khoản người dùng.
- **-s, shell** : thay đổi shell đăng nhập.
- **-u, uid** : thay đổi chỉ số người dùng.

Lệnh **usermod** không cho phép thay đổi tên của người dùng đang đăng nhập. Phải đảm bảo rằng người dùng đó không thực hiện bất kỳ quá trình nào trong khi lệnh **usermod** đang thực hiện thay đổi các thuộc tính của người dùng đó.

Ví dụ muốn thay đổi tên người dùng **new** thành tên mới là **newuser**, hãy gõ lệnh sau:

```
# usermod -l new newuser
```

5.2.4 Xóa bỏ một người dùng (lệnh userdel)

Để xóa bỏ một người dùng, trước hết phải xóa bỏ mọi thứ có liên quan đến người dùng đó.

Lệnh hay được dùng để xóa bỏ một tài khoản người dùng là lệnh **userdel** với cú pháp:

```
userdel [-r] <tên-người-dùng>
```

Lệnh này sẽ thay đổi nội dung của các file tài khoản hệ thống bằng cách xóa bỏ các thông tin về người dùng được đưa ra trên dòng lệnh. Người dùng này phải thực sự tồn tại. Tuỳ chọn **-r** có ý nghĩa:

- **-r** : các file tồn tại trong thư mục riêng của người dùng cũng như các file nằm trong các thư mục khác có liên quan đến người dùng bị xóa bỏ cùng lúc với thư mục người dùng.

Lệnh **userdel** sẽ không cho phép xóa bỏ người dùng khi họ đang đăng nhập vào hệ thống. Phải hủy bỏ mọi quá trình có liên quan đến người dùng trước khi xoá bỏ người dùng đó.

Ngoài ra cũng có thể xóa bỏ tài khoản của một người dùng bằng cách hiệu chỉnh lại file **/etc/passwd**.

5.3 Các lệnh cơ bản liên quan đến nhóm người dùng

Mỗi người dùng trong hệ thống Linux đều thuộc vào một nhóm người dùng cụ thể. Tất cả những người dùng trong cùng một nhóm có thể cùng truy nhập một trình tiện ích, hoặc đều cần truy cập một thiết bị nào đó như máy in chẳng hạn.

Một người dùng cùng lúc có thể là thành viên của nhiều nhóm khác nhau, tuy nhiên tại một thời điểm, người dùng chỉ thuộc vào một nhóm cụ thể.

Nhóm có thể thiết lập các quyền truy nhập để các thành viên của nhóm đó có thể truy cập thiết bị, file, hệ thống file hoặc toàn bộ máy tính mà những người dùng khác không thuộc nhóm đó không thể truy cập được.

5.3.1 Nhóm người dùng và file /etc/group

Thông tin về nhóm người dùng được lưu trong file **/etc/group**, file này có cách bố trí tương tự như file **/etc/passwd**. Ví dụ nội dung của file **/etc/group** có thể như sau:

```
root:x:0:root
bin:x:1:root,bin,daemon
daemon:x:2:root,bin,daemon
sys:x:3:root,bin,adm
adm:x:4:root,adm,daemon
disk:x:6:root
lp:x:7:daemon,lp
mail:x:12:mail
huyen:x:500:
langnu:x:501:
```

Mỗi dòng trong file có bốn trường được phân cách bởi dấu '**:**'. ý nghĩa của các trường theo thứ tự xuất hiện như sau:

- Tên nhóm người dùng (groupname)
- Mật khẩu nhóm người dùng (**passwd** - được mã hóa), nếu trường này rỗng, tức là nhóm không yêu cầu mật khẩu
- Chỉ số nhóm người dùng (group id)
- Danh sách các người dùng thuộc nhóm đó (users)

5.3.2 Thêm nhóm người dùng

Cho phép hiệu chỉnh thông tin trong file **/etc/group** bằng bất kỳ trình soạn thảo văn bản nào có trên hệ thống của để thêm nhóm người dùng, nhưng cách nhanh nhất là sử dụng lệnh **groupadd**.

Cú pháp lệnh :

```
groupadd [tùy-chọn] <tên-nhóm>
```

Các tuỳ chọn là:

- **-g, gid** : tùy chọn này xác định chỉ số nhóm người dùng, chỉ số này phải là duy nhất. Chỉ số mới phải có giá trị lớn hơn 500 và lớn hơn các chỉ số nhóm đã có trên hệ thống. Giá trị từ 0 đến 499 chỉ dùng cho các nhóm hệ thống.
- **-r** : tùy chọn này được dùng khi muốn thêm một tài khoản hệ thống.
- **-f** : tùy chọn này sẽ bỏ qua việc nhắc nhở, nếu nhóm người dùng đó đã tồn tại, nó sẽ bị ghi đè.

Ví dụ:

- Thêm nhóm người dùng bằng cách soạn thảo file **/etc/group**:

```
installer:x:102:hieu, huy, sang
tiengviet:x:103:minh, long, dung
```

Hai dòng trên sẽ bổ sung hai nhóm người dùng mới cùng danh sách các thành viên trong nhóm: nhóm **installer** với chỉ số nhóm là **102** và các thành viên là các người dùng có tên **hieu**, **huy**, **sang**. Tương tự là nhóm **tiengviet** với chỉ số nhóm là **103** và danh sách các thành viên là **minh**, **long**, **dung**. Đây là hai nhóm (**102**, **103**) người dùng hệ thống.

- Thêm nhóm người dùng mới với lệnh **groupadd**:

```
# groupadd -r installer
```

Lệnh trên sẽ cho phép tạo một nhóm người dùng mới có tên là **installer**, tuy nhiên các thành viên trong nhóm sẽ phải bổ sung bằng cách soạn thảo file **/etc/group**.

5.3.3 Sửa đổi các thuộc tính của một nhóm người dùng (lệnh groupmod)

Trong một số trường hợp cần phải thay đổi một số thông tin về nhóm người dùng bằng lệnh **groupmod** với cú pháp như sau:

```
groupmod [tùy-chọn] <tên-nhóm>
```

Thông tin về các nhóm xác định qua tham số **tên-nhóm** được điều chỉnh.

Các tùy chọn của lệnh:

- **-g, gid** : thay đổi giá trị chỉ số của nhóm người dùng.
- **-n, group_name** : thay đổi tên nhóm người dùng.

5.3.4 Xóa một nhóm người dùng (lệnh groupdel)

Nếu không muốn một nhóm nào đó tồn tại nữa thì chỉ việc xóa tên nhóm đó trong file **/etc/group**. Nhưng phải lưu ý rằng, chỉ xóa được một nhóm khi không có người dùng nào thuộc nhóm đó nữa.

Ngoài ra có thể sử dụng lệnh **groupdel** để xóa một nhóm người dùng.

Cú pháp lệnh:

```
groupdel <tên-nhóm>
```

Lệnh này sẽ sửa đổi các file tài khoản hệ thống, xóa tất cả các thực thể liên quan đến nhóm. Tên nhóm phải thực sự tồn tại.

5.4 Các lệnh cơ bản khác có liên quan đến người dùng

Ngoài các lệnh như thêm người dùng, xóa người dùng ..., còn có một số lệnh khác có thể giúp ích rất nhiều nếu đang làm việc trên một hệ thống đa người dùng.

5.4.1 Đăng nhập với tư cách một người dùng khác khi dùng lệnh su

Đôi lúc muốn thực hiện lệnh như một người dùng khác và sử dụng các file hay thiết bị thuộc quyền sở hữu của người dùng đó. Lệnh **su** cho phép thay đổi tên người dùng một cách hiệu quả và cấp cho các quyền truy nhập của người dùng đó.

Cú pháp lệnh:

```
su <người-dùng>
```

Nếu đăng nhập với tư cách người dùng bình thường và muốn trở thành siêu người dùng (root) dùng lệnh sau:

```
# su root
```

Khi đó hệ thống sẽ yêu cầu nhập mật khẩu của siêu người dùng. Nếu cung cấp đúng mật mã, thì sẽ là người dùng **root** cho tới khi dùng lệnh **exit** hoặc **CTRL+d** để đăng xuất ra khỏi tài khoản này và trở về đăng nhập ban đầu. Tương tự, nếu đăng nhập với tư cách **root** và muốn trở thành người dùng bình thường có tên là **newer** thì hãy gõ lệnh sau:

```
# su newer
```

sẽ không bị hỏi về mật khẩu khi thay đổi từ siêu người dùng sang một người dùng khác. Tuy nhiên nếu đăng nhập với tư cách người dùng bình thường và muốn chuyển đổi sang một đăng nhập người dùng khác thì phải cung cấp mật khẩu của người dùng đó.

5.4.2 Xác định người dùng đang đăng nhập (lệnh who)

* Lệnh **who** là một lệnh đơn giản, cho biết được hiện tại có những ai đang đăng nhập trên hệ thống với cú pháp như sau:

```
who [tùy-chọn]
```

Các tuỳ chọn là:

- **-H, --heading** : hiển thị tiêu đề của các cột trong nội dung lệnh.
- **-m** : hiển thị tên máy và tên người dùng với thiết bị vào chuẩn.
- **-q, --count** : hiển thị tên các người dùng đăng nhập và số người dùng đăng nhập.

Ví dụ:

```
# who
root tty1 Nov 15 03:54
lan pts/0 Nov 15 06:07
#
```

Lệnh **who** hiển thị ba cột thông tin cho từng người dùng trên hệ thống. Cột đầu là tên của người dùng, cột thứ hai là tên thiết bị đầu cuối mà người dùng đó đang sử dụng, cột thứ ba hiển thị ngày giờ người dùng đăng nhập.

Ngoài **who**, có thể sử dụng thêm lệnh **users** để xác định được những người đăng nhập trên hệ thống.

Ví dụ:

```
# users
lan root
#
```

* Trong trường hợp người dùng không nhớ nổi tên đăng nhập trong một phiên làm việc (điều này nghe có vẻ như hơi vô lý nhưng là tình huống đôi lúc gặp phải), hãy sử dụng lệnh **whoami** và **who am i**.

Cú pháp lệnh:

```
whoami
```

hoặc

```
who am i
```

Ví dụ:

```
# whoami
lan
#
# who am i
may9!lan pts/0 Nov 15 06:07
#
```

Lệnh **who am i** sẽ hiện kết quả đầy đủ hơn với tên máy đăng nhập, tên người dùng đang đăng nhập, tên thiết bị và ngày giờ đăng nhập.

*** <u>Có một cách khác để xác định thông tin người dùng với lệnh *id*</u>**

Cú pháp lệnh:

```
id [tùy-chọn] [người-dùng]
```

Lệnh này sẽ đưa ra thông tin về người dùng được xác định trên dòng lệnh hoặc thông tin về người dùng hiện thời.

Các tuỳ chọn là:

- **-g, --group** : chỉ hiển thị chỉ số nhóm người dùng.
- **-u, --user** : chỉ hiển thị chỉ số của người dùng.
- **--help** : hiển thị trang trợ giúp và thoát.

Ví dụ:

```
# id
uid=506(lan) gid=503(lan) groups=503(lan)
#
# id -g
503
#
# id -u
506
#
# id root
```

```
uid=0(root)gid=0(root)groups=0(root),1(bin),2(daemon),
3(sys),4(adm),6(disk),10(wheel)
#
```

5.4.3 Xác định các quá trình đang được tiến hành (lệnh w)

Lệnh **w** cho phép xác định được thông tin về các quá trình đang được thực hiện trên hệ thống và những người dùng tiến hành quá trình đó.

Cú pháp lệnh:

```
w [người-dùng]
```

Lệnh **w** đưa ra thông tin về người dùng hiện thời trên hệ thống và quá trình họ đang thực hiện. Nếu chỉ ra người dùng trong lệnh thì chỉ hiện ra các quá trình liên quan đến người dùng đó.

Ví dụ:

```
# w
root tty2 - 2:14pm 13:03 9.30s 9.10s /usr/bin/mc -P
lan pts/1 192.168.2.213 3:20pm 0.00s 0.69s 0.10s w
root pts/2 :0 3:33pm 9:32 0.41s 0.29s /usr/bin/mc –P
```

CHƯƠNG 6. TRUYỀN THÔNG VÀ MẠNG UNIX-LINUX

6.1. Lệnh truyền thông

6.1.1. Lệnh write

Lệnh write được dùng để trao đổi giữa những người hiện đang cùng làm việc trong hệ thống. Thông thường, một người dùng muốn liên hệ với người dùng khác, cần sử dụng lệnh who:

```
$who
```

hiện thông tin như sau:

```
user1     tty17          Oct 15 10:20
user2     tty43          Oct 15 8:25
user4     tty52          Oct 15 12:20
```

trong đó có tên người dùng, số hiệu terminal, ngày giờ vào hệ thống.

Sau đó sử dụng lệnh write để chuyển thông báo cho nhau.

```
$write <tên người dùng>  [<tên trạm cuối>]
```

cần gửi thông báo đến người dùng user1 có tên user2 sẽ gõ:

```
$write user2 tty43
```

- Nếu người dùng user2 hiện không làm việc thì trên màn hình người dùng user1 sẽ hiện ra: "user2 is not logged in" và hiện lại dấu mời shell.
- Nếu người dùng user2 đang làm việc, máy người dùng user2 sẽ phát ra tiếng chuông và trên màn hình hiện ra:

```
Message from user1 on tty17 at <giờ, phút>
```

Cùng lúc đó, tại máy của user1 màn hình trắng để hiện những thông tin gửi tới người dùng user2. Người gửi gõ thông báo của mình theo quy tắc:

- Kết thúc một dòng bằng cụm -o,
- Kết thúc dòng cuối cùng (hết thông báo) bằng cụm -oo.

Để kết thúc kết nối với người dùng user2, người dùng user1 gõ ctrl-d.

Để từ chối mọi việc nhận thông báo từ người khác, sử dụng lệnh không nhận thông báo:

```
$mesg n   (n - no)
```

Một người khác gửi thông báo đến người này sẽ nhận được việc truy nhập không cho phép permission denied.

Để tiếp tục cho phép người khác gửi thông báo đến, sử dụng lệnh:

```
$mesg y   (y - yes)
```

6.1.2. Lệnh mail

Lệnh mail cho phép gửi thư điện tử giữa các người dùng, song hoạt động theo chế độ off-line (gián tiếp). Khi dùng lệnh write để truyền thông cho nhau thì đòi hỏi hai người gửi và nhận đồng thời đang làm việc và cùng chấp nhận cuộc trao đổi đó. Cách thức sử dụng mail là khác hẳn: một trong hai người gửi hoặc nhận có thể không đăng nhập vào hệ thống. Để đảm bào cách thức truyền thông gián tiếp (còn gọi là off-line) như vậy, hệ thống tạo ra cho mỗi người dùng một hộp thư riêng. Khi một người dùng lệnh mail gửi thư đến một người khác thì thư được tự động cho vào hộp thư của người nhận và người nhận sau đó cũng dùng lệnh mail để xem trong hộp thư có thư mới hay không. Không những thế mail còn cho phép sử dụng trên mạng internet (địa chỉ mail thường dưới dạng ***tên-login@máy.mạng.lĩnh-vực.quốc-gia***).

- Lệnh mail chỉ yêu cầu người gửi (hoặc người nhận) login trong hệ thống. Việc nhận và gửi thư được tiến hành từ một người dùng. Thư gửi đi cho người dùng khác, được lưu tại hộp thư của hệ thống.
- Tại thời điểm login hệ thống, người dùng có thể thấy được có thư mới khi trên màn hình xuất hiện dòng thông báo "you have mail".

Lệnh mail trong UNIX gồm 2 chức năng: gửi thư và quản lý thư. Tương ứng, có hai chế độ làm việc với lệnh mail: mode lệnh (command mode) quản trị thư và mode soạn (compose mode) cho phép tạo thư.

a/ Mode soạn

Mode soạn làm việc trực tiếp với một thư và gửi ngay cho người khác. Mode soạn thực chất là sử dụng lệnh mail có tham số:

$mail tên_người_nhận> **Ví dụ, $mail user2**

Lệnh này cho phép soạn và gửi thư cho người nhận có tên được chỉ.

Sau khi gõ lệnh, màn hình bị xóa và con trỏ soạn thảo nhấp nháy ở góc trên, trái để người dùng gõ nội dung thư.

Để kết thúc soạn thư, hãy gõ ctrl-d, màn hình của mail biến mất và dấu mời của shell lại xuất hiện.

Chú ý: Dạng sau đây được dùng để gửi thư đã soạn trong nội dung một file nào đó (chú ý dấu "<" chỉ dẫn thiết bị vào chuẩn là nội dung file thay vì cho bàn phím):

$mail tên_người_nhận < tên_file_nội_dung_thư

Ví dụ, **$ mail user2 < thu1**

Nội dung thư từ File thu1 được gửi cho người nhận user2, dấu mời của shell lại hiện ra.

Cách làm trên đây hay được sử dụng trong gửi / nhận thư điện tử hoặc liên kết truyền thông vì cho phép tiết kiệm được thời gian kết nối vào hệ thống, đặc biệt chi phí phải trả khi kết nối là đáng kể.

b/ Mode lệnh

Như đã nói sử dụng mode lệnh của mail để quản lý hộp thư. Vào mail theo mode lệnh khi dùng lệnh mail không tham số:

$mail

Sau khi gõ lệnh, màn hình mail ở mode lệnh được hiện ra với dấu mời của mode lệnh. (phổ biến là dấu chấm hỏi "?") Tại đây người dùng sử dụng các lệnh của mail quản lý hệ thống thư của mình.

- Cần trợ giúp gõ dấu chấm hỏi (màn hình có hai dấu ??): ? màn hình hiện ra dạng sau:

<số>	Hiện thư số <số>
(dấu cách)	Hiện thư ngay phía trước
+	Hiện thư ngay tiếp theo
l cmd	thực hiện lệnh cmd
dq	xóa thư hiện thời và ra khỏi mail
m user	gửi thư hiện thời cho người dùng
s tên-file	ghi thư hiện thời vào file có tên
r [tên-file]	trả lời thư hiện thời (có thể từ file)
d <số>	xóa thư số
u	khôi phục thư hiện thời
u <số>	khôi phục thư số
m <user> ...	chuyển tiếp thư tới các người dùng khác
q	ra khỏi mail

- Thực hiện các lệnh theo chỉ dẫn trên đây để quản trị được hộp thư của cá nhân.

6.1.3. Lệnh talk

Trong Linux cho phép sử dụng lệnh talk thay thế cho lệnh write.

6.2 Cấu hình Card giao tiếp mạng

Để các máy có thể giao tiếp được với nhau trong mạng theo giao thức TCP/IP, thiết bị dùng làm phương tiện giao tiếp đó là Card giao tiếp mạng (network card). Để quản lý thiết bị này Linux cung cấp lệnh ifconfig. Lệnh này dùng để xem các thông tin về cấu hình mạng hiện thời của máy cũng như gán các địa chỉ cho các card giao tiếp mạng (interface). Ngoài ra ta cũng có thể dùng lệnh này để kích hoạt hoặc tắt một card mạng.

```
/sbin/ifconfig <giao diện> [ <địa chỉ> ] [ arp | -arp][
broadcast <địa chỉ>][netmask <mặt nạ mạng> ]
```

trong đó:

<giao diện>

tên của thiết bị giao tiếp mạng, chẳng hạn eth0 cho card mạng đầu tiên, eth1 cho card mạng thứ hai.

<địa chỉ>

địa chỉ mạng sẽ gán cho giao diện này.

up

tuỳ chọn này sẽ kích hoạt giao diện được chỉ ra.

down

tuỳ chọn này sẽ tắt giao diện được chỉ ra.

arp | -arp

cho phép hay cấm giao thức ARP trên giao diện này.

broadcast <địa chỉ>

xác định địa chỉ quảng bá cho giao diện này.

netmask <mặt nạ mạng>

xác định mặt nạ mạng cho giao diện này.

Để xem cấu hình của máy hiện tại ta dùng lệnh

```
# ifconfig
```

Và ví dụ về kết quả thu được là:

```
eth0 Link encap:Ethernet HWaddr 00:02:55:07:63:07
inet addr:203.113.130.201 Bcast:203.113.130.223 Mask:255.255.255.224
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:3912830 errors:84463 dropped:0 overruns:0 frame:0
TX packets:2402090 errors:0 dropped:0 overruns:0 carrier:0
collisions:84463 txqueuelen:100
RX bytes:2767096664 (2638.9 Mb) TX bytes:1265930467 (1207.2 Mb)
Interrupt:29
```

eth1 Link encap:Ethernet HWaddr 00:05:1C:98:05:B1
inet addr:10.10.0.10 Bcast:10.10.255.255 Mask:255.255.0.0
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:15389731 errors:0 dropped:0 overruns:0 frame:0
TX packets:7768909 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:100
RX bytes:2578998337 (2459.5 Mb) TX bytes:1471928637 (1403.7 Mb)
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:45868 errors:0 dropped:0 overruns:0 frame:0
TX packets:45868 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:5338927 (5.0 Mb) TX bytes:5338927 (5.0 Mb)

Trong trường hợp này ta thấy máy hiện tại có 2 card mạng và được gán các địa chỉ tương ứng như trên.

Muốn chỉ xem các thông tin về một card mạng nào đó thôi ta dùng lệnh:

```
# ifconfig eth0
```

Muốn kích hoạt một card mạng ta dùng lệnh

```
# ifconfig eth0 up
```

Muốn tắt một card mạng ta dùng lệnh

```
# ifconfig eth0 down
```

Muốn đặt lại địa chỉ cho một card mạng ta dùng lệnh:

```
# ifconfig eth0 203.162.9.154 netmask 255.255.255.248
```

Ngoài ra nếu máy tính có cài giao diện GNOME cùng các package quản lý mạng của GNOME thì ta có thể sử dụng lệnh có giao diện đồ hoạ giúp cho việc cấu hình các tham số card mạng dễ dàng hơn. Để có công cụ này ta phải cài đặt package redhat-config-network-xxx.rpm trong đó xxx là số hiệu phiên bản của chương trình.

Trong giao diện đồ hoạ GNOME ta đánh lệnh redhat-config-network, một hộp thoại sẽ hiện lên cho phép ta thay đổi các tham số cho từng card mạng được cài trên máy.

6.3. Các dịch vụ mạng

6.3.1 Hệ thông tin mạng NIS

Khi sử dụng hệ thống mạng nói chung, mục đích của chúng ta là làm cho môi trường mạng trở nên trong suốt đối với người dùng. Một trong những điểm quan trọng là làm cho các dữ liệu quan trọng như là thông tin về người dùng, về các trạm trong mạng là đồng nhất trên tất cả các trạm làm việc. NIS (Network Information System) là một ứng dụng cung cấp các tiện ích truy nhập cơ sở dữ liệu để phân phối thông tin, chẳng hạn như dữ liệu trong /etc/passwd và /etc/group cho tất cả các máy trạm trên mạng. Điều này làm cho mạng trở nên một hệ thống duy nhất. NIS được xây dựng trên việc sử dụng dịch vụ RPC (Remote Procedure Call). Nó bao gồm một thư viện máy chủ, thư viện máy trạm và các công cụ quản trị. Ban đầu NIS được gọi là những trang vàng (Yellow Pages –YP). Cùng với sự phát

triển của NIS mà có sự xuất hiện khác nhau trong các phiên bản. NIS truyền thống được xây dựng trên thư viện libc 4/5. NIS+ là sự mở rộng của NIS song vẫn hỗ trợ bảo mật thông tin. NYS là một phiên bản chuẩn hỗ trợ cả NIS và NIS+.

Hoạt động của NIS

NIS lưu trữ cơ sở dữ liệu về thông tin quản trị mạng trong các file maps. Các file này được đặt trên một NIS server trung tâm, từ đó các NIS client có thể truy nhập đến các thông tin thông qua dịch vụ RPC. Các file maps thường là các file theo định dạng DMB, một dạng cơ sở dữ liệu đơn giản. Các file maps được tạo ra từ các file văn bản như /etc/hosts hay /etc/passwd. Mỗi file văn bản này có thể có nhiều file maps khác nhau tùy thuộc vào khóa của nó. Ví dụ nếu khóa là tên máy trạm thì ta có file hosts.byname, nếu khóa là địa chỉ IP thì ta có file hosts.byname.

File chủ	File maps tương ứng
/etc/hosts	hosts.addr Hosts.byname
/etc/networks	network.byname network.byaddr
/etc/passwd	passwd.byname passwd.byid
/etc/groups	Groups.byname group.byid
/etc/services	service.byname service.bynumber
/etc/rpc	rpc.bynumber rpc.byname
/etc/protocol	protocol.byname protocol.bynumber
/usr/lib/aliases	mail.aliases

Mỗi một file maps có một tên ngắn hơn để đễ nhớ đối với người dùng gọi là các nickname. Để hiển thị danh sách các nickname ta dùng lệnh ***ypcat***:

```
#ypcat -x
Use "ethers" for map "ethers.byname"
Use "aliases" for map "mail.aliases"
Use "services" for map "services.byname"
Use "protocols" for map "protocols.bynumber"
Use "hosts" for map "hosts.byname"
Use "networks" for map "networks.byaddr"
....
```

Các chương trình máy chủ của NIS thường có tên là ypserv. Trong các mạng cỡ nhỏ ta chỉ cần một máy làm máy chủ NIS. Một miền (domain) NIS là một tập hợp các máy trạm được quản lý bởi một máy chủ NIS. Để hiển thị và đặt tên cho một miền ta sử dụng lệnh

```
#domainname nis-domain
```

Tên miền NIS sẽ cho biết máy chủ của miền nào các ứng dụng sẽ truy cập để nhận thông tin cần thiết. Để biết được máy chủ nào trong mạng là NIS server, các chương trình ứng dụng phải hỏi ***ypbind***, một chương trình chạy ngầm có nhiệm vụ phát hiện các NIS server trên mạng. Nó sẽ phát các gói tin quảng bá để tìm các máy chủ NIS trên mạng hoặc sử dụng các thông tin trong các file cấu hình người quản trị đã cung cấp.

Cài đặt và cấu hình cho máy chủ NIS

Với NIS ta có khái niệm máy chủ NIS chính và máy chủ NIS phụ, một miền chỉ có thể có một máy chủ NIS chính. Khi trong mạng có nhiều máy trạm làm việc, một máy chủ NIS có thể bị quá tải, hoặc khi có sự cố thì toàn bộ hệ thống mạng đó sẽ không thể hoạt động được. Các máy chủ NIS phụ sẽ giúp giải quyết vấn đề này. Việc cài đặt các máy chủ NIS phụ chỉ khác máy chủ NIS chính ở chỗ tạo ra các file map. Chúng không được tạo ra bằng *makedbm* mà được lấy về từ máy chủ chính.

Bây giờ ta tìm hiểu cách cài đặt máy chủ NIS chính. Trước tiên ta phải cài đặt phần mềm ***ypserv*** lên máy tính. Chương trình sẽ nằm trong package ypserv-xxx.rpm. Ta có thể cài đặt bằng lệnh:

```
#rpm -ivh ypserv-xxx.rpm
#mkdir /var/yp/nis-domain
```

Để tạo các file cơ sở dữ liệu ta sử dụng chương trình *makedbm*. Do đó phải đảm bảo chương trình đã được cài trên máy, việc tạo lập sẽ được tiến hành thông qua một makefile. Trong file này sẽ chứa các lệnh cần thiết để tạo ra file maps. Sau khi cài đặt phần mềm ta dùng lệnh ***make***:

```
#domainname nis-domain
#cd /var/yp
#make
```

Các file map không tự động cập nhật mỗi khi ta sửa thông tin quản trị. Do vậy mỗi khi có sự thay đổi, ta cần thực hiện lại lệnh *make* để cập nhật sự sửa đổi.

Cài đặt các máy trạm NIS

Trước tiên ta cần cài đặt phần mềm *ypbind* lên máy trạm bằng lệnh:

```
#rpm -ivh ypbind-xxx.rpm
```

Bước tiếp theo là chỉ ra tên của máy chủ và tên miền NIS mà trạm này sẽ sử dụng bằng cách thay đổi thông tin trong file */etc/yp.conf* như sau:

```
#/etc/yp.conf
domainname nis-domain
server lnserver
```

Dòng đầu tiên cho biết máy trạm này thuộc vào miền NIS có tên là ***nis-domain.*** Nếu không có dòng lệnh này thì ta có thể chỉ ra bằng cách đánh lệnh *domainname* tại dấu nhắc dòng lệnh. Dòng thứ 2 chỉ ra tên máy chủ NIS. Địa chỉ IP của tên máy chủ này phải xuất hiện trong file */etc/hosts*. Hoặc ta có thể sử dụng địa chỉ IP ngay trên dòng này.

Khi ta sử dụng máy tính thường xuyên phải thay đổi miền NIS, ta có thể chỉ ra nhiều miền NIS và các máy chủ tương ứng với nó bằng lệnh server. File cấu hình dưới đây cho phép thực hiện điều đó:

```
#yp.conf
server ln-server1 domainname1
server ln-server2 domainname2
```

Khi muốn sử dụng một miền khác thì ta chỉ cần đánh lại lệnh *domainname* để xác định miền ta tương ứng.

Sau khi đã tạo ra các file cấu hình cơ bản, ta nên kiểm tra xem chương trình ypbind đã hoạt động hay chưa. Trước hết, khởi động *ypbind,* sau đó dùng tiện ích *ypcat* để lấy thông tin quản lý bởi NIS server. Để xem thông tin về địa chỉ IP của các trạm ta dùng lệnh:

```
#ypbind
#ypcat hosts.byname
```

192.168.50.1 may1
192.168.50.1 may2
....

Nếu ta không nhận được kết quả như trên hoặc ta nhận được một thông báo lỗi "can't bind to servers domain", có nghĩa là hệ thống NIS hoạt động chưa tốt, ta có thể kiểm tra xem tên miền và tên máy chủ trong file yp.conf đã chính xác chưa và sau đó *ping* máy chủ. Nếu máy chủ đã hoạt động ta kiểm tra xem sự hoạt động của ypserv bằng lệnh rpcinfo:

```
#rpcinfo –u serverhost ypserv
```

program 10004 version 2 ready and waiting

Nếu ta nhận được thông báo như trên là ypserv đang hoạt động tốt.

Lựa chọn các file map

Khi sử dụng NIS ta cần xác định những file cấu hình nào của các máy trạm sẽ được thay thể bởi NIS. Thông thường NIS được sử dụng để tra cứu các thông tin về máy trạm và tài khoản người dùng. Mặc dù ta đã sử dụng NIS như là một hệ quản trị tập trung, hệ thống này vẫn cho phép các máy trạm làm việc được quyền tự do lựa chọn sử dụng các file cấu hình cục bộ hoặc sử dụng từ NIS server. Thứ tự được chỉ ra trong file */etc/nsswitch.conf*.

Ví dụ sau cho biết thứ tự sử dụng dịch vụ của các hàm *gethostbyname()*, *gethostbyaddr()* và *getservbyname()*. Các dịch vụ được liệt kê trước sẽ được sử dụng, nếu không thành công thì sử dụng dịch vụ sau đó.

```
#nsswitch.conf
hosts: nis dns files
services files nis
```

Dưới đây là danh sách các dịch vụ có thể sử dụng trong file */etc/nsswitch.conf*. Các file, chương trình cụ thể được sử dụng sẽ phụ thuộc vào từng loại dịch vụ:

nisplus hay nis+: sử dụng NIS+ server cho miền NIS hiện thời. Tên của server được chỉ ra trong file /etc/nis.conf.

nis: sử dụng NIS server cho domain hiện thời. Tên của server được chỉ ra trong file */etc/yp.conf*. Với thành phần *hosts*, các file map là *hosts.byname* và *hosts.byaddr* sẽ được sử dụng.

dns: sử dụng DNS server, dịch vụ này được sử dụng cho mình thành phần hosts. Tên của máy chủ được đặt trong file */etc/resolv.conf*.

files: sử dụng các file cấu hình cục bộ, ví dụ: /etc/passwd cho thành phần passwd.

dbm: tìm thông tin trong các file cơ sở dữ liệu /var/dbm. Tên của các file là tên của các file map tương ứng của dịch vụ NIS.

Các thành phần được hỗ trợ hiện thời của NYS là: hosts, networks, passwd, group, shadow, services, protocols, rpc, và một số file khác.

Nếu có từ khóa [NOTFOUND=return] trong các thành phần của file *nsswitch.conf,* NIS sẽ thoát ra ngay mà không sử dụng tiếp các dịch vụ sau trong trường hợp nó không tìm thấy thông tin ở dịch vụ trước đó. Chỉ khi nào dịch vụ trước bị lỗi, NIS mới dùng tiếp dịch vụ

sau. Trong ví dụ dưới NIS chỉ sử dụng các file cục bộ khi khởi động hoặc DNS, NIS server bị hỏng.

```
#/ect/nsswitch.conf
hosts: nis dns [NOTFOUND=return] files
network: nis [NOTFOUND=return] files
services: file nis
protocol: files nis
rpc: files nis
```

Sử dụng các file map passwd và group

Một trong những ứng dụng chính của NIS là đồng bộ thông tin về các tài khoản của người sẻ dụng trên tất cả các máy trạm trong miền NIS. Khi đó thông tin về người dùng trên trạm được liệt kê một phần nhỏ trong */etc/passwd*, phần còn lại được lưu trong file map *passwd.byname*. Việc chọn nis trong file */etc/nsswitch.conf* chưa đủ để NIS có thể hoạt động.

Khi sử dụng tài khoản của người dùng được cung cấp bởi NIS, trước tiên phải đảm bảo số hiệu của người dùng trong file *passwd* phải trùng với số hiệu của người dùng đó trên NIS. Nếu số hiệu người dùng, số hiệu nhóm của người dùng đó khác với thông tin trong miền NIS, ta cần sửa lại cho trùng nhau.

Trước tiên ta thay đổi số hiệu người dùng (uid) và số hiệu nhóm (gid) trong đó file */etc/passwd* và */etc/group* trên trạm cục bộ sang các giá trị mới của NIS. Sau đó đổi quyền sở hữu của tất cả các file bằng cách thay đổi số hiệu uid và gid cũ sang uid và gid mới. Giả sử người dùng anhnv có số hiệu uid là 501, thuộc nhóm sinhvien có gid là 423, ta sửa đổi quyền sở hữu như sau:

```
#find / -uid 501 – print > /tmp/uid/uid.501
#find / -gid 501 – print > /tmp/uid/gid.501
#cat /tmp/uid.501 | xargs chown anhnv
#cat /tmp/gid.501 | xargs chgrp sinhvien
```

Sau khi thực hiện các công việc trên số hiệu uid và gid của một người dùng trên máy trạm sẽ đồng nhất với NIS. Bước tiếp theo là sửa đổi file */etc/nsswitch.conf* như sau:

```
#/etc/nsswitch.conf
passwd: nis files
group: nis files
```

File trên quy định lệnh login và các lệnh thuộc họ này sẽ truy vấn NIS server khi một người dùng muốn truy nhập, nếu không tìm thấy nó sẽ tìm tiếp đến các file cục bộ. Thông thường ta loại bỏ hầu hết người dùng khỏi các file cục bộ, chỉ giữ lại root hoặc các tài khoản chung như mail, news,… cho một số tác vụ cần chuyển đổi số hiệu uid sang tên và ngược lại. Ví dụ trong chương trình quản lý công việc cron sử dụng lệnh *su* để tạm trở thành news. Nếu news không có trong */etc/passwd*, chương trình trên sẽ không thực hiện được.

Khi người dùng muốn thay đổi mật khẩu, họ không thể dùng lệnh *passwd* như khi chưa có NIS. Lệnh *passwd* chỉ có tác dụng sửa đổi các file cấu hình cục bộ. NIS cung cấp một công cụ là *yppasswd*, nó không những cho phép sửa đổi mật khẩu người dùng trên NIS mà còn thay đổi các thuộc tính khác như shell… Chương trình này được thực hiện khi khởi động hệ thống bằng cách chạy thêm dịch vụ *rpc.yppasswd*. Vì thói quen người dùng có thể

gõ lệnh *passwd* khi muốn thay đổi mật khẩu. Giải pháp ở đây là thay đổi *passwd* bằng một liên kết đến *yppasswd*.

```
#cd /bin
#mv passwd passwd.old
#ln yppasswd passwd
```

6.4 Hệ thống file trên mạng

Linux có dịch vụ chia sẻ file trên mạng máy tính. Khi ta muốn có khả năng các máy Linux có thể chia sẻ tài nguyên là các file với nhau, dịch vụ NFS sẽ cung cấp khả năng này. Dịch vụ này cho phép chia sẻ file cho các người dùng trên mạng LAN, các file này có khả năng xuất hiện đối với các người dùng như là các file ở trên máy của mình.

6.4.1 Cài đặt NFS

Để cài đặt dịch vụ này ta cần chuẩn bị một package là nfs-utils-xxx.rpm trong đó xxx là số hiệu phiên bản. Đăng nhập với quyền root và sử dụng lệnh:

```
# rpm –ivh nfs-utils-xxx.rpm
```

Nếu không có lời thông báo lỗi thì việc cài đặt đã thành công.

NFS sử dụng thủ tục RPC (Remote Procedure Calls) để gửi và nhận yêu cầu giữa các máy chủ và máy trạm trên mạng, do vậy dịch vụ ánh xạ cổng portmap (dịch vụ quản lý các yêu cầu RPC) phải được khởi động trước. Trên máy chủ NFS dự định sẽ chia sẻ các file dữ liệu phải khởi động hai dịch vụ nfs và portmap bằng lệnh:

```
# service nfs start
# service portmap start
```

Để NFS hoạt động thì ta cần phải khởi động các dịch vụ sau:

Portmapper: tiến trình này không làm việc trực tiếp với NFS mà tham gia quản lý các yêu cầu RPC từ máy trạm gửi đến.

Mountd: tiến trình này sẽ ánh xạ các file trên máy chủ tới các thư mục trên máy trạm yêu cầu. Nó sẽ huỷ bỏ ánh xạ này nếu có lệnh umount từ máy trạm.

Nfs: là tiến trình chính thực hiện các nhiệm vụ của giao thức NFS. Nó có nhiệm vụ cung cấp cho các máy trạm các thư mục hoặc file được yêu cầu.

Ta có thể kiểm tra các thông tin về các dịch vụ NFS bằng lệnh:

```
#rpcinfo -p
```

Ta sẽ thu được kết quả:

```
program vers proto port
100000 2 tcp 111 portmapper
100000 2 udp 111 portmapper
…
100005 3 udp 1024 mountd
100005 3 tcp 1024 mountd
100003 2 udp 2049 nfs
100003 3 udp 2049 nfs
...
```

6.4.2 Khởi động và dừng NFS

Việc khởi động dịch vụ NFS cũng khá đơn giản và đã được giới thiệu ở trên bằng cách khởi động portmap và nfs.

```
# service nfs start
```

hoặc

```
#/etc/init.d/nfs start
```

Việc dừng (tắt) dịch vụ này cũng khá đơn giản, ta dùng lệnh sau:

```
#service nfs stop
```

hoặc

```
#/etc/init.d/nfs stop
```

Ta có thể đặt cho dịch vụ này được tự động khởi động khi ta khởi động máy tính bằng cách dùng lệnh:

```
#setup
```

Hình Đặt các ứng dụng tự khởi động khi Linux khởi động

Sau đó chọn "System services", tiếp đó ta sẽ nhận được một danh sách các dịch vụ hiện đang có trong hệ thống. Muốn cho dịch vụ nào được tự động khởi động ta chỉ cần chọn dịch vụ đó, ở đây ta chọn dịch vụ có tên nfs. Chọn OK và cuối cùng chọn Quit.

6.4.3 Cấu hình NFS server và Client

Cấu hình nfs ta chỉ cần sửa file /etc/exports, đây là file chứa danh sách các thư mục được chia sẻ cho các máy khác. Nó cũng đồng thời chứa danh sách các máy trạm có quyền được truy cập và quyền truy cập của các máy trạm này. Một chú ý về định dạng của file này như sau: các dòng trống sẽ được bỏ qua, các dòng bắt đầu bằng dấu # được coi là các dòng

chú thích và sẽ được bỏ qua. Các dòng dài quá ta có thể ngắt trên nhiều dòng bằng cách sử dụng dấu ngắt dòng (\).

Cấu trúc của mỗi dòng khai báo trong file này như sau:

Tên thư mục	Danh sách địa chỉ các máy trạm, quyền truy nhập của các máy trạm đó	Danh sách địa chỉ các máy trạm, quyền truy nhập của các máy trạm đó
/software/project /software/setup	192.168.0.172(rw) 192.168.0.0/28(ro)	192.168.0.127(ro)

Trong đó, các tham số có ý nghĩa như sau: tên thư mục là đường dẫn đến thư mục ta muốn chia sẻ cho các máy khác. Danh sách địa chỉ các máy trạm, có thể là địa chỉ IP hoặc tên máy (được liệt kê trong file /etc/hosts). Trong trường hợp muốn liệt kê danh sách các máy có địa chỉ gần kề nhau trong một khoảng nào đó ta có thể có cách viết rút gọn như sau: chẳng hạn ta muốn liệt kê các địa chỉ của máy trạm trong khoảng từ 192.168.0.0 đến các máy trạm có địa chỉ 192.168.0.15 ta chỉ cần viết 192.168.0.0/28. Quyền truy nhập được viết dưới dạng (rw) chỉ quyền đọc và ghi, còn (ro) thì các máy trạm chỉ có quyền đọc trên thư mục đó, (noaccess) cấm các máy trạm truy nhập vào các thư mục con của thư mục chia sẻ. Chú ý, giữa địa chỉ của máy và quyền truy nhập không có dấu cách.

6.4.4 Sử dụng mount

Để đọc dữ liệu trên các thư mục đã được chia sẻ này ta có thể dùng cách sau: ta sẽ dùng lệnh mount, nhưng trong trường hợp này ta phải cần quyền quản trị (root). Cú pháp của lệnh sẽ như sau:

```
#mount <tên_máy_chủ:/tên_thư_mục_chia_sẻ>
</tên_thư_mục_cần_ánh_xạ>
```

Lưu ý, trước khi ra lệnh này ta phải tạo ra thư mục cần ánh xạ (trong trường hợp nó chưa tồn tại).

Ví dụ, để ánh xạ thư mục /software/project trên một máy chủ 192.168.0.33 vào thư mục /mnt/project trên máy hiện tại ta dùng lệnh sau:

```
#mount 192.168.0.33:/software/project /mnt/project
```

Bây giờ thư mục /mnt/project trên máy hiện tại sẽ bình đẳng như các thư mục khác trên máy. Ta có thể sao chép, đọc các file trên thư mục này.

6.4.5 Unmount

Sau khi thực hiện xong các thao tác cần thiết, ta có thể hủy bỏ ánh xạ này bằng lệnh umount như sau:

```
#umount /mnt/project
```

Sau lệnh này thì ta không còn có khả năng thao tác với thư mục trên máy chủ được nữa, nếu muốn ta lại phải ánh xạ lại.

Ngoài ra muốn xem trạng thái hoạt động của dịch vụ nfs ta có thể dùng lệnh:

```
#/etc/init.d/nfs status
```

Nó sẽ hiển thị thông tin về trạng thái hiện tại của dịch vụ này đang chạy hay đã dừng lại.

```
rpc.mountd (pid 936) is running...
nfsd (pid 948 947 946 945 944 943 942 941) is running...
rpc.rquotad (pid 931) is running...
```

6.4.6 Mount tự động qua tệp cấu hình

Bây giờ nếu ta muốn hệ thống sẽ tự động ánh xạ thư mục này khi máy khởi động để cho những người dùng không có quyền quản trị có thể dùng được thì ta có thể sử dụng cách sửa đổi nội dung của file /etc/fstab.

Cũng tương tự như lệnh mount ở trên, trong file /etc/fstab cũng có các trường giống như đã nói ở trên. Mỗi một dòng trong file này sẽ có cấu trúc như sau:

```
<tên_máy_chủ:/đường_dẫn_đến_thư_mục_chia_sẻ>
</đường_dẫn_đến_thư_mục_cục_bộ> nfs
```

tham số nfs chỉ cho hệ điều hành biết kiểu file là nfs. Ví dụ ta có thể thêm dòng

```
192.168.0.33:/software/project /mnt/project nfs
```

vào cuối file /etc/fstab.

CHƯƠNG 7. LẬP TRÌNH SHELL VÀ LẬP TRÌNH C TRÊN LINUX

7.1. Cách thức pipes và các yếu tố cơ bản lập trình trên shell

7.1.1. Cách thức pipes

Trong Linux có một số loại shell, shell ngầm định là bash. Shell cho phép người dùng chạy từng lệnh shell (thực hiện trực tiếp) hoặc dãy lệnh shell (file script) và đặc biệt hơn là theo dạng thông qua ống dẫn (pipe).

- Trong một dòng lệnh của shell có thể thực hiện một danh sách các lệnh tuần tự nhau dạng:

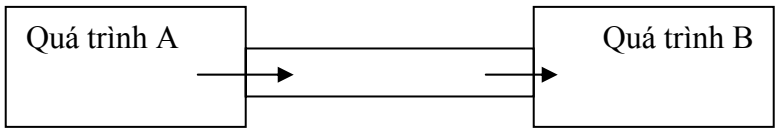
  <lệnh> [; <lệnh>]...

  Như vậy danh sách lệnh là dãy các lệnh liên tiếp nhau, cái sau cách cái trước bởi dấu chấm phảy ";"

  Ví dụ, **$ cal 10 1999; cal 11 1999 ; cal 12 1999**

  Shell cho người dùng cách thức đặc biệt thực hiện các lệnh tuần tự nhau, cái ra của lệnh trước là cái vào của lệnh sau và không phải thông qua nơi lưu trữ trung gian.

- Sử dụng ống dẫn là cách thức đặc biệt trong UNIX và Linux, được thể hiện là một cách thức của shell để truyền thông liên quá trình. ống dẫn được tổ chức theo kiểu cấu trúc dữ liệu dòng xếp hàng "vào trước ra trước" FIFO "First In First Out". Trong cấu trúc dòng xếp hàng, một đầu của dòng nhận phần tử vào và còn đầu kia lại xuất phần tử ra. Trong ngữ cảnh của shell, với hai quá trình A và B được kết nối một ống dẫn được thể hiện như sau:



Như vậy đầu ra của A thông thường hoặc là thiết bị ra chuẩn (màn hình) hoặc là một File (là một tham số của lệnh) được thay bằng "đầu nhập của ống dẫn". Tương tự, đầu vào của B thông thường hoặc là thiết bị vào chuẩn (bàn phím) hoặc là một File (là một tham số của lệnh) được thay bằng "đầu xuất của ống dẫn". Dòng byte lần lượt "chảy" từ quá trình A sang quá trình B.

Mô tả cách thức sử dụng đường ống trong shell như sau:

<lệnh phức hợp> là hoặc **<lệnh>** hoặc **(<lệnh>[;<lệnh>]...)**

Vậy đường ống có dạng

<lệnh phức hợp> | <lệnh phức hợp>

Lệnh phức hợp phía sau có thể không có đối số. Trong trường hợp đó, thông tin kết quả từ lệnh phía trước trở thành thông tin input của lệnh ngay phía sau mà không chịu tác động theo cách thông thường của lệnh trước nữa.

Ví dụ, **$ cal 1999 | more**

Nội dung lịch năm 1999 (lệnh ***cal*** đóng vai trò quá trình A) không được in ngay ra màn hình như thông thường theo tác động của lệnh ***cal*** nữa mà được lưu lên một "file" tạm thời kiểu "ống dẫn" của hệ thống và sau đó trở thành đối số của lệnh ***more*** (lệnh ***more*** đóng vai trò quá trình B).

Trong chương trình, có thể dùng ống dẫn làm file vào chuẩn cho các lệnh đọc tiếp theo. Ví dụ,

```
ls -L | \
```

thì ký hiệu "\" chỉ ra rằng ống dẫn được dùng như file vào chuẩn.

7.1.2. Các yếu tố cơ bản để lập trình trong shell

Shell có công cụ cho phép có thể lập trình trên shell làm tăng thêm độ thân thiện khi giao tiếp với người dùng. Các đối tượng tham gia công cụ như thế có thể được liệt kê:

- Các biến (trong đó chú ý tới các biến chuẩn),
- Các hàm vào - ra
- Các phép toán số học,
- Biểu thức điều kiện,
- Cấu trúc rẽ nhánh,
- Cấu trúc lặp.

a. Một số nội dung trong chương trình shell

- Chương trình là dãy các dòng lệnh shell song được đặt trong một file văn bản (được soạn thảo theo soạn thảo văn bản),
- Các dòng lệnh bắt đầu bằng dấu # chính là dòng chú thích, bị bỏ qua khi shell thực hiện chương trình,
- Thông thường các bộ dịch lệnh shell là sh (/bin/sh) hoặc ksh (/bin/ksh)

Để thực hiện một chương trình shell ta có các cách sau đây:

$sh <<tên chương trình>

hoặc **$sh <tên chương trình>**

hoặc nhờ đổi mod của chương trình:

$chmod u+x <tên chương trình>

và chạy chương trình **$<tên chương trình>**

- Phần lớn các yếu tố ngôn ngữ trong lập trình shell là tương đồng với lập trình C. Trong tài liệu này sử dụng chúng một cách tự nhiên.

b. Các biến trong file script

Trong shell có thể kể tới 3 loại biến:

- Biến môi trường (biến shell đặc biệt, biến từ khóa, biến shell xác định trước hoặc biến shell chuẩn) được liệt kê như sau (các biến này thường gồm các chữ cái hoa):
  - HOME : đường dẫn thư mục riêng của người dùng,
  - MAIL: đường dẫn thư mục chứa hộp thư người dùng,
  - PATH: thư mục dùng để tìm các file thể hiện nội dung lệnh,
  - PS1: dấu mời ban đầu của shell (ngầm định là $),
  - PS2: dấu mời thứ 2 của shell (ngầm định là >),
  - PWD: Thư mục hiện tại người dùng đang làm,
  - SHELL: Đường dẫn của shell (/bin/sh hoặc /bin/ksh)
  - TERM: Số hiệu gán cho trạm cuối,
  - USER: Tên người dùng đã vào hệ thống,

  Trong ***.profile*** ở thư mục riêng của mỗi người dùng thường có các câu lệnh dạng:

  <biến môi trường> = <giá trị>

- Biến người dùng: Các biến này do người dùng đặt tên và có các cánh thức nhận giá trị các biến người dùng từ bàn phím (lệnh read).

Biến được đặt tên gồm một xâu ký tự, quy tắc đặt tên như sau: ký tự đầu tiên phải là một chữ cỏi hoặc dấu gạch chân (_), sau tên là một hay nhiều ký tự khác. Để tạo ra một biến ta chỉ cần gán biến đó một giá trị nào đó. Phép gán là một dấu bằng (=). Ví dụ:

myname=”TriThanh”

Chú ý: không được có dấu cách (space) đằng trước hay đằng sau dấu bằng. Tên biến là phân biệt chữ hoa chữ thường. Để truy xuất đến một biến ta dùng cú pháp sau; $tên_biến. Chẳng hạn ta muốn in ra giá trị của biến myname ở trên ta chỉ cần ra lệnh: ***echo $myname.***

$myname.

$myname.

Một số ví dụ về cách đặt tên biến:

$ no=10 #đây là một cách khai báo hợp lệ

Nhưng cách khai báo dưới đây là không hợp lệ

$ no =10 #có dấu cách sau tên biến

$ no= 10 # có dấu cách sau dấu =

$ no = 10 # có dấu cách cả đằng trước lẫn đằng sau dấu =

Ta có thể khai báo một biến nhưng nó có giá trị NULL như trong những cách sau:

$ vech=

$ vech=""

Nếu ta ra lệnh in giá trị của biến này thì ta sẽ thu được một giá trị NULL ra màn hình (một dòng trống).

- Biến tự động (hay biến-chỉ đọc, tham số vị trí) là các biến do shell đã có sẵn; tên các biến này cho trước. Có 10 biến tự động:

  $0, $1, $2, ..., $9

Tham biến "$0" chứa tên của lệnh, các tham biến thực bắt đầu bằng "$1" (nếu tham số cú vị trí lớn hơn 9, ta phải sử dụng cú pháp ${} – ví dụ, ${10} để thu được các giá trị của chúng). Shell ***bash*** có ba tham biến vị trí đặc biệt, "$#", "$@", và "$#". "$#" là số lượng tham biến vị trí (không tính "$0"). "$*" là một danh sách tất cả các tham biến vị trí loại trừ "$0", đã được định dạng như là một xâu đơn với mỗi tham biến được phân cách bởi kớ tự $IFS. "$@" trả về tất cả các tham biến vị trí được đưa ra dưới dạng N xâu được bao trong dấu ngoặc kép.

Sự khác nhau giữa "$*" và "$@" là gì và tại sao lại có sự phân biệt? Sự khác nhau cho phép ta xử lý các đối số dòng lệnh bằng hai cách. Cách thứ nhất, "$*", do nó là một xâu đơn, nên có thể được biểu diễn linh hoạt hơn không cần yêu cầu nhiều mã shell. "$@" cho phép ta xử lý mỗi đối số riêng biệt bởi vì giá trị của chúng là N đối số độc lập.

Dòng ra (hay dòng vào) tương ứng với các tham số vị trí là các "từ" có trong các dòng đó. Ví dụ,

```
$chay vao chuong trinh roi
```

Nếu chay là một lệnh thì dòng vào này thì:

$0 có giá trị chay $1 có giá trị vao $2 có giá trị chuong

$3 có giá trị trinh $4 có giá trị roi

Một ví dụ khác về biến vị trí giúp ta phân biệt được sự khác nhau giữa biến $* và $@:

```
#!/bin/bash
#testparm.sh
function cntparm
{
     echo –e "inside cntparm $# parms: $*"
```

```
}
cntparm '$*'
cntparm '$@'
echo –e "outside cntparm $* parms\n"
echo –e "outside cntparm $# parms\n"
```

Khi chạy chương trình này ta sẽ thu được kết quả:

```
$./testparm.sh Kurt Roland Wall
```

inside cntparm 1 parms: Kurt Roland Wall
inside cntparm 3 parms: Kurt Roland Wall
outside cntparm: Kurt Roland Wall
outside cntparm: Kurt Roland Wall

Trong dòng thứ nhất và thứ 2 ta thấy kết quả có sự khác nhau, ở dòng thứ nhất biến "$*" trả về tham biến vị trí dưới dạng một xâu đơn, vì thế cntparm báo cáo một tham biến đơn. Dòng thứ hai gọi cntparm, trả về đối số dòng lệnh của là 3 xâu độc lập, vì thế cntparm báo cáo ba tham biến.

c. Các ký tự đặc biệt trong bash

Ký tự	Mô tả
<	Định hướng đầu vào
>	Định hướng đầu ra
(	Bắt đầu subshell
)	Kết thúc subshell
|	Ký hiệu dẫn
\	Dùng để hiện ký tự đặc biệt
&	Thi hành lệnh chạy ở chế độ ngầm
{	Bắt đầu khối lệnh
}	Kết thúc khối lệnh
~	Thư mục home của người dùng hiện tại
`	Thay thế lệnh
;	Chia cắt lệnh
#	Lời chú giải
‘	Trích dẫn mạnh
“	Trích dẫn yếu
$	Biểu thức biến
*	Ký tự đại diện cho chuỗi
?	Ký tự đại diện cho một ký tự

Các ký tự đặc biệt của bash

Dấu chia cắt lệnh, ; , cho phép thực hiện những lệnh bash phức tạp đánh trên một dòng. Nhưng quan trọng hơn, nó là kết thúc lệnh theo lý thuyết POSIX.

Ký tự chú giải, # , khiến bash bỏ qua mọi ký tự từ đó cho đến hết dòng. điểm khác nhau giữa các ký tự trích dẫn mạnh và trích dẫn yếu, ‘ và “, tương ứng là: trích dẫn mạnh bắt bash hiểu tất cả các ký tự theo nghĩa đen; trích dẫn yếu chỉ bảo hộ cho một vài ký tự đặc biệt của bash .

7.2. Một số lệnh lập trình trên shell

7.2.1. Sử dụng các toán tử bash

Các toán tử string

Các toán tử string, cũng được gọi là các toán tử thay thế trong tài liệu về bash, kiểm tra giá trị của biến là chưa gán giá trị hoặc không xác định. Bảng dưới là danh sách các toán tử này cùng với miêu tả cụ thể cho chức năng của từng toán tử.

Toán tử	Chức năng
${var:- word}	Nếu biến tồn tại và xác định thì trả về giá trị của nó, nếu không thì trả về word
${var:= word}	Nếu biến tồn tại và xác định thì trả về giá trị của nó, nếu không thì gán biến thành word, sau đó trả về giá trị của nó
${var:+ word}	Nếu biến tồn tại và xác định thì trả về word, còn không thì trả về null
${var:?message}	Nếu biến tồn tại và xác định thì trả về giá trị của nó, còn không thì hiển thị "bash: $var:$message" và thoát ra khỏi lệnh hay tập lệnh hiện thời.
${var: offset[:length]}	Trả về một xâu con của var bắt đầu tại offset của độ dài length. Nếu length bị bỏ qua, toàn bộ xâu từ offset sẽ được trả về.

Các toán tử string của bash

Để minh hoạ, hãy xem xét một biến shell có tên là status được khởi tạo với giá trị defined. Sử dụng 4 toán tử string đầu tiên cho kết quả status như sau:

$echo ${status:-undefined}
defined
$echo ${status:=undefined}
defined
$echo ${status:+undefined}
undefined
$echo ${status:?Dohhh\! undefined}
defined

Bây giờ sử dụng lệnh unset để xoá biến status, và thực hiện vẫn các lệnh đó, được output như sau:

$unset status
$echo ${status:-undefined}
undefined
$echo ${status:=undefined}
undefined
$echo ${status:+undefined}
undefined
$unset status
$echo ${status:?Dohhh\! undefined}
bash:status Dohhh! Undefined

Cần thiết unset status lần thứ hai vì ở lệnh thứ ba, echo ${status:+undefined}, khởi tạo lại status thành undefined.

Các toán tử substring đã có trong danh sách ở bảng trên đặc biệt có ích. Hãy xét biến foo có giá trị Bilbo_the_Hobbit. Biểu thức ${foo:7} trả về he_Hobbit, trong khi ${foo:7:5} lại trả về he_Ho.

Các toán tử Pattern-Matching

Các toán tử pattern-matching có ích nhất trong công việc với các bản ghi độ dài biến hay các xâu đã được định dạng tự do được định giới bởi các kí tự cố định. Biến môi trường $PATH là một ví dụ. Mặc dù nó có thể khá dài, các thư mục riêng biệt được phân định bởi dấu hai chấm. Bảng dưới là danh sách các toán tử Pattern-Matching của bash và chức năng của chúng.

Toán tử	Chức năng
${var#pattern}	Xoá bỏ phần khớp (match) ngắn nhất của pattern trước var và trả về phần còn lại
${var##pattern}	Xoá bỏ phần khớp (match) dài nhất của pattern trước var và trả về phần còn lại
${var%pattern}	Xoá bỏ phần khớp ngắn nhất của pattern ở cuối var và trả về phần còn lại
${var%%pattern}	Xoá bỏ phần khớp dài nhất của pattern ở cuối var và trả về phần còn lại
${var/pattern/string}	Thay phần khớp dài nhất của pattern trong var bằng string. Chỉ thay phần khớp đầu tiên. Toán tử này chỉ có trong bash 2.0 hay lớn hơn.
${var//pattern/string}	Thay phần khớp dài nhất của pattern trong var bằng string. Thay tất cả các phần khớp. Toán tử này có trong bash 2.0 hoặc lớn hơn.

Các toán tử bash Pattern-Matching

Thông thường quy tắc chuẩn của các toán tử bash pattern-matching là thao tác với file và tên đường dẫn. Ví dụ, giả sử ta có một tên biến shell là mylife có giá trị là /usr/src/linux/Documentation/ide.txt (tài liệu về trình điều khiển đĩa IDE của nhân). Sử dụng mẫu "/*" và "*/" ta có thể tách được tên thư mục và tên file.

```
#!/bin/bash
################################################
myfile=/usr/src/linux/Documentation/ide.txt
echo '${myfile##*/}=' ${myfile##*/}
echo 'basename $myfile =' $(basename $myfile)
echo '${myfile%/*}=' ${myfile%/*}
echo 'dirname $myfile =' $(dirname $myfile)
```

Lệnh thứ 2 xoá xâu matching "*/" dài nhất trong tên file và trả về tên file. Lệnh thứ 4 làm khớp tất cả mọi thứ sau "/", bắt đầu từ cuối biến, bỏ tên file và trả về đường dẫn của file. Kết quả của tập lệnh này là:

$ ./pattern.sh

${myfile##/} = ide.txt*
basename $myfile   = ide.txt
${myfile%/}   = /usr/src/linux/Documentation*
dirname $myfile   = /usr/src/linux/Documentation

Để minh hoạ về các toán tử pattern-matching và thay thế, lệnh thay thế mỗi dấu hai chấm trong biến môi trường $PATH bằng một dòng mới, kết quả hiển thị đường dẫn rất dễ đọc (ví dụ này sẽ sai nếu ta không có bash phiên bản 2.0 hoặc mới hơn):

$ echo –e ${PATH//:/\n}
/usr/local/bin
/bin
/usr/bin
/usr/X11R6/bin
/home/kwall/bin
/home/wall/wp/wpbin

Các toán tử so sánh chuỗi

kiểm tra	Điều kiện thực
str1 = str2	str1 bằng str2
str1 != str2	str1 khác str2
-n str	str có độ dài lớn hơn 0 (khác null)
-z str	str có độ dài bằng 0 (null)

Toán tử sánh chuỗi của bash

Các toán tử so sánh số học

kiểm tra	Điều kiện thực
-eq	bằng
-ge	lớn hơn hoặc bằng
-gt	lớn hơn
-le	nhỏ hơn hoặc bằng
-lt	nhỏ hơn
-ne	khác

Các cách test số nguyên của bash

7.2.2. Điều khiển luồng

Các cấu trúc điều khiển luồng của bash, nó bao gồm:

- if – Thi hành một hoặc nhiều câu lệnh nếu có điều kiện là true hoặc false.
- for – Thi hành một hoặc nhiều câu lệnh trong một số cố định lần.
- while – Thi hành một hoặc nhiều câu lệnh trong khi một điều kiện nào đó là true hoặc false.
- until – Thi hành một hoặc nhiều câu lệnh cho đến khi một điều kiện nào đó trở thành true hoặc false.
- case – Thi hành một hoặc nhiều câu lệnh phụ thuộc vào giá trị của biến.

- select – Thi hành một hoặc nhiều câu lệnh dựa trên một khoảng tuỳ chọn của người dùng.

7.2.2.1 Cấu trúc rẽ nhánh có điều kiện if

Bash cung cấp sự thực hiện có điều kiện lệnh nào đó sử dụng câu lệnh if, câu lệnh if của bash đầy đủ chức năng như của C. Cú pháp của nó được khái quát như sau:

```
if condition
then
        statements
[elif  condition
        statements]
[else
        statements]
fi
```

Đầu tiên, ta cần phải chắc chắn rằng mình hiểu if kiểm tra trạng thái thoát của câu lệnh last trong condition. Nếu nó là 0 (true), sau đó statements sẽ được thi hành, nhưng nếu nó khác 0, thì mệnh đề else sẽ được thi hành và điều khiển nhảy tới dòng đầu tiên của mã fi. Các mệnh đề elif (tuỳ chọn) (có thể nhiều tuỳ ý) sẽ chỉ thi hành khi điều kiện if là false. Tương tự, mệnh đề else (tuỳ chọn) sẽ chỉ thi hành khi tất cả else không thỏa mãn. Nhìn chung, các chương trình Linux trả về 0 nếu thành công hay hoàn toàn bình thường, và khác 0 nếu ngược lại, vì thế không có hạn chế nào cả.

Chú ý: Không phải tất cả chương trình đều tuân theo cùng một chuẩn cho giá trị trả về, vì thế cần kiểm tra tài liệu về các chương trình ta kiểm tra mã thoát với điều kiện if. Ví dụ chương trình diff, trả về 0 nếu không có gì khác nhau, 1 nếu có sự khác biệt và 2 nếu có vấn đề nào đó. Nếu một câu điều kiện hoạt động không như mong đợi thì hãy kiểm tra tài liệu về mã thoát .

Không quan tâm đến cách mà chương trình xác định mã thoát của chúng, bash lấy 0 có nghĩa là true hoặc bình thường còn khác 0 là false. Nếu ta cần cụ thể để kiểm tra một mã thoát của lệnh, sử dụng toán tử ***$?*** ngay sau khi chạy lệnh. ***$?*** trả về mã thoát của lệnh chạy ngay lúc đó.

Phức tạp hơn, bash cho phép ta phối hợp các mã thoát trong phần điều kiện sử dụng các toán tử && và || được gọi là toán tử logic AND và OR. Cú pháp đầy đủ cho toán tử AND như sau:

command1 && command2

Câu lệnh ***command2*** chỉ được chạy khi và chỉ khi ***command1*** trả về trạng thái là số 0 (true).

Cú pháp cho toán tử OR thì như sau:

command1 || command2

Câu lệnh ***command2*** chỉ được chạy khi và chỉ khi ***command1*** trả lại một giá trị khác 0 (false).

Ta có thể kết hợp lại cả 2 loại toán tử lại để có một biểu thức như sau:

command1 && comamnd2 || command3

Nếu câu lệnh ***command1*** chạy thành công thì shell sẽ chạy lệnh ***command2*** và nếu ***command1*** không chạy thành công thì ***command3*** được chạy.

Ví dụ:

$ rm myf && echo "File is removed successfully" || echo "File is not removed"

Nếu file myf được xóa thành công (giá trị trả về của lệnh là 0) thì lệnh "*echo File is removed successfully*" sẽ được thực hiện, nếu không thì lệnh "*echo File is not removed*" được chạy.

Giả sử trước khi ta vào trong một khối mã, ta phải thay đổi một thư mục và copy một file. Có một cách để thực hiện điều này là sử dụng các toán tử ***if*** lồng nhau, như là đoạn mã sau:

```
if cd /home/kwall/data
then
        if cp datafile datafile.bak
        then
                # more code here
        fi
fi
```

Tuy nhiên, bash cho phép ta viết đoạn mã này súc tích hơn nhiều như sau:

```
if cd /home/kwall/data && cp datafile datafile.bak
then
        # more code here
fi
```

Cả hai đoạn mã đều thực hiện cùng một chức năng, nhưng đoạn thứ hai ngắn hơn nhiều, gọn nhẹ và đơn giản. Mặc dù ***if*** chỉ kiểm tra các mã thoát, ta có thể sử dụng cấu trúc ***[…]*** lệnh ***test*** để kiểm tra các điều kiện phức tạp hơn. ***[condition]*** trả về giá trị biểu thị condition là true hay false. ***test*** cũng có tác dụng tương tự.

Một ví dụ khác về cách sử dụng cấu trúc ***if:***

```
#!/bin/sh
# Script to test if..elif...else
#
if [ $1 -gt 0 ]; then
  echo "$1 is positive"
elif [ $1 -lt 0 ]
then
  echo "$1 is negative"
elif [ $1 -eq 0 ]
then
  echo "$1 is zero"
else
  echo "Opps! $1 is not number, give number"
fi
```

Số lượng các phép toán điều kiện của biến hiện tại khoảng 35, khá nhiều và hoàn chỉnh. Ta có thể kiểm tra các thuộc tính file, so sánh các xâu và các biểu thức số học.

Chú ý: Các khoảng trống trước dấu mở ngoặc và sau dấu đóng ngoặc trong ***[condition]*** là cần phải có. Đây là điều kiện cần thiết trong cú pháp shell của bash.

Bảng dưới là danh sách các toán tử ***test*** file phổ biến nhất (danh sách hoàn chỉnh có thể tìm thấy trong những trang manual đầy đủ về bash).

Toán tử	Điều kiện true
-d file	file tồn tại và là một thư mục
-e file	file tồn tại
-f file	file tồn tại và là một file bình thường(không là một thư mục hay một file đặc biệt)
-r file	file cho phép đọc
-s file	file tồn tại và khác rỗng
-w file	file cho phép ghi
-x file	file khả thi hoặc nếu file là một thư mục thì cho phép tìm kiếm trên file
-O file	file của người dùng hiện tại
-G file	file thuộc một trong các nhóm người dùng hiện tại là thành viên
file1 -nt file2	file1 mới hơn file2
file1 -ot file2	file1 cũ hơn file2

Các toán tử test file của bash

Ví dụ chng trình shell cho các toán tử ***test*** file trên các thư mục trong biến $PATH. Mã cho chương trình descpath.sh như sau:

```
#!/bin/bash
###################################

IFS=:
for dir in $PATH;
do
        echo $dir
        if [ -w $dir ]; then
                echo -e "\tYou have write permission in $dir"
        else
                echo –e “\tYou don’t have write permission in $dir”
        fi
        if [ -0 $dir ]; then
                echo -e "\tYou own $dir"
        else
                echo –e “\tYou don’t own $dir”
        fi
        if [ -G $dir ]; then
                echo -e "\tYou are a member of $dir's group"
        else
                echo -e "\tYou aren't a member of $dir's group"
        fi
done
```

Chương trình descpath.sh

Vòng lặp for (giới thiệu trong phần dưới) sẽ duyệt toàn bộ các đường dẫn thư mục trong biến PATH sau đó kiểm tra các thuộc tính của thư mục đó. Kết quả như sau (kết quả có thể khác nhau trên các máy khác nhau do giá trị của biến PATH khác nhau):

/usr/local/bin
You don't have write permission in /usr/local/bin
You don't own /usr/local/bin
You aren't a member of /usr/local/bin's group
/bin
You don't have write permission in /bin
You don't own /bin
You aren't a member of /bin's group
/usr/bin
You don't have write permission in /usr/bin
You don't own /usr/bin
You aren't a member of /usr/bin's group
/usr/X11R6/bin
You don't have write permission in /usr/X11R6/bin
You don't own /usr/X11R6/bin
You aren't a member of /usr/X11R6/bin's group
/home/kwall/bin
You have write permission in /home/kwall/bin
You own /home/kwall/bin
You are a member of /home/kwall/bin's group
/home/kwall/wp/wpbin
You have write permission in /home/kwall/wp/wpbin
You own /home/kwall/wp/wpbin
You are a member of /home/kwall/wp/wpbin's group

Các biểu thức trong phần điều kiện cũng có thể kết hợp với nhau tạo thành các biểu thức phức tạp hơn bằng các phép toán logic. Dưới đây là một bảng các biểu thức logic trong shell.

Toán tử	Ý nghĩa
! expression	Logical NOT
expression1 -a expression2	Logical AND
expression1 -o expression2	Logical OR

7.2.2.2 Các vòng lặp đã quyết định: for

Như đã thấy ở chương trình trên, ***for*** cho phép ta chạy một đoạn mã một số lần nhất định. Tuy nhiên cấu trúc ***for*** của bash chỉ cho phép ta lặp đi lặp lại trong danh sách các giá trị nhất định bởi vì nó không tự động tăng hay giảm con đếm vòng lặp như là C, Pascal, hay Basic. Tuy nhiên vòng lặp ***for*** là công cụ lặp thường xuyên được sử dụng bởi vì nó điều

khiển gọn gàng trên các danh sách, như là các tham số dòng lệnh và các danh sách các file trong thư mục. Cú pháp đầy đủ của for là:

for value in list
do
statements using $value
done

list là một danh sách các giá trị, ví dụ như là tên file. Giá trị là một thành viên danh sách đơn và ***statements*** là các lệnh sử dụng value. Một cú pháp khác của lệnh ***for*** có dạng như sau:

```
for (( expr1; expr2; expr3 ))
do
      .....
                ...
      repeat all statements between do and
      done until expr2 is TRUE
done
```

Linux không có tiện ích để đổi tên hay copy các nhóm của file. Trong MS-DOS nếu ta có 17 file có phần mở rộng a*.doc, ta có thể sử dụng lệnh COPY để copy *.doc thành file *.txt. Lệnh DOS như sau:

C:\ cp doc*.doc doc*.txt

sử dụng vòng lặp for của bash để bù đắp những thiếu sót này. Đoạn mã dưới đây có thể được chuyển thành chương trình shell thực hiện đúng như những gì ta muốn:

for docfile in doc/*.doc
do
cp $docfile ${docfile%.doc}.txt
done

Sử dụng một trong các toán tử pattern-matching của bash, đoạn mã này làm việc copy các file có phần mở rộng là *.doc bằng cách thay thế .doc ở cuối của tên file bằng .txt.

Một ví dụ khác về vòng for đơn giản như sau:

```
#!/bin/bash
for i in 1 2 3 4 5
do
      echo "Welcome $i times"
done
```

Ta cũng có một cấu trúc về ***for*** như sau, chương trình này cũng có cùng chức năng như chương trình trên nhưng ta chú ý đến sự khác biệt về cú pháp của lệnh ***for***.

```
#!/bin/bash
for ((´ i = 0 ;´ i <= 5;´ i++´ ))
do
´ echo "Welcome $i times"
done
```

$ chmod +x for2
$ ./for2
Welcome 0 times

Welcome 1 times
Welcome 2 times
Welcome 3 times
Welcome 4 times
Welcome 5 times

Tiếp theo là một ví dụ về vòng ***for*** lồng nhau:

```
#!/bin/bash

for (( i = 1; i <= 5; i++ ))      ### Outer for loop ###
do

    for (( j = 1 ; j <= 5; j++ )) ### Inner for loop ###
    do
          echo -n "$i "
done
```

Ví dụ khác về cách sử dụng cấu trúc ***if*** và ***for*** như sau:

```
#!/bin/sh
#Script to test for loop
#
#
if [ $# -eq 0 ]
then
echo "Error - Number missing form command line argument"
echo "Syntax : $0 number"
echo "Use to print multiplication table for given number"
exit 1
fi
n=$1
for i in 1 2 3 4 5 6 7 8 9 10
do
echo "$n * $i = `expr $i \* $n`"
done
```

Khi ta chạy chương trình với tham số:

$ chmod 755 mtable
$ ./mtable 7
Ta thu được kết quả như sau:

```
7 * 1 = 7
7 * 2 = 14
...
..
7 * 10 = 70
```

7.2.2.3 Các vòng lặp không xác định: while và until

Vòng lặp ***for*** giới hạn số lần mà một đoạn mã được thi hành, các cấu trúc ***while*** và ***until*** của bash cho phép một đoạn mã được thi hành liên tục cho đến khi một điều kiện nào đó xảy ra. Chỉ với chú ý là đoạn mã này cần viết sao cho điều kiện cuối phải xảy ra nếu không sẽ tạo ra một vòng lặp vô tận. Cú pháp của nó như sau:

while condition
do
    statements
done

Cú pháp này có nghĩa là khi nào condition còn true, thì thực hiện statements cho đến khi condition trở thành false (cho đến khi một chương trình hay một lệnh trả về khác 0):

until condition
do
    statements
done

Cú pháp ***until*** có nghĩa là trái ngược với ***while***: cho đến khi condition trở thành true thì thi hành statements (có nghĩa là cho đến khi một lệnh hay chương trình trả về mã thoát khác 0)

Cấu trúc ***while*** của bash khắc phục thiếu sót không thể tự động tăng, giảm con đếm cua vòng lặp for. Ví dụ, ta muốn copy 150 bản của một file, thì vòng lặp while là một lựa chọn để giải quyết bài toán này. Dưới đây là chương trình:

```
#!/bin/sh
#
declare  -i  idx
idx=1
while [ $idx != 150]
do
        cp somefile  somefile.$idx
        idx=$idx+1
done
```

Chương trình này giới thiệu cách sử dụng tính toán số nguyên của bash. Câu lệnh ***declare*** khởi tạo một biến, idx, định nghĩa là một số nguyên. Mỗi lần lặp idx tăng lên, nó sẽ được kiểm tra để thoát khỏi vòng lặp. Vòng lặp until tuy cũng có khả năng giống while nhưng không được dùng nhiều vì rất khó viết và chạy chậm.

Một ví dụ nữa về cách sử dụng vòng lặp while được minh họa trong chương trình in bản nhân của một số:

```
#!/bin/sh
#Script to test while statement
#
#
if [ $# -eq 0 ]
then
   echo "Error - Number missing form command line argument"
   echo "Syntax : $0 number"
   echo " Use to print multiplication table for given number"
exit 1
fi
```

```
n=$1
i=1
while [ $i -le 10 ]
do
  echo "$n * $i = `expr $i \* $n`"
  i=`expr $i + 1`
done
```

7.2.2.4 Các cấu trúc lựa chọn: case và select

Cấu trúc điều khiển luồng tiếp theo là ***case***, hoạt động cũng tương tự như lệnh switch của C. Nó cho phép ta thực hiện các khối lệnh phụ thuộc vào giá trị của biến. Cú pháp đầy đủ của ***case*** như sau:

```
case expr in
        pattern1 )
        statements ;;
        pattern2 )
        statements ;;
        …
[*)
        statements ;;]
esac
```

expr được đem đi so sánh với từng pattern, nếu nó bằng nhau thì các lệnh tương ứng sẽ được thi hành. Dấu ;; là tương đương với lệnh break của C, tạo ra điều khiển nhảy tới dòng đầu tiên của mã ***esac***. Không như từ khoá ***switch*** của C, lệnh case của bash cho phép ta kiểm tra giá trị của ***expr*** dựa vào pattern, nó có thể chứa các kí tự đại diện. Cách làm việc của cấu trúc case như sau: nó sẽ khớp (match) biểu thức expr với các mẫu pattern1, pattern2,…nếu có một mẫu nào đó khớp thì khối lệnh tương ứng với mẫu đó sẽ được thực thi, sau đó nó thoát ra khỏi lệnh case. Nếu tất cả các mẫu đều không khớp và ta có sử dụng mẫu * (trong nhánh ****)***), ta thấy đây là mẫu có thể khớp với bất kỳ giá trị nào (ký tự đại diện là *), nên các lệnh trong nhánh này sẽ được thực hiện.

Cấu trúc điều khiển ***select*** (không có trong các phiên bản bash nhỏ hơn 1.14) chỉ riêng có trong Korn và các shell bash. Thêm vào đó, nó không có sự tương tự như trong các ngôn ngữ lập trình quy ước. ***select*** cho phép ta dễ dàng trong việc xây dựng các menu đơn giản và đáp ứng các chọn lựa của người dùng. Cú pháp của nó như sau:

```
select  value [in list]
do
        statements that manipulate $value
done
```

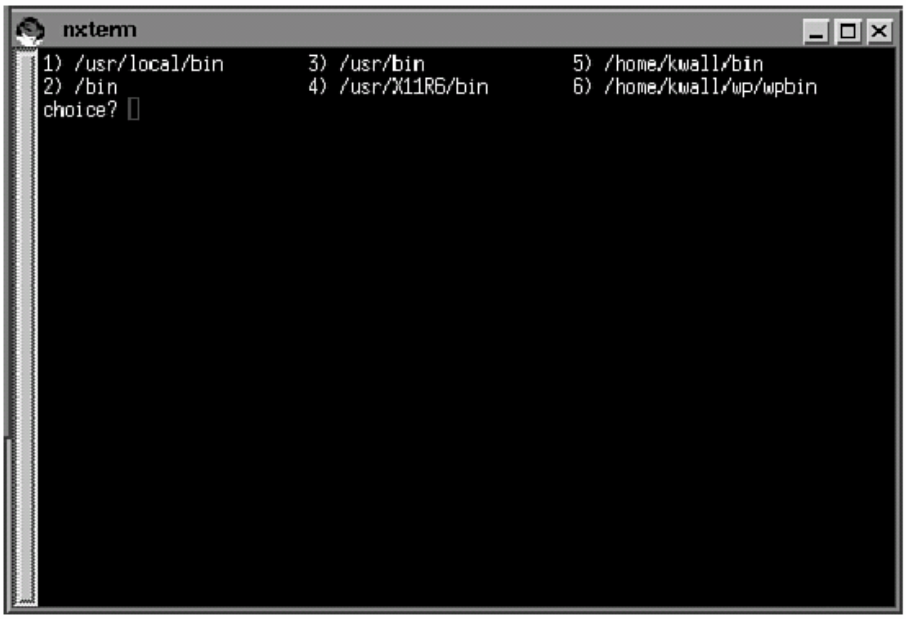
Dưới đây là một ví dự về cách sử dụng lệnh select:

```
#!/bin/bash
# menu.sh – Createing simple menus with select
##########################################

IFS=:
PS3=“choice? ”
```

```
# clear the screen
clear

select dir in $PATH
do
    if [ $dir ]; then
            cnt=$(ls –Al $dir | wc -l)
            echo “$cnt files in $dir”
    else
            echo “Dohhh! No such choice!”
    fi
    echo –e “\nPress ENTER to continue, CTRL –C to quit”
    read
    clear
done
```

Chương trình tạo các menu bằng select

Lệnh đầu tiên đặt kí tự IFS là : (ký tự phân cách), vì thế select có thể phân tích hoàn chỉnh biến môi trường $PATH. Sau đó nó thay đổi lời nhắc default khi ***select*** bằng biến PS3. Sau khi xoá sạch màn hình, nó bước vào một vòng lặp, đưa ra một danh sách các thư mục nằm trong $PATH và nhắc người dùng chọn lựa như là minh hoạ trong hình dưới.



Nếu người dùng chọn hợp lệ, lệnh ***ls*** được thực hiện kết quả được gửi cho lệnh đếm từ ***wc*** để đếm số file trong thư mục và hiển thị kết quả có bao nhiêu file trong thư mục đó. Do ***ls*** có thể sử dụng mà không cần đối số, script đầu tiên cần chắc chắn là $dir khác null (nếu nó là null, ls sẽ hoạt động trên thư mục hiện hành nếu người dùng chọn 1 menu không hợp lệ). Nếu người dùng chọn không hợp lệ, một thông báo lỗi sẽ được hiển thị. Câu lệnh ***read*** (được giới thiệu sau) cho phép người dùng đánh vào lựa chọn của mình và nhấn Enter để lặp lại vòng lặp hay nhấn Ctrl + C để thoát.

Chú ý: Như đã giới thiệu, các vòng lặp script không kết thuc nếu ta không nhấn Ctrl+C. Tuy nhiên ta có thể sử dụng lệnh break để thoát ra.

7.2.2.5 Các hàm shell

Các hàm chức năng của bash là một cách mở rộng các tiện ích sẵn có trong shell, nó có các điểm lợi sau:

- Thi hành nhanh hơn do các hàm shell luôn thường trực trong bộ nhớ.
- Cho phép việc lập trình trở nên dễ dàng hơn vì ta có thể tổ chức chương trình thành các module.

Ta có thể định nghĩa các hàm shell sử dụng theo hai cách:

```
function        fname
{
        commands
}
hoặc là
fname()
{
        commands

}
```

Cả hai dạng đều được chập nhận và không có gì khác giữa chúng. Để gọi một hàm đã định nghĩa đơn giản là gọi tên hàm cùng với các đối số mà nó cần.

Nếu so sánh với C hay Pascal, hàm của bash không được chặt chẽ, nó không kiểm tra lỗi và không có phương thức trả về đối số bằng giá trị. Tuy nhiên giống như C và Pascal, các biến địa phương có thể khai báo cục bộ đối với hàm, do đó tránh được sự xung đột với biến toàn cục. Để thực hiện điều này ta dùng từ khoá local như trong đoạn mã sau:

```
function        foo
{
        local  myvar
        local  yourvar=1
}
```

Trong ví dụ về các biến vị trí ở trên ta cũng thấy được cách sử dụng hàm trong bash. Các hàm shell giúp mã của ta dễ hiểu và dễ bảo dưỡng. Sử dụng các hàm và các chú thích ta sẽ đỡ rất nhiều công sức khi ta phải trở lại nâng cấp đoạn mã mà ta đã viết từ thời gian rất lâu trước đó.

7.2.3 Các toán tử định hướng vào ra

Ta đã được biết về các toán tử định hướng vào ra, > và <. Toán tử định hướng ra cho phép ta gửi kết quả ra của một lệnh vào một file. Ví dụ như lệnh sau:

```
$ cat  $HOME/.bash_profile > out
```

Nó sẽ tạo một file tên là out trong thư mục hiện tại chứa các nội dung của file bash_profile, bằng cách định hướng đầu ra của cat tới file đó.

Tương tự, ta có thể cung cấp đầu vào là một lênh từ một file hoặc là lệnh sử dụng toán tử đầu vào, <. Tacó thể viết lại lệnh cat để sử dụng toán tử định hướng đầu vào như sau:

$ cat < $HOME/.bash_profile > out

Kết quả của lệnh này vẫn như thế nhưng nó cho ta hiểu thêm về cách sử dụng định hướng đầu vào đầu ra.

Toán tử định hướng đầu ra, >, sẽ ghi đè lên bất cứ file nào đang tồn tại. Đôi khi điều này là không mong muốn, vì thế bash cung cấp toán tử nối thêm dữ liệu, >>, cho phép nối thêm dữ liệu vào cuôi file. Hay xem lệnh thêm bí danh cdlpu vào cuối của file .bashrc của tôi:

$echo "alias cdlpu='cd $HOME/kwall/projects/lpu' " >> $HOME/.bashrc

Một cách sử dụng định hướng đầu vào là đầu vào chuẩn (bàn phím). Cú pháp của lệnh này như sau:

Command << label
Input …
Label

Cú pháp này nói lên rằng ***command*** đọc các input cho đến khi nó gặp label. Dưới đây là ví dụ về cách sử dụng cấu trúc này:

```
#!/bin/bash
#####################################

USER=anonymous
PASS=kwall@xmission.com

ftp -i   -n  <<  END
open  ftp.caldera.com
user $USER $PASS
cd  /pub
ls
close
END
```

7.2.4. Hiện dòng văn bản

Lệnh **echo** hiện ra dòng văn bản được ghi ngay trong dòng lệnh có cú pháp:

`echo [tùy chọn] [xâu ký tự]`…

với các tùy chọn như sau:

- **`-n`** : hiện xâu ký tự và dấu nhắc trên cùng một dòng.
- **`-e`** : bật khả năng thông dịch được các ký tự điều khiển.
- **`-E`** : tắt khả năng thông dịch được các ký tự điều khiển.
- **`--help`** : hiện hỗ trợ và thoát. Một số bản Linux không hỗ trợ tham số này.

Ví dụ, dùng lệnh **echo** với tham số **-e**

 # echo -e 'thử dùng lệnh echo \n'

sẽ thấy hiện ra chính dòng văn bản ở lệnh:

```
thử dùng lệnh echo
#
```

ở đây ký tự điều khiển '\n' là ký tự xuống dòng.

7.2.5. Lệnh read độc dữ liệu cho biến người dùng

Lệnh read có dạng read <tên biến>

Ví dụ chương trình shell có tên thu1.arg có nội dung như sau:

```
#!/bin/sh
# Chuong trinh hoi ten nguoi va hien lai
echo "Ten anh la gi?"
read name
echo "Xin chao, $name , anh go $# doi so"
echo "$*"
```

Sau đó, ta thực hiện

```
$chmod u+x thu1.arg
```

và `$thu1.arg Hoi ten nguoi va hien lai`

sẽ thấy xuất hiện

```
Ten anh la gi? Tran Van An
Xin chao, Tran Van An, anh go 6 doi so
Hoi ten nguoi va hien lai
```

7.2.6. Lệnh set

Để gán kết quả đư ra từ lệnh shell ra các biến tự động, ta dùng lệnh set

Dạng lệnh set set `<lệnh>`

Sau lệnh này, kết quả thực hiện lệnh không hiệ ra lê màn hình mà gán kết quả đó tương ứng cho các biến tự động. Một cách tự động các từ trong kết quả thực hiện lệnh sẽ gán tương ứng cho các biến tự động (từ $1 trở đi).

Xem xét một ví dụ sau đây (chương trình thu2.arg) có nội dung:

```
#!/bin/sh
# Hien thoi diem chay chuong trinh nay
set `date`
echo "Thoi gian: $4 $5"
echo "Thu: $1"
echo "Ngay $3 thang $2 nam $6"
```

Sau khi đổi mode của File chương trình này và chạy, chúng ta nhận được:

```
Thoi gian: 7:20:15 EST
Thu: Tue
Ngay 20 thang Oct nam 1998
```

Như vậy,

$# = 6

$* = Tue Oct 20 7:20:15 EST 1998

$1 = Tue $2=Oct $3 = 20 $4 = 7:20:15

$5 = EST $6 = 1998

7.2.7. Tính toán trên các biến

Các tính toán trong shell được thực hiện với các đối số nguyên. Các phép toán gồm có: cộng (+), trừ (-), nhân (*), chia (/), mod (%).

Biểu thức thực hiện theo các phép toán đã nêu.

Tính toán trên shell có dạng:

`expr <biểu thức>`

Ví dụ, chương trình với tên ***cong.shl*** sau đây:

```
#!/bin/sh
# Tinh va in hai so
        tong = `expr $1 + $2`
        echo "Tong = $tong"
```

Sau đó, khi đổi mod và chạy

```
$cong.shl 5 6
```

sẽ hiện ra:

```
Tong = 11
```

7.2.8. Chương trình ví dụ

```
/* Program 5 */
#!/bin/sh
# Chuong trinh liet ke cac thu muc con cua 1 thu muc
# Minh hoa cach su dung if then fi, while do done
# va cac CT test, expr
if test $# -ne 1
then
        echo Cu phap: $0 \<Ten thu muc\>
        exit 1
fi

cd $1                   # Chuyen vao thu muc can list
if test $? -ne 0        # Neu thu muc khong ton tai thi ra khoi CT
then
        exit 1
fi

ls -lL |\
                        # Liet ke ca cac thong tin cua symbolic link
                        # Su dung sub-shell de tu giai phong bien
{
sum=0
# Lenh read x y de bo di dong 'total 1234..' cua lenh ls -lL
read x y ; while read mode link user group size month day hour name
do
        if [ -d $name ]
        then
                echo $name $size \($mode\)
```

```
        fi
done
        }
```

7.3. Lập trình C trên UNIX

7.3.1. Trình biên dịch gcc

Hệ điều hành UNIX luôn kèm theo bộ dịch ngôn ngữ lập trình C với tên gọi là cc (C compiler). Trong Linux, bộ dịch có tên là ***gcc*** (GNU C Compiler) với ngôn ngữ lập trình không khác nhiều với C chuẩn. Nội dung chi tiết về các ngôn ngữ lập trình trên Linux thuộc phạm vi của các tài liệu khác.

gcc cho người lập trình kiểm tra trình biên dịch. Quá trình biên dịch bao gồm bốn giai đoạn:

- Tiền xử lý
- Biên dịch
- Tạp hợp
- Liên kết

Ta có thể dừng quá trình sau một trong những giai đoạn để kiểm tra kết quả biên dịch tại giai đoạn ấy. ***gcc*** cũng có thể chấp nhận ngôn ngữ khác của C, như ANSI C hay C truyền thống. Như đã nói ở trên, ***gcc*** thích hợp biên dịch C++ hay Objective-C. Ta có thể kiểm soát lượng cũng như kiểu thông tin cần debug, tất nhiên là có thể nhúng trong quá trình nhị phân hóa kết quả và giống như hầu hết các trình biên dịch, gcc cũng thực hiện tối ưu hóa mã.

Trước khi bắt đầu đi sâu vào nghiên cứu **gcc**, ta xem một ví dụ sau:

```
#include<stdio.h>
int main (void)
{
        fprintf( stdout, “Hello, Linux programming world!\n”);
        return  0;
 }
```

*Một chương trình điển hình dùng để minh hoạ việc sử dụng **gcc***

Để biên dịch và chạy chương trình này hãy gõ:

$ gcc hello.c –o hello

$ ./hello

Hello, Linux programming world!

Dòng lệnh đầu tiên chỉ cho **gcc** phải biên dịch và liên kết file nguồn **hello.c**, tạo ra tập tin thực thi, bằng cách chỉ định sử dụng đối số **-o hello**. Dòng lệnh thứ hai thực hiện chương trình, và kết quả cho ra trên dòng thứ 3.

Có nhiều chỗ mà ta không nhìn thấy được, **gcc** trước khi chạy **hello.c** thông qua bộ tiền xử lý của **cpp**, để mở rộng bất kỳ một **macro** nào và chèn thêm vào nội dung của những file #include. Tiếp đến, nó biên dịch mã nguồn tiền xử lý sang mã **obj** . Cuối cùng, trình liên kết, tạo ra mã nhị phân cho chương trình **hello**.

Ta có thể tạo lại từng bước này bằng tay, chia thành từng bước qua tiến trình biên dịch. Để chỉ cho **gcc** biết phải dừng việc biên dịch sau khi tiền xử lý, ta sử dụng tuỳ chọn **–E** của **gcc**:

$ gcc –E hello.c –o hello.cpp

Xem xét **hello.cpp** và ta có thể thấy nội dung của **stdio.h** được chèn vào file, cùng với những mã thông báo tiền xử lý khác. Bước tiếp theo là biên dịch **hello.cpp** sang mã **obj**. Sử dụng tuỳ chọn **–c** của **gcc** để hoàn thành:

$ gcc –x cpp-output -c hello.cpp –o hello.o

Trong trường hợp này, ta không cần chỉ định tên của file output bởi vì trình biên dịch tạo một tên file **obj** bằng cách thay thế **.c** bởi **.o**. Tuỳ chọn –x chỉ cho **gcc** biết bắt đầu biên dịch ở bước được chỉ báo trong trường hợp này với mã nguồn tiền xử lý.

Làm thế nào **gcc** biết chia loại đặc biệt của file? Nó dựa vào đuôi mở rộng của file ở trên để xác định rõ phải xử lý file như thế nào cho dúng. Hầu hết những đuôi mở rộng thông thường và chú thích của chúng được liệt kê trong bảng dưới.

Phần mở rộng	Kiểu
.c	Mã nguồn ngôn ngữ C
.c, .cpp	Mã nguồn ngôn ngữ C++
.i	Mã nguồn C tiền xử lý
.ii	Mã nguồn C++ tiền xử lý
.S, .s	Mã nguồn Hợp ngữ
.o	Mã đối tượng biên dịch (obj)
.a, .so	Mã thư viện biên dịch

Các phần mở rộng của tên file ối với ***gcc***

Liên kết file đối tượng, và cuối cùng tạo ra mã nhị phân:

$ gcc hello.o –o hello

Trong trường hợp , ta chỉ muốn tạo ra các file **obj,** và như vậy thì bước liên kết là không cần thiết.

Hầu hết các chương trình C chứa nhiều file nguồn thì mỗi file nguồn đó đều phải được biên dịch sang mã **obj** trước khi tới bước liên kết cuối cùng. Giả sử có một ví dụ, ta đang làm việc trên **killerapp.c** là chương trình sử dụng phần mã của **helper.c**, như vậy để biên dịch **killerapp.c** ta phải dùng dòng lệnh sau:

$ gcc killerapp.c helper.c –o killerapp

gcc qua lần lượt các bước tiền xử lý - biên dịch – liên kết, lúc này tạo ra các file **obj** cho mỗi file nguồn trước khi tạo ra mã nhị phân cho **killerapp**.

Một số tuỳ chọn dòng lệnh của **gcc**:

-o FILE	Chỉ định tên file output; không cần thiết khi biên dịch sang mã obj. Nếu FILE không được chỉ rõ thì tên mặc định sẽ là a.out.
-c	Biên dịch không liên kết.

-DF00=BAR	Định nghĩa macro tiền xử lý đặt tên F00 với một giá trị của BAR trên dòng lệnh.
-IDIRNAME	Trước khi chưa quyết định được DIRNAME hãy tìm kiếm những file include trong danh sách các thư mục(tìm trong danh sách các đường dẫn thư mục)
-LDIRNAME	Trước khi chưa quyết định được DIRNAME hãy tìm kiếm những file thư viện trong danh sách các thư mục. Với mặc định gcc liên kết dựa trên những thư viện dùng chung
-static	Liên kết dựa trên những thư viện tĩnh
-lF00	Liên kết dựa trên libF00
-g	Bao gồm chuẩn gỡ rối thông tin mã nhị phân
-ggdb	Bao gồm tất cả thông tin mã nhị phân mà chỉ có chương trình gỡ rối GNU- gdb mới có thể hiểu được
-O	Tối ưu hoá mã biên dịch
-ON	Chỉ định một mức tối ưu hoá mã N, 0<=N<=3.
-ANSI	Hỗ trợ chuẩn ANSI/ISO của C, loại bỏ những mở rộng của GNU mà xung đột với chuẩn(tuỳ chọn này không bảo đảm mã theo ANSI).
-pedantic	Cho ra tất cả những cảnh báo quy định bởi chuẩn
-pedantic-erors	Thông báo ra tất cả các lỗi quy định bởi chuẩn ANSI/ISO của C.
-traditional	Hỗ trợ cho cú pháp ngôn ngữ C của Kernighan và Ritchie (giống như cú pháp định nghĩa hàm kiểu cũ).
-w	Chặn tất cả thông điệp cảnh báo.
-Wall	Thông báo ra tất cả những cảnh báo hữu ích thông thường mà gcc có thể cung cấp.
-werror	Chuyển đổi tất cả những cảnh báo sang lỗi mà sẽ làm ngưng tiến trình biên dịch.
-MM	Cho ra một danh sách sự phụ thuộc tương thích được tạo.
-v	Hiện ra tất cả các lệnh đã sử dụng trong mỗi bước của tiến trình biên dịch.

7.3.2. Công cụ GNU make

Trong trường hợp ta viết một chương trình rất lớn được cấu thành bởi từ nhiều file, việc biên dịch sẽ rất phức tạp vì phải viết các dòng lệnh gcc rất là dài. Để khắc phục tình trạng này, công cụ GNU make đã được đưa ra. GNU make được giải quyết bằng cách chứa tất cả các dòng lệnh phức tạp đó trong một file gọi là makefile. Nó cũng làm tối ưu hóa quá trình dịch bằng cách phát hiện ra những file nào có thay đổi thì nó mới dịch lại, còn file nào không bị thay đổi thì nó sẽ không làm gì cả, vì vậy thời gian dịch sẽ được rút ngắn.

Một makefile là một cơ sở dữ liệu văn bản chứa cách luật, các luật này sẽ báo cho chương trình make biết phải làm gì và làm như thế nào. Một luật bao gồm các thành phần như sau:

- Đích (target) – cái mà make phải làm
- Một danh sách các thành phần phụ thuộc (dependencies) cần để tạo ra đích
- Một danh sách các câu lệnh để thực thi trên các thành phần phụ thuộc

Khi được gọi, GNU make sẽ tìm các file có tên là GNUmakefile, makefile hay Makefile. Các luật sẽ có cú pháp như sau:

target: dependency1, dependency2, ….
command
command
……

Target thường là một file như file khả thi hay file object ta muốn tạo ra. Dependency là một danh sách các file cần thiết như là đầu vào để tạo ra target. Command là các bước cần thiết (chẳng hạn như gọi chương trình dịch) để tạo ra target.

Dưới đây là một ví dụ về một makefile về tạo ra một chương trình khả thi có tên là editor (số hiệu dòng chỉ đưa vào để tiện theo dõi, còn nội dung của makefile không chứa số hiệu dòng). Chương trình này được tạo ra bởi một số các file nguồn: editor.c, editor.h, keyboard.h, screen.h, screen.c, keyboard.c.

```
1.  editor : editor.o screen.o keyboard.o
2.          gcc -o  editor.o screen.o keyboard.o
3.  editor.o : editor.c editor.h keyboard.h screen.h
4.          gcc -c  editor.c
5.  screen.o : screen.c screen.h
6.          gcc -c  screen.c
7.  keyboard.o : keyboard.c keyboard.h
8.          gcc -c  keyboard.c
9.  clean:
10.          rm *.o
```

Để biên dịch chương trình này ta chỉ cần ra lệnh make trong thư mục chứa file này.

Trong makefile này chứa tất cả 5 luật, luật đầu tiên có đích là ***editor*** được gọi là đích ngầm định. Đây chính là file mà make sẽ phải tạo ra, editor có 3 dependencies editor.o, screen.o, keyboard.o. Tất cả các file này phải tồn tại thì mới tạo ra được đích trên. Dòng thứ 2 là lệnh mà make sẽ gọi thực hiện để tạo ra đích trên. Các dòng tiếp theo là các đích và các lệnh tương ứng để tạo ra các file đối tượng (object).

7.3.3. Làm việc với file

Trong Linux, để làm việc với file ta sử dụng mô tả file (file descriptor). Một trong những thuận lợi trong Linux và các hệ thống UNIX khác là giao diện file làm như nhau đối với nhiều loại thiết bị. Đĩa từ, các thiết bị vào/ra, cổng song song, giả

máy trạm (pseudoterminal), cổng máy in, bảng mạch âm thanh, và chuột được quản lý như các thiết bị đặc biệt giống như các tệp thông thường để lập trình ứng dụng. Các socket TCP/IP và miền, khi kết nối được thiết lập, sử dụng mô tả file như thể chúng là các file chuẩn. Các ống (pipe) cũng tương tự các file chuẩn.

Một mô tả file đơn giản chỉ là một số nguyên được sử dụng như chỉ mục (index) vào một bảng các file mở liên kết với từng tiến trình. Các giá trị 0, 1 và 2 liên quan đến các dòng (streams) vào ra chuẩn: *stdin, stderr* và *stdout*; ba dòng đó thường kết nối với máy của người sử dụng và có thể được chuyển tiếp (redirect).

Một số lời gọi hệ thống sử dụng mô tả file. Hầu hết các lời gọi đó trả về giá trị -1 khi có lỗi xảy ra và biến *errno* ghi mã lỗi. Mã lỗi được ghi trong trang chính tuỳ theo từng lời gọi hệ thống. Hàm *perror()* được sử dụng để hiển thị nội dung thông báo lỗi dựa trên mã lỗi.

Hàm open()

Lời gọi open() sử dụng để mở một file. Khuôn mẫu của hàm và giải thích tham số và cờ của nó được cho dưới đây:

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
int open(const char *pathname, int flags);
int open(const char *pathname, int flags, mode_t mode);
```

Đối số pathname là một xâu chỉ ra đường dẫn đến file sẽ được mở. Thông số thứ ba xác định chế độ của file Unix (các bit được phép) được sử dụng khi tạo một file và nên được sử dụng khi tạo một file . Tham số flags nhận một trong các giá trị O_RDONLY, O_WRONLY hoặc O_RDWR

Cờ	Chú giải
O_RDONLY	Mở file để đọc
O_WRONLY	Mở file để ghi
O_RDWR	Mở file để đọc và ghi
O_CREAT	Tạo file nếu chưa tồn tại file đó
O_EXCL	Thất bại nếu file đã có
O_NOCTTY	Không điều khiển *tty* nếu tty đã mở và tiến trình không điều khiển *tty*
O_TRUNC	Cắt file nếu nó tồn tại
O_APPEND	Nối thêm và con trỏ đặt ở cuối file
O_NONBLOCK	Nếu một quá trình không thể hoàn thành mà không có trễ, trả về trạng thái trước đó
O_NODELAY	Tương tự O_NONBLOCK
O_SYNC	Thao tác sẽ không trả về cho đến khi dữ liệu được ghi vào đĩa hoặc thiết bị khác

Các giá trị cờ của hàm open()

open() trả về một mô tả file nếu không có lỗi xảy ra. Khi có lỗi , nó trả về giá trị -1 và đặt giá trị cho biến errno. Hàm create() cũng tương tự như open() với các cờ O_CREATE | O_WRONLY | O_TRUNC

Hàm close()

Chúng ta nên đóng mô tả file khi đã thao tác xong với nó. Chỉ có một đối số đó là số mô tả file mà lời gọi open() trả về. Dạng của lời gọi close() là:

```
#include <unistd.h>
int close(int fd);
```

Tất cả các khoá (lock) do tiến trình xử lý trên file được giải phóng, cho dù chúng được đặt mô tả file khác. Nếu quá trình đóng file làm cho bộ đếm liên kết bằng 0 thì file sẽ bị xoá. Nếu đây là mô tả file cuối cùng liên kết đến một file được mở thì bản ghi ở bảng file mở được giải phóng. Nếu không phải là một file bình thường thì các hiệu ứng không mong muốn có thể xảy ra.

Hàm read()

Lời gọi hệ thống ***read()*** sử dụng để đọc dữ liệu từ file tương ứng với một mô tả file.

```
#include <unistd.h>
ssize_t read(int fd, void *buf, size_t count);
```

Đối số đầu tiên là mô tả file mà được trả về từ lời gọi ***open()*** trước đó. Đối số thứ hai là một con trỏ tới bộ đệm để sao chép dữ liệu và đối số thứ ba là số byte sẽ được đọc. ***read()*** trả về số byte được đọc hoặc -1 nếu có lỗi xảy ra.

Hàm write()

Lời gọi hệ thống ***write()*** sử dụng để ghi dữ liệu vào file tương ứng với một mô tả file.

```
#include <unistd.h>
ssize_t write(int fd, const void *buf, size_t count);
```

Đối số đầu tiên là số mô tả file được trả về từ lời gọi ***open()*** trước đó. Đối số thứ hai là con trỏ tới bộ đệm (để sao chép dữ liệu, có dung lượng đủ lớn để chứa dữ liệu) và đối số thứ ba xác định số byte sẽ được ghi. write() trả về số byte đọc hoặc -1 nếu có lỗi xảy ra

Hàm ftruncate()

Lời gọi hệ thống ftruncate() cắt file tham chiếu bởi mô tả file fd với độ dài được xác định bởi tham số length

```
#include <unistd.h>
int ftruncate(int fd, size_t length);
```

Trả về giá trị 0 nếu thành công và -1 nếu có lỗi xảy ra.

Hàm lseek()

Hàm lseek() đặt vị trí đọc và ghi hiện tại trong file được tham chiếu bởi mô tả file files tới vị trí offset

```
#include <sys/types.h>
#include <unistd.h>
off_t lseek(int fildes, off_t offset, int whence);
```

Phụ thuộc vào giá trị của whence, giá trị của offset là vị trí bắt đầu (SEEK_SET), vị trí hiện tại (SEEK_CUR), hoặc cuối file (SEEK_END). Giá trị trả về là kết quả của offset: bắt đầu file, hoặc một giá trị của off_t , giá trị -1 nếu có lỗi.

Hàm fstat()

Hàm fstat () đưa ra thông tin về file thông qua việc mô tả các file, nơi kết quả của struct stat được chỉ ra ở con trỏ chỉ đến buf().Kết quả trả về giá trị 0 nếu thành công và nhận giá trị -1 nếu sai (kiểm tra lỗi).

```
#include <sys/stat.h>
#include <unistd.h>
int fstat(int filedes, struct stat *buf);
```

Sau đây là định nghĩa của struct stat:

```
struct stat
{
        dev_t                   st_dev;/ * thiết bị */
        int_t                   st_ino  ;       /* inode */
        mode_t          st_mode;        /* chế độ bảo vệ */
        nlink_t         st_nlink;       /* số lượng các liên kết cứng  */
        uid_t                   st_uid; /* số hiệu của người chủ */
        gid_t                   st_gid; /* số hiệu nhóm của người chủ*/
        dev_t                   st_rdev;        /* kiểu thiết bị */
        off_t                   st_size;/* kích thước bytes */
        unsigned long st_blksize;       /* kích thước khối*/
        unsigned long st_blocks;        /* Số lượng các khối đã sử dụng*/
        time_t                  st_atime;       /* thời gian truy cập cuối cùng*/
        time_t                  st_mtime;       /* thời gian cập nhật cuối cùng */
        time_t                  st_ctime;       /* thời gian thay đổi cuối cùng */
};
```

Hàm fchown()

Lời gọi hệ thống fchown() cho phép tathay đổi người chủ và nhóm người chủ kết hợp với việc mở file.

```
#include <sys/types.h>
#include <unistd.h>
int fchown(int  fd, uid_t  owner, gid_t  group);
```

Tham số đầu tiên là mô tả file, tham số thứ hai là số định danh của người chủ, và tham số thứ ba là số định danh của nhóm người chủ**.** Người dùng hoặc nhóm người dùng sẽ được

phép sử dụng khi giá trị -1 thay đổi. Giá trị trả về là 0 nếu thành công và –1 nếu gặp lỗi (kiểm tra biến errno).

Thông thường người dùng có thể thay đổi nhóm các file thuộc về họ. Chỉ root mới có quyền thay đổi người chủ sở hữu của nhiều nhóm.

Hàm fchdir( )

Lời gọi hàm fchdir( ) thay đổi thư mục bằng cách mở file được mô tả bởi biến fd. Giá trị trả về là 0 nếu thành công và –1 nếu có lỗi (kiểm tra biến errno).

```
#include <unistd.h>
int fchdir(int fd);
```

Một ví dụ về cách sử dụng các hàm thao tác với file:

```
/* filedes_io.c */
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/file.h>
#include <fcntl.h>
#include <unistd.h>

#include < assert.h>
#include <errno.h>
#include <string.h>
#include <stdio.h>  /*for print */
char sample1[] = “This is sample data 1\n”;
char sample2[] = “This is sample data 2\n”;
char data[16];

main ( )
{
        int fd;
        int rc;
        struct   stat      statbuf;

        printf( “Creating file\n”);
        fd = open(“junk.out”, 0_WRONLY | 0_CREAT| 0_TRUNG, 0666);
        assert(fd>=0);

        rc = write(fd, sample1, strlen(sample 1) );
        assert(fd>=0);

        rc = write(fd, sample1, strlen(sample1));
        assert(rc == strlen(sample1));

        close(fd);
```

```
printf(“ Appending to file\n”);
fd = open(“junk.out”, 0_WRONLY| 0_APPEND);
assert(fd>=0);

printf( “ locking file\n”);
rc = flock(fd, LOCK_EX);
assert(rc == 0);

printf(“sleeping for 10 seconds\n”);
sleep(10);

printf(“writing data\n”);
rc = write(fd, sample2, strlen(sample2));
assert(rc == strlen(sample2));

printf(“unlocking file\n”);
rc = flock(fd, LOCK_UN);
assert(rc == 0);

close(fd);

printf(“Reading file \n”);
fd = open(“junk.out”, 0_RDONLY);
assert (fd >=0);

while (1)
{
        rc = read (fd, data, sizeof (data) );
        if( rc > 0 )
        {
                data[rc] =0;     /* kết thúc xâu */
                printf (“ Data read (rc = %d): <%s>\n”, rc, data);
        }
        else if (rc == 0)
        {
                printf (“ End of file read \ n”);
                break;
        } else
        {
                perror ( “ read error ” );
                break;
        }
}
close (fd);

printf (“ Fiddling with inode\n”);
fd = open (“ junk.out”, 0_RDONLY);
```

```
        assert (fd >= 0);

        printf (" changing file mode\n");
        rc = fchmod ( fd, 0600);
        assert (rc == 0);
        if ( getuid ( ) == 0 )     {
                 printf ( " changing file owner \n " );
                rc = fchown (fd, 99, 99);
                assert (rc == 0);
        } else
        {
                printf ( " not changing file owner\n");
        }

        fstat (fd, &statbuf);
        printf (" file mode = 0% ( octal ) \ n", statbuf.st_mode);
        printf("Owner uid = %d \ n", statbuf.st_uid);
        printf(" Owner gid = %d \n", statbuf.st_uid);

        close(fd);
}
```

7.3.4. Thư viện liên kết

Phần này sẽ giới thiệu cách tạo ra và sử dụng thư viện (các module chương trình đã được viết và được tái sử dụng nhiều lần). Thư viện gốc của C/C++ trên Linux chính là *glibc,* thư viện này cung cấp cho ng ời dùng rất nhiều lời gọi hệ thống. Các thư viện trên Linux thường được tổ chức dưới dạng tĩnh (static library), thư viện chia sẻ (shared library) và động (dynamic library - giống như DLL trên MS Windows). Thư viện tĩnh được liên kết cố định vào trong chương trình trong quá trình liên kết. Thư viện dùng chung được nạp vào bộ nhớ trong khi chương trình bắt đầu thực hiện và cho phép các ứng dụng cùng chia sẻ loại thư viện này. Thư viện liên kết động được nạp vào bộ nhớ chỉ khi nào chương trình gọi tới.

7.3.4.1 Thư viện liên kết tĩnh

Thư viện tĩnh và các thư viện dùng chung (shared library) là các file chứa các file được gọi là các module đã được biên dịch và có thể sử dụng lại được. Chúng được lưu trữ dưới một định dạng đặc biệt cùng với một bảng (hoặc một bản đồ) phục vụ cho quá trình liên kết và biên dịch. Các thư viện liên kết tĩnh có phần mở rộng là .a. Để sử dụng các module trong thư viện ta cần thêm phần *#include* file tiêu đề (header) vào trong chương trình nguồn và khi liên kết (sau quá trình biên dịch) thì liên kết với thư viện đó. Dưới đây là một ví dụ về cách tạo và sử dụng một thư viên liên kết tĩnh. Có 2 phần trong ví dụ này, phần thứ nhất là mã nguồn cho thư viện và phần thứ 2 cho chương trình sử dụng thư viện.

```
/*
* liberr.h
*/

#ifndef  _LIBERR_H
#define  _LIBERR_H
#include <stdarg.h>
/* in ra một thông báo lỗi tới việcgọi stderr và return hàm gọi */
void err_quit(const char *fmt, … );
/*  in ra một thông điệp lỗi cho logfile và trả về hàm gọi */
void log_ret(char *logfile, const char *fmt, …);
 /* in ra một thông điệp lỗi cho logfile và thoát */
void log_quit( char *logfile, const char  *fmt , …);
/* in ra một  thông  báo lỗi và trả lại hàm gọi */
void err_prn(const char *fmt, va_list ap, char *logfile);
#endif //_LIBERR_H
```

Mã nguồn cho file liberr.h

```
#include <errno.h>
#include <stdarg.h>
#include <stdlib.h>
#include <stdio.h>
#include  "liberr.h"
#define MAXLINELEN 500
void err_ret(const char *fmt, ...)
{
                va_list ap;
                va_start(ap, fmt);
                err_prn(fmt, ap, NULL);
                va_end(ap);
                return;
}
void err_quit(const char *fmt, ...)
{
                va_list ap;
                va_start(ap, fmt);
                err_prn(fmt, ap, NULL);
                va_end(ap);
                exit(1);
}
void log_ret(char *logfile, const char *fmt, ...)
{
                va_list ap;
                va_start(ap, fmt);
                err_prn(fmt,ap, logfile);
                va_end(ap);
                return;
}
```

```
void log_quit(char *logfile, const char *fmt,... )
{
		va_list ap;
		va_start(ap, fmt);
		err_prn(fmt, ap,logfile);
		va_end(ap);
		exit(1);
}
extern void err_prn( const char *fmt, va_list ap, char *logfile)
{
		int save_err;
		char buf[MAXLINELEN];
		FILE *plf;

		save_err = errno;
		vsprintf(buf,fmt, ap);
		sprintf( buf+strlen(buf),  ": %s", strerror(save_err));
		strcat(buf, "\n");
		fflush(stdout);
		if(logfile !=NULL){
		if((plf=fopen(logfile, "a") ) != NULL){
			fputs(buf, plf);
			fclose(plf);
		}else
			fputs("failed to open log file \n", stderr);
		}else	fputs(buf, stderr);
		fflush(NULL);
		return;
}
```

Mã nguồn file liberr.c

Để tạo một thư viện tĩnh, bước đầu tiên là dịch đoạn mã của form đối tượng:

$gcc –H –c liberr.c –o liberr.o

tiếp theo:

$ar rcs liberr.a liberr.o

```
/*
* errtest.c
*/
#include <stdio.h>
#include <stdlib.h>
#include "liberr.h"
#define ERR_QUIT_SKIP 1
#define LOG_QUIT_SKIP 1

int main(void)
{
```

```
        FILE *pf;

        fputs("Testing err_ret()...\n", stdout);
        if((pf = fopen("foo", "r")) == NULL)
        err_ret("%s %s", "err_ret()", "failed to open foo");

        fputs("Testing log_ret()...\n", stdout);
        if((pf = fopen("foo", "r")) == NULL);
        log_ret("errtest.log", "%s %s", "log_ret()",
        "failed to open foo");

#ifndef ERR_QUIT_SKIP
        fputs("Testing err_quit()...\n", stdout);
        if((pf = fopen("foo", "r")) == NULL)
        err_ret("%s %s", "err_quit()", "failed to open foo");
#endif /* ERR_QUIT_SKIP */

#ifndef LOG_QUIT_SKIP
        fputs("Testing log_quit()...\n", stdout);
        if((pf = fopen("foo", "r")) == NULL)
        log_ret("errtest.log", "%s %s", "log_quit()", "failed to open foo");
#endif /* LOG_QUIT_SKIP */

        return EXIT_SUCCESS;
}
```

Mã nguồn file testerr.c

Biên dịch chương trình kiểm tra, ta sử dụng dòng lệnh:

$ gcc -g errtest.c -o errtest -L. -lerr

Tham số -L. chỉ ra đường dẫn tới thư mục chứa file thư viện là thư mục hiện thời, tham số –lerr chỉ rõ thư viện thích hợp mà chúng ta muốn liên kết. Sau khi dịch ta có thể kiểm tra bằng cách chạy chương trình.

7.3.4.2 Thư viện dùng chung

Thư viện dùng chung có nhiều thuận lợi hơn thư viện tĩnh.Thứ nhất, thư viện dùng chung tốn ít tài nguyên hệ thống, chúng sử dụng ít không gian đĩa vì mã nguồn thư viện dùng chung không biên dịch sang mã nhị phân nhưng được liên kết và được dùng tự động mỗi lần dùng. Chúng sử dụng ít bộ nhớ hệ thống vì nhân chia sẻ bộ nhớ cho thư viện dùng chung này và tất cả các chương trình đều sử dụng chung miền bộ nhớ này. Thứ 2, thư viện dùng chung nhanh hơn vi chúng chỉ cần nạp vào một bộ nhớ. Lí do cuối cùng là mã nguồn trong thư viện dùng chung dễ bảo trì. Khi các lỗi được sửa hay thêm vào các đặc tính, người dùng cần sử dụng thư viện nâng cấp. Đối với thư viện tĩnh, mỗi chương trình khi sử dụng thư viện phải biên dịch lại.

Trình liên kết (linker)/module tải (loader) ***ld.so*** liên kết tên biểu tượng tới thư viện dùng chung mỗi lần chạy. Thư viện dùng chung có tên đặc biệt (gọi là soname), bao gồm tên thư viện và phiên bản chính. Ví dụ: tên đầy đủ của thư viện C trong hệ thống là libc.so.5.4.46, tên thư viện là libc.so, tên phiên bản chính là 5, tên phiên bản phụ là 4, 46 là mức vá (patch

level). Như vậy, soname thư viện C là libc.5. Thư viện libc6 có soname là libc.so.6, sự thay đổi phiên bản chính là sự thay đổi đáng kể thư viện. Phiên bản phụ và patch level thay đổi khi lỗi được sửa nh ng soname không thay đổi và bản mới có sự thay khác biệt đáng kể so với bản cũ.

Các chương trình ứng dụng liên kết dựa vào soname. Tiện ích ***idconfig*** tạo một biểu tượng liên kết từ thư viện chuẩn libc.so.5.4.46 tới soname libc.5 và lưu trữ thông tin này trong /etc/ld.so.cache. Trong lúc chạy, ld.so đọc phần lưu trữ, tìm soname thích hợp và nạp thư viện hiện tai vào bộ nhớ, kết nối hàm ứng dụng gọi tới đối tượng thích hợp trong thư viện.

Các phiên bản thư viện khác nhau nếu:

- Các giao diện hàm đầu ra thay đổi.
- Các giao diện hàm mới được thêm.
- Chức năng hoạt động thay đổi so với đặc tả ban đầu
- Cấu trúc dữ liệu đầu ra thay đổi
- Cấu trúc dữ liệu đầu ra được thêm

Để duy trì tính tương thích của thư viện, cần đảm bảo các yêu cầu:

- Không thêm vào những tên hàm đã có hoặc thay đổi hoạt động của nó
- Chỉ thêm vào cuối cấu trúc dữ liệu đã có hoặc làm cho chúng có tính tuỳ chọn hay được khởi tạo trong thư viện
- Không mở rộng cấu trúc dữ liệu sử dụng trong các mảng

Xây dựng thư viện dùng chung hơi khác so với thư viện tĩnh, quá trình xây dựng thư viện dùng chung được minh hoạ dưới đây:

- Khi biên dịch file đối tượng, sử dụng tùy chọn -fpic của gcc nó sẽ tạo ra mã độc lập vị trí (position independence code) từ đó có thể liên kết hay sử dụng ở bất cứ chỗ nào
- Không loại bỏ file đối tượng và không sử dụng các tùy chọn ***–fomit –frame -pointer*** của ***gcc***, vì nếu không sẽ ảnh hưởng đến quá trình gỡ rối (debug)
- Sử dụng tuỳ chọn ***-shared*** and ***–soname*** của ***gcc***
- Sử dụng tuỳ chọn ***–Wl*** của ***gcc*** để truyền tham số tới trình liên kết ***ld***.
- Thực hiện quá trình liên kết dựa vào thư viện C, sử dụng tuỳ chọn ***–l*** của ***gcc***

Trở lại thư viện xử li lỗi , để tạo thư viện dùng chung trước hết xây dụng file đối tượng:

$ gcc -fPiC -g -c liberr.c -o liberr.o

Tiếp theo liên kết thư viện:

$ gcc -g -shared -Wl,-soname,liberr.so -o liberr.so.1.0.0 liberr.o -lc

Vì không thể cài đặt thư viện này như thư viện hệ thống trong /usr hay /usr/lib chúng ta cần tạo 2 kiên kết, một cho soname:

Và cho trình liên kết khi kết nối dựa vào liberr, sử dụng ***–lerr***:

$ ln -s liberr.so.1.0.0 liberr.so

Bây giờ, để dử dụng thư viện dùng chung mới chúng ta quay lại chương trình kiểm tra, chúng ta cần hướng trình liên kết tới thư viện nào để sử dụng và tìm nó ở đâu, vì vậy chúng ta sẽ sử dụng tuỳ chọn –l và –L:

$ gcc -g errtest.c -o errtest -L. -lerr

Cuối cùng để chạy chương trình, chúng ta cần chỉ cho ld.so nơi để tìm thư viện dùng chung :

$ LD_LIBRARY_PATH=$(pwd) ./errtest

7.3.4.3 Sử dụng đối tượng dùng chung theo cách động

Một cách để sử dụng thư viện dùng chung là nạp chúng tự động mỗi khi chạy không giống như những thư viện liên kết và nạp một cách tự động. Ta có thể sử dụng giao diện dl (dynamic loading) vì nó tạo sự linh hoạt cho lập trình viên hay người dùng.

Giả sử ta đang tạo một ứng dụng sử lý đồ hoạ. Trong ứng dụng, ta biểu diễn dữ liệu ở một dạng không theo chuẩn nhưng lại thuận tiện cho ta xử lý, và ta cần có nhu cầu chuyển dữ liệu đó ra các định dạng thông dụng đã có (số lượng các định dạng này có thể có hàng trăm loại) hoặc đọc dữ liệu từ các định dạng mới này vào để xử lý. Để giải quyết vấn đề này ta có thể sử dụng giải pháp là thư viện. Nhưng khi có thêm một định dạng mới thì ta lại phải biên dịch lại chương trình. Đây lại là một điều không thích hợp lắm. Khả năng sử dụng thư viện động sẽ giúp ta giải quyết vấn đề vừa gặp phải. Giao diện ***dl*** cho phép tạo ra giao diện (các hàm) đọc và viết chung không phụ thuộc vào định dạng của file ảnh. Để thêm hoặc sửa các định dạng của file ảnh ta chỉ cần viết thêm một module để đảm nhận chức năng đó và báo cho chương trình ứng dụng biết là có thêm một module mới bằng cách chỉ cần thay đổi một file cấu hình trong một thư mục xác định nào đó.

Giao diện ***dl*** (cũng đơn thuần được xây dựng như một thư viện - thư viện ***libdl***) chứa các hàm để tải (load), tìm kiếm và giải phóng (unload) các đối tượng chia sẻ. Để sử dụng các hàm này ta thêm file *<dlfcn.h>* vào phần *#include* vào trong mã nguồn, và khi dịch thì liên kết nó với thư viện ***libdl*** bằng cách sử dụng tham số và tên ***–ldl*** trong dòng lệnh dịch.

dl cung cấp 4 hàm xử lí các công việc cần thiết để tải, sử dụng và giải phóng đối tượng dùng chung.

Truy cập đối tượng chia sẻ

Để truy cập một đối tượng chia sẻ, dùng hàm dlopen() có đặc tả như sau:

```
void *dlopen(const char *filename, int flag);
```

dlopen() truy cập đối tượng chia sẻ bằng filename và bằng cờ. Filename có thể là đường dẫn đầy đủ, tên file rút gọn hay NULL. Nếu là NULL ***dlopen()*** mở chương trình đang chạy, đó là chương trình của bạn, nếu filename là đường dẫn ***dlopen()*** mở file đó, nếu là tên rút gọn ***dlopen()*** sẽ tìm trong vị trí sau để tìm file:

$LD_ELF_LiBRARY_PATH,
$LD_LIBRARY_PATH, /etc/ld.so.cache, /usr/lib, và /lib.

Cờ có thể là RTLD_LAZY, có nghĩa là các kí hiệu (symbol) hay tên hàm từ đối tượng truy cập sẽ được tìm mỗi khi chúng được gọi, hoặc cờ có thể là RTLD_NOW, có nghĩa tất cả kí hiệu từ đối tượng truy cập sẽ được tìm trước khi hàm ***dlopen()*** trả về. ***dlopen()*** trả điều khiển tới đối tượng truy nhập nếu nó tìm thấy từ filename hay trả về giá trị NULL nếu không tìm thấy.

Sử dụng đối tượng chia sẻ

Trước khi có thể sử dụng mã nguồn trong thư viện ta phải biết đang tìm cái gì và tìm ở đâu. Hàm dlsym() sẽ giúp điều đó:

```
void *dlsym(void *handle, char *symbol);
```

dlsym() tìm kí hiệu hay tên hàm trong truy cập và trả lại con trỏ kiểu void tới đối tượng hay NULL nếu không thành công.

Kiểm tra lỗi

Hàm ***dlerror()*** sẽ giúp ta kiểm tra lỗi khi sử dụng đối tượng truy cập động:

```
const char *dlerror(void);
```

Nếu một trong các hàm lỗi, ***dlerror()*** trả về thông báo chi tiết lỗi và gán giá trị NULL cho phần bị lỗi.

Giải phóng đối tượng chia sẻ

Để bảo vệ tài nguyên hệ thống đặc biệt bộ nhớ, khi ta sử dụng xong module trong một đối tượng chia sẻ, thì giải phóng chúng. Hàm ***dlclose()*** sẽ đóng đối tượng chia sẻ:

```
int dlclose(void *handle);
```

Sử dụng giao diện dl

Để minh hoạ cách sử dụng ***dl***,chúng ta quay lại thư viện xử lí lỗi, sử dụng một ch ng trình khác như sau:

```
/*
* dltest.c
* Dynamically load liberr.so and call err_ret()
*/
#include <stdio.h>
#include <stdlib.h>
#include <dlfcn.h>
 int main(void)
 {
        void *handle;
        void (*errfcn)();
        const char *errmsg;
        FILE *pf;

        handle = dlopen("liberr.so", RTLD_NOW);
        if(handle == NULL) {
                fprintf(stderr, "Failed to load liberr.so: %s\n", dlerror());
                exit(EXIT_FAILURE);
        }
        dlerror();
        errfcn = dlsym(handle, "err_ret");
        if((errmsg = dlerror()) != NULL) {
                fprintf(stderr, "Didn't find err_ret(): %s\n", errmsg);
                exit(EXIT_FAILURE);
        }
        if((pf = fopen("foobar", "r")) == NULL)
                errfcn("couldn't open foobar");
        dlclose(handle);
        return EXIT_SUCCESS;
 }
```

Mã nguồn chương trình dltest.c

Biên dịch ví dụ trên bằng lệnh:

$ gcc -g -Wall dltest.c -o dltest -ldl

Như tacó thể thấy, chúng ta không liên kết dựa vào liberr hay liberr.h trong mã nguồn. Tất cả truy cập tới liberr.so thông qua ***dl***. Chạy chương trình bằng cách sau:

$ LD_LIBRARY_PATH=$(pwd) ***./dltest***

Nếu thành công thì ta nhận được kết quả như sau:

couldn’t open foobar: No such file or directory

7.3.5 Các công cụ cho thư viện

Công cụ nm

Lệnh ***nm*** liệt kê toàn bộ các tên hàm (symbol) được mã hoá trong file đối tượng (object) và nhị phân (binary). Lệnh nm sử dụng cú pháp sau:

nm [options] file

Lênh nm liệt kê những tên hàm chứa trong file. Bảng dưới liệt kê các tuỳ chọn của lệnh nm:

Tuỳ chọn	Miêu tả
-C| -demangle	Chuyển tên ký tự vào tên mức người dùng để cho dễ đọc.
-s|-print-armap	Khi sử dụng các file lưu trữ (phần mở rộng là ".a"), in ra các chỉ số của module chứa hàm đó.
-u| -undefined-only	Chỉ đưa ra các hàm không được định nghĩa trong file này, tức là các hàm được định nghĩa ở một file khác.
- l | -line-numbers	Sử dụng thông tin gỡ rối để in ra số dòng nơi hàm được định nghĩa.

Các tuỳ chọn của lệnh nm

Công cụ ar

Lệnh ar sử dụng cú pháp sau:

ar {dmpqrtx} [thành viên] file

Lệnh ar tạo, chỉnh sửa và trích các file lưu trữ. Nó thường được sử dụng để tạo các thư viện tĩnh- những file mà chứa một hoặc nhiều file đối tượng chứa các chương trình con thường được sử dụng (subrountine) ở định dạng tiền biên dịch (precompiled format), lệnh ar cũng tạo và duy trì một bảng mà tham chiếu qua tên ký tự tới các thành viên mà trong đó chúng được định nghĩa. Chi tiết của lệnh này đã được trình bày trong chương trước.

Công cụ idd

Lệnh ***nm*** liệt kê các hàm được định nghĩa trong một file đối tượng, nhưng trừ khi ta biết những gì thư viện định nghĩa những hàm nào. Lệnh ***idd*** hữu ích hơn nhiều. ***idd*** liệt kê các thư viện được chia sẻ mà một chương trình yêu cầu để mà chạy. Cú pháp của nó là:

idd [options] file

Lệnh ***idd*** in ra tên của thư viện chia sẻ mà file này sử dụng. Ví dụ: chương trình thư "client mutt" cần 5 thư viện chia sẻ, như được minh hoạ sau đây:

```
$ idd /usr/bin/mutt
        libnsl.so.1 => /lib/libns1.so.1 (0x40019000)
        libslang.so.1 => /usr/lib/libslang.so.1 (0x4002e000)
        libm.so.6 => /lib/libm/so.6 (0x40072000)
        libc.so.6 => /lib/libc.so.6 (0x4008f000)
        /lib/id-linux.so.2 => /lib/id-Linux.so.2 (0x4000000)
```

Tìm hiểu lệnh idconfig

Lệnh ***idconfig*** sử dụng cú pháp sau:

ldconfig [tuỳ chọn] [libs]

Lệnh ***ldconfig*** xác định rõ các liên kết động (liên kết khi chạy) được yêu cầu bởi thư viện được chia sẻ nằm trong các thư mục /usr/lib và /lib. Dưới đây là các tùy chọn của lệnh này:

Các tuỳ chọn	Các miêu tả
-p	Đơn thuần chỉ in ra nội dung của /etc/ld.so.cache, một danh sách hiện thời các thư viện được chia sẻ mà ld.so biết.
-v	Cập nhật /etc/ld.so.cache , liệt kê số phiên bản của mỗi thư viện, quét các thư mục và bất kỳ liên kết mà được tạo ra hoặc cập nhật.

Các tuỳ chọn của hàm idconfig

Biến môi trường và file cấu hình.

Chương trình tải (loader) và trình liên kết (linker) ***ld.so*** sử dụng 2 biến môi trường. Biến thứ nhất là $LD_LIBRARY, chứa danh sách các thư mục chứa các file thư viện được phân cách bởi dấu hai chấm để tìm ra các thư viện cần thiết khi chạy. Nó giống như biến môi trường $PATH. Biến môi trường thứ hai là $LD_PRELOAD, một danh sách các thư viện được người dùng thêm vào được phân cách nhau bởi khoảng trống (space).

ld.so cũng cho phép sử dụng 2 file cấu hình mà có cùng mục đích với biến môi trường được đề cập ở trên. File /etc/ld.so.conf chứa một danh sách các thư mục mà chương trình tải và trình liên kết (loader/linker) nên tìm kiếm các thư viện chia sẻ bên cạnh /usr/lib và /lib. /etc/ld.so.preload chứa danh sách các file thư viện được phân cách bằng một khoảng trống các thư viện này là thư viện người dùng tạo ra.

TÀI LIỆU THAM KHẢO

……

CHÚ THÍCH MỘT SỐ THUẬT NGỮ

Vấn đề chọn thuật ngữ tiếng Việt cho các thuật ngữ gốc tiếng Anh luôn là vấn đề cần được thảo luận để việc chọn lựa như vậy là thích hợp. Dù cho trong tài liệu này, nhiều thuật ngữ tiếng Việt đã được sử dụng một cách phổ biến nhưng tài liệu cũng liệt kê vào bảng chú thích dưới đây với mục đích giúp bạn đọc thuận tiện trong tra cứu và liên hệ. Trong bảng chú thích, mỗi thuật ngữ tiếng Việt dùng trong tài liệu sẽ được kèm theo thuật ngữ tiếng Anh gốc và sau đó có thể là một lời chú thích ngắn. Bảng chú thích là một cố gắng nhỏ nhằm hỗ trợ sử dụng tài liệu này hiệu quả hơn và phạm vi sử dụng của nó giới hạn trong tài liệu.

Tiếng Việt	Tiếng Anh	Giải thích ngắn
File	File	Tập hợp dữ liệu có tổ chức theo một mục đích sử dụng: đoạn văn bản, một chương trình nguồn, một tập hợp dữ liệu ... Còn được gọi là *tệp* hay gọi theo cách giữ nguyên gốc tiếng Anh là *File*.
Tên file	Filename	Tên gọi của file (file thường, thư mục, file đặc biệt ...) được dùng để xác định file.
Thư mục	Directory	Nơi chứa một danh sách các file trong hệ thống File, được Linux coi như một dạng file đặc biệt.
Hệ thống file	File system	Hệ thống toàn bộ các file có trong hệ điều hành Linux (và UNIX nói chung).
Lệnh	Command	Linux cho phép người sử dụng dùng lệnh để trình bày một "thao tác" yêu cầu hệ thống thực hiện.
Tiện ích (còn gọi là lệnh thường trực)	Utility	Lệnh có sẵn trong Linux (trong shell hoặc được đặt tại những thư mục theo quy định của Linux).
Dòng lệnh	Command line	Dòng lệnh bao gồm một hoặc một số lệnh trong một lần yêu cầu của người dùng. Kết thúc dòng lệnh là dấu xuống dòng.
Đăng nhập	Login	Thủ tục bắt đầu một phiên làm việc của một người sử dụng. Login là lệnh cho phép đăng nhập. Thủ tục logout kết thúc phiên làm việc.
Người dùng	User	Người sử dụng Linux cho công việc của mình.
Siêu người dùng (người dùng tối cao/người quản trị)	Superuser	Linux quy định có một siêu người dùng thực hiện có quyền hạn tối cao trong hệ thống bao gồm các lệnh quản trị hệ thống. Với một số lệnh Linux (thêm, bớt người dùng ...) thì chỉ siêu người dùng được phép sử dụng.
Con trỏ	cursor	Điểm đánh dấu trên màn hình đánh dấu vị trí hiện thời để hiện thông tin. Trong trình soạn thảo văn bản, nó là vị trí soạn thảo hiện thời trong văn bản
Hệ điều hành	Operating System (OS)	Bộ chương trình bao gồm các file trên đĩa (băng) từ có chức năng quản lý tài nguyên máy tính và đóng vai trò là "máy tính ảo" đối với người dùng.

Hệ điều hành đa chương trình	Multiprogram-ming OS	Hệ điều hành hoạt động theo cách thức trong bộ nhớ đồng thời có nhiều chương trình được "bình đẳng" trong phân phối tài nguyên (CPU, bộ nhớ).
Hệ điều hành đa người dùng	Multi-users OS	Mỗi người dùng sử dụng một trạm cuối được kết nối máy tính để trực tiếp thực hiện công việc trên máy tính (có đa chương trình).
Nhân	Kernel	Bộ phận cốt lõi nhất của Linux, thường trực để thực hiện các chức năng cơ bản của hệ điều hành (còn được gọi là ***lõi***)
Quá trình	Process	Một lần thực hiện của một chương trình (Còn được gọi là tiến trình). Một số hệ điều hành gọi là bài toán (task).
Chương trình giải thích lệnh	Command Comment Program (CCP)	Một chương trình thuộc hệ điều hành có chức năng tiếp nhận, phân tích dòng lệnh người dùng để lệnh được thực hiện. Trong Linux là shell, trong MS-DOS là COMMAND.COM.
Đăng nhập	Login	Thủ tục mà một người dùng cần phải thực hiện khi bắt đầu phiên làm việc với Linux. Thủ tục này yêu cầu người dùng đưa vào tên kí hiệu và mật khẩu đã được đăng kí trong hệ thống.
Đăng xuất	Logout	Thủ tục mà một người dùng cần phải thực hiện khi kết thúc phiên làm việc với Linux. Trên máy tính cá nhân, sau khi đăng xuất sẽ kéo theo một đăng nhập mới.
Dấu nhắc shell		Còn gọi là dấu nhắc dòng lệnh
Kí hiệu mô tả nhóm	wildcards	Kí hiệu được dùng để xác định một nhóm file (ví dụ *, ?, [] và [] với -). Còn được gọi là kí hiệu thay thế.
Cài đặt Linux	Installation Linux	Quá trình tạo ra hệ điều hành Linux lên máy tính để sử dụng

PHỤ LỤC A. QUÁ TRÌNH CÀI ĐẶT REDHAT-LINUX

Linux sử dụng phần cứng của máy PC hiệu quả hơn MS-DOS, Window hay WinNT, và do đó khả năng chịu các lỗi do cấu hình sai phần cứng sẽ kém hơn. Chúng ta cần làm một số việc trước khi bắt đầu cài đặt để giảm thiểu các khả năng không thể cài đặt tiếp khi gặp phải vấn đề này.

Trước tiên, hãy cố gắng tìm càng nhiều càng tốt các tài liệu về phần cứng máy PC mà mình định cài, như mainboard, card đồ hoạ, màn hình, modem... và để chúng ở nơi có thể tìm thấy và tra cứu dễ dàng.

Tiếp theo, tìm hiểu thông tin về phần cứng máy tính của và tập hợp chúng lại. Chúng ta có thể làm điều này khi sử dụng chức năng in cấu hình máy của một số tiện ích như MSD trong DOS, hoặc System Information trong Windows. Những thông tin chính xác về bàn phím, chuột, màn hình... sẽ giúp rất nhiều trong quá trình cấu hình X sau này.

Sau đó, kiểm tra phần cứng máy tính của để tìm ra các vấn đề nếu có, bởi chúng có thể làm quá trình cài đặt Linux bị treo. Sau đây là một số vấn đề thường gặp.

Một hệ thống DOS hay Windows có thể quản lý ổ đĩa IDE và CDROM cả khi jumper master/slave không được đặt đúng. Trong khi Linux không giải quyết được vấn đề này. Vì vậy nếu có nghi ngờ hãy xem lại các jumper này đã được đặt đúng chưa.

Một số thiết bị ngoại vi cần có những tiện ích để đặt cấu hình cho chúng khi máy khởi động. Các thiết bị như card mạng, CD-ROM, card âm thanh hoặc băng từ có thể gặp phải vấn đề này. Nếu trường hợp này xảy ra, có thể sử dụng lệnh đặt cấu hình lại tại dấu nhắc khởi động.

Một số hệ điều hành khác cho phép chuột dạng bus chia sẻ một IRQ với các thiết bị khác, trong khi Linux không hỗ trợ điều này. Nếu thử làm điều đó, hệ thống có thể bị treo.

Nên liên hệ với người dùng Linux có kinh nghiệm để được giải đáp những vướng mắc.

Cần tiến hành công việc chuẩn bị thời gian cho việc cài đặt. Quá trình cài đặt có thể kéo dài một tiếng hoặc hơn với những hệ thống chỉ cài Linux, hoặc lên tới ba tiếng với hệ thống cần chạy nhiều hệ điều hành khác nhau (thường với những hệ thống này khả năng bị treo máy hoặc có lỗi khi cài đặt sẽ cao hơn).

Phụ lục này giới thiệu cách cài đặt hai phiên bản RedHat là 6.2 và 7.1 nhằm cung cấp tính đa dạng khi cài đặt, tạo điều kiện cho người sử dụng.

AA. Cài đặt phiên bản RedHat 6.2

AA.1. Tạo đĩa mềm khởi động

Bước này chỉ cần thiết khi không thể khởi động từ ổ CD-ROM.

Nếu mua Red Hat Linux trực tiếp từ Red Hat Linux sẽ nhận kèm đĩa khởi động. Còn nếu mua bản copy từ công ty thứ 3, ta phải tự tạo đĩa khởi động và cách tạo là như sau:

Trong Windows, đưa đĩa mềm vào ổ. Bấm phím phải chuột vào desktop để tạo folder mới, đặt tên ***Bootdisk*** rồi mở folder này.

Đưa đĩa CD Red Hat vào ổ CD. Mở My Computer, nhấn vào ổ CD, mở folder có tên ***Dosutils***, nhấn vào file ***Rawrite***, bấm phím phải chuột và kéo nó vào ***Bootdisk***. Chọn Copy here từ menu xuất hiện.

Đóng cửa sổ Dosutils. Mở folder ***Images*** trong CD-ROM. Chép file ***Boot.img*** vào folder Bootdisk, giống như đã làm với ***Rawrite***.

Chọn Start.Run gõ vào Command trong hộp hội thoại, nhấn OK. Một cửa sổ xuất hiện với dấu nhắc DOS "C:\Windows\Desktop". Gõ vào ***cd bootdisk***, nhấn Enter.

Khi đã hoàn tất việc tạo đĩa mềm khởi động: gõ ***rawrite*** tại dấu nhắc DOS. Nhập ***boot.img*** là tên file muốn copy, nhấn Enter. Gõ ***a:*** (tên ổ đĩa mềm), và nhấn Enter khi được hỏi ổ đích.

AA.2. Phân vùng lại ổ đĩa DOS/Windows hiện thời

Trong hầu hết các hệ thống được sử dụng, ổ cứng thường được phân vùng cho MS-DOS, OS/2,... Cần thay đổi kích thước, sắp xếp lại các phân vùng này để tạo chỗ trống cho việc cài đặt Linux.

Cách tốt nhất để làm việc này là dùng một phần mềm chuyên dụng, chẳng hạn như phần mềm ***PQMagic*** của Power Quest. Dùng phần mềm này, có thể di chuyển / thay đổi kích thước / thêm / xoá / format các phân vùng trong ổ cứng một cách dễ dàng với giao diện đồ hoạ. Còn việc dùng FDisk của MS-DOS thì cực kỳ vất vả, muốn di chuyển / thay đổi kích thước của một phân vùng nào đó, đầu tiên phải ***backup*** tất cả các dữ liệu trong phân vùng, xoá phân vùng đó (việc này sẽ làm mất các thông tin về dữ liệu trong phân vùng), tiếp theo là tạo một phân vùng mới với kích thước mong muốn, cuối cùng là ***restore*** lại toàn bộ dữ liệu đã backup vào phân vùng mới tạo này! Tốt nhất dùng một phần mềm miễn phí cực mạnh như ***PQMagic***.

AA.3. Các bước cài đặt (bản RedHat 6.2 và khởi động từ CD-ROM)

Đưa đĩa CDROM Redhat 6.2 vào ổ CD, sau đó trong BIOS SETUP ta đặt chế độ khởi động từ ổ CD. Khi khởi động lại máy, quá trình sẽ được boot từ CDROM. Sau đây là chi tiết quá trình một đĩa khởi động cài đặt tiến hành.

<u>Lựa chọn chế độ cài đặt</u>

Hệ thống đưa ra các chế độ cho chúng ta lựa chọn :

- Gõ Enter chọn chế độ cài đặt đồ họa.
- Gõ "text" + Enter chọn chế độ Text.
- Gõ "expert" + Enter chọn chế độ Expert. Chọn chế độ này có nghĩa ta tự chọn cấu hình phần cứng còn hai chế độ trên sẽ tự động detect.
- Gõ "linux ks" + Enter chọn chế độ cài đặt từ mạng hoặc từ đĩa mềm.

<u>Lựa chọn ngôn ngữ hiển thị.</u>

Hệ thống đưa ra rất nhiều ngôn ngữ cho phép lựa chọn, ví dụ như Czech, English, French, German.... Thường chọn English.

<u>Lựa chọn cấu hình bàn phím</u>

Linux đòi hỏi lựa chọn cấu hình bàn phím từ các mô hình sau:

- Model :

  Brazilian ABNT 2
  Dell
  Generic.
  Microsoft.
  .....

Ta sẽ lựa chọn bàn phím tương thích theo các dạng trên, nếu không rõ bàn phím thuộc loại nào thì nên chọn kiểu Generic.

- **Layout**: Ngôn ngữ sử dụng để gõ (khoảng 24 ngôn ngữ)
- **Variant**: Có 2 chế độ là.

  Eliminate Dead Keys (khi chúng ta sử dụng các kí tự đặc biệt).
  None (kiểu mặc định).
- **Test**: chúng ta sẽ gõ các phím để kiểm tra thử.

Chọn cấu hình chuột

Hệ thống đưa ra 10 loại chuột để lựa chọn loại tương thích với chuột của mình, nếu không biết rõ chuột thuộc loại nào thì nên chọn kiểu Generic. Và phải chọn đúng kiểu chuột là PS/2 hay Serial nếu không thì sẽ không sử dụng được chuột.

Hệ thống đưa ra lời giới thiệu về bản Red Hat đang cài đặt.

Lựa chọn kiểu cài đặt.

Install (cài mới) gồm có các chế độ:

GNome Workstation.

KDE Workstation.

Server

Custom

Upgrade (Nâng cấp)

- **Tuỳ chọn cài đặt WorkStation**

Nếu lựa chọn kiểu cài mới là GNome Workstation hoặc KDE Workstation thì hệ thống sẽ cài đặt X Window System và chương trình quản lý Desktop theo dạng GNome hoặc KDE. Nếu chưa thạo lắm về Linux thì hãy sử dụng tuỳ chọn này, nó sẽ bỏ qua nhiều bước. Lựa chọn **Custom** là chỉ phù hợp với người đã quen với Linux. Cả hai lựa chọn WorkStation này sẽ chuẩn bị các việc sau:

Nếu đĩa cứng của chưa hề được phân vùng trước đó, Linux sẽ xoá hết tất cả các phân vùng trên các ổ đĩa cứng và cài đặt các phân vùng sau:

+ Một phân vùng swap kích thước 64MB

+ Một phân vùng root (được mount là /) chứa đựng mọi file được cài đặt.

Chú ý là chúng ta sẽ phải cần tối thiểu 600MB ổ cứng để tiến hành cài đặt theo kiểu WorkStattion.

Nếu hệ thống đã được cài sẵn Windows (Windows 3.1/95/98), kiểu cài đặt WorkStation sẽ tự động cấu hình hệ thống để khởi động ở chế độ song song sử dụng LILO.

- **Tuỳ chọn cài đặt Server**

Nếu lựa chọn kiểu cài đặt mới là Server thì hệ thống sẽ cài đặt theo kiểu máy chủ Linux và cũng như kiểu WorkStation, tuỳ chọn này sẽ tránh phải cấu hình nhiều thành phần phần cứng. Và khi cài đặt theo kiểu này, nên cẩn thận vì hệ thống sẽ xoá tất cả các partition trên

tất cả các ổ đĩa. Vì vậy, *chỉ chọn kiểu cài đặt **Server** nếu chắc chắn trong ổ cứng không có một dữ liệu gì cả*. Sau đây là các bước mà kiểu này tiến hành phân vùng ổ đĩa:

+ Tạo một phân vùng Swap kích thước 64MB.

+ Tạo một phân vùng kích thước 256MB với mount point là "/".

+ Tạo một phân vùng kích thước tối thiểu 512MB được mount là "/usr".

+ Tạo một phân vùng tối thiểu 215MB được mount là "/home".

+ Tạo một phân vùng kích thước 256MB được mount là "/var".

Như vậy chúng ta phải cần tối thiểu **1.6BG** ổ cứng để tiến hành cài đặt theo kiểu Server.

- **Tuỳ chọn cài đặt *Custom***

Đối với kiểu Custom hệ thống sẽ cho phép tự chọn các thành phần và cấu hình phần cứng để cài đặt một cách đầy đủ nhất, tuy nhiên cũng rối rắm và phức tạp nhất. Từ bước 7 trở đi, chỉ giới thiệu về tuỳ chọn cài đặt này. Sau đây sẽ giới thiệu tổng quát một số bước mà quá trình này thiết đặt:

Tạo các phân vùng: cần phải chỉ rõ Redhat sẽ được cài đặt vào phân vùng nào.

Format các phân vùng: Những phân vùng mới được thêm vào sẽ phải được format lại theo định dạng Linux filesystem. Tuy nhiên cũng có thể lựa chọn phân vùng nào cần phải format.

Lựa chọn và cài đặt các gói phần mềm đi kèm: Thực hiện sau khi đã phân vùng đĩa cứng.

Thiết đặt cấu hình LILO: có thể lựa chọn cài đặt LILO vào Master Boot Record hoặc Sector đầu tiên của phân vùng Root hoặc không lựa chọn cài LILO.

Xác định các Partition

Đầu tiên cần xác định các *điểm kích hoạt* (mount point) cho một hoặc nhiều partition.

Trong bảng partition có các thông tin sau:

- **Mount Point**: Xác định partition nào sẽ được kích hoạt khi Linux được cài đặt và chạy. Nếu partition tồn tại và có nhãn là ***not set*** thì ta xác định mount point bằng cách kích chuột vào nút Edit hoặc double - click trên partition.

Và hệ thống khuyên chúng ta nên tạo các partition theo cách sau:

Một ***swap partition*** (ít nhất 16MB) - dùng hỗ trợ bộ nhớ ảo. Nếu máy chúng ta có 16Mb Ram hoặc ít hơn thì bắt buộc chúng ta phải tạo swap partition. Thậm chí nếu có nhiều bộ nhớ hơn, chúng ta cũng nên tạo swap partition. Kích thước tối thiểu của swap partition = max |bộ nhớ Ram và 16Mb|.

Một ***boot partition*** (tối đa 16Mb) - chứa nhân của HĐH cùng với các file trong quá trình khởi động.

Một ***root partition*** (từ 500Mb - 1Gb) là nơi chứa thư mục gốc và tất cả các file (trừ các file ở trong boot partition). Với 500Mb cho phép cài theo kiểu Workstation và với 1Gb cho phép cài mọi thứ.

Device:

Hiện tên các device partition (Ví dụ: hda2 đại diện cho partition thứ 2 trên ổ cứng primary).

- **Request**: Cho biết không gian mà partition hiện có. Nếu muốn thay đổi kích thước thì chúng ta phải xoá partition đó và tạo lại bằng cách dùng nút "Add".
- **Actual**: Cho biết không gian mà partition đang sử dụng.
- **Type**: Cho biết kiểu của partition.

Và chúng ta có thể add, edit, và delete các partition bằng cách kích chuột vào các nút đó. Chức năng của từng nút là:

- **Add:** Dùng để tạo một partition mới, gồm có các thông tin sau:

Mount Point: gồm có các kiểu

/ :

/boot :

/usr :

/home :

/var :

/opt :

/tmp :

/usr/local :

Size (Megs): chọn kích thước của partition.

Grow to fill disk: nếu chọn thì partition này sẽ sử dụng toàn bộ vùng đĩa trống còn lại.

Partition type: gồm có các kiểu

Linux Swap: chọn kiểu này nếu chúng ta muốn tạo partition swap.

Linux Native : chọn kiểu này nếu chúng ta muốn tạo partition root.

Linux RAID :

DOS 16-bit < 32 :

DOS 16-bit > 32 :

Edit: Dùng để thay đổi mount point của partition.

Delete: Dùng để xoá partition.

Reset: Khôi phục lại những thay đổi.

Make RAID Device: Sử dụng Make Raid device chỉ khi đã có kinh nghiệm về RAID.

Drive Summaries: hiển thị thông tin về cấu hình đĩa.

Chọn Partition để Format

Gồm có các thông tin:

Partition muốn Format.

Lựa chọn “Check for bad blocks while formating”

Chọn “Check for bad blocks while formating” để tìm ra những bad bocks trên đĩa sau đó sẽ đánh dấu lại nhằm không ghi dữ liệu lên chúng nữa.

Chọn cấu hình LILO (Linux Loader)

Để chọn cấu hình LILO, phải không đặt dấu kiểm “Do not install LILO”. Nếu ổ Linux Native có tên là /dev/hda5 thì màn hình sẽ hiện ra cho phép cài đặt chương trình nạp Linux vào Master Boot Record (MBR) hoặc First Sector of boot partition (sector đầu tiên của phân vùng khởi động). Nói chung là nên chọn Master Boot Record để có thể khởi động từ nhiều ổ.

Nếu có các ổ đĩa SCSI hoặc đĩa cứng của hỗ trợ LBA thì cần phải đánh dấu kiểm vào mục “Use Linear Mode”.

Kernel Parameters: Các tham số sẽ dùng bất cứ khi nào nhân được khởi động.

Bảng ở dưới sẽ cho biết thông tin về các phân vùng: tên phân vùng, loại phân vùng, có phải là phân vùng khởi động không ?. cần chọn phân vùng khởi động là phân vùng có tên Linux (thường là phân vùng mà Redhat sẽ đặt mặc định và chúng ta không phải thay đổi gì). Nếu đã có phân vùng tên là ‘dos’, chính là phân vùng trước khi cài đặt thì sau này mỗi khi khởi động máy tính, chúng ta có thể chọn phân vùng này để khởi động một cách bình thường bằng cách gõ tên phân vùng đó tại dấu nhắc “LILO Boot” trong quá trình khởi động. Cho phép thay đổi tên của phân vùng tương ứng trong phần “Boot Label”.

Tuỳ chọn “Create boot disk”: chỉ cần chọn mục này nếu không cài LILO hoặc cài LILO nhưng không cài vào MBR. Chú ý là nếu không cài LILO thì bắt buộc phải chọn mục này để khởi động từ đĩa mềm.

Chọn múi giờ

Nếu cài trong chế độ đồ họa, màn hình mặc định sẽ hiện ra một bản đồ thế giới. Có thể thay đổi bản đồ theo các kiểu sau:

World (bản đồ thế giới)

North American (Bắc Mỹ)

South American (Nam Mỹ)

Pacific Rim (Châu úc)

Europe (Châu Âu)

Africa (Châu Phi)

Asia (Châu á)

Đối với Việt Nam ta thì chọn Asia/Saigon (trong đĩa CD cài đặt LinuxVN thì mục chọn là Asia/Hanoi).

Có thể chọn múi giờ theo kiểu UTC Offset (Universal Time Coordinated), tức là kiểu GMT +/- độ lệch. Nếu hệ thống giờ sở tại sử dụng UTC thì chọn “System clock uses UTC”.

Thiết đặt cấu hình Account (người sử dụng)

Như đã biết Linux kế thừa từ Unix, do đó nó cũng hỗ trợ chế độ đa người dùng như Unix. Bản cài đặt Redhat cho phép đặt luôn cấu hình người dùng ngay trong quá trình cài đặt.

Đầu tiên chúng ta cần đặt mật khẩu cho người dùng Root trong phần "Root Password" và xác nhận lại mật khẩu này trong mục "Confirm". Chú ý là mặc định tất cả các mật khẩu đều phải tối thiểu 6 ký tự.

Sau đó ta có thể thêm ngay một số người dùng đầu tiên: Tên đặt trong "Account Name", gõ password trong mục "Password" và xác nhận mật khẩu trong mục Password (confirm) ở ngay bên cạnh, tên đầy đủ của người dùng trong mục "Full Name". Sau đó gõ phím "Add" thể thêm người dùng mới vào danh sách các người dùng. Phím "Edit" cho phép hiển thị người dùng hiện hành trong bảng danh sách người dùng, phím "Delete" để xoá người dùng hiện thời.

Thiết đặt cấu hình quyền hạn (Authentication Configuration)

Thực hiện các mục sau:

Enable MD5 Password: cách mã hoá này cho phép mật khẩu dài tới 256 ký tự.

Shadow Password: đây là cách bảo mật tối đa cho mật khẩu, file chứa mật khẩu /etc/passwd sẽ được thay bằng etc/shadow và file này chỉ được phép hiển thị bởi người dùng Root. Chọn cả 2 mục này sẽ tăng tính bảo mật của hệ thống.

Enable NIS: chọn mục này nếu máy tính kết nối vào mạng NIS (Network Information System) cho phép một nhóm máy tính trong vùng Network Information Service với cùng một password.

NIS domain: tên của domain hoặc nhóm máy tính chứa hệ thống.

NIS server: cho phép máy tính dùng một NIS server rêng, hơn là một thông điệp rộng rãi trong mạng LAN.

Lựa chọn các gói phần mềm cài đặt (Pakage Selection)

Có rất nhiều mục để chọn sẽ nói chi tiết ở phần sau như: Printer Support, hệ thống X Window, GNome, KDE, DOS / Window Connectivity,.... Lựa chọn "Everything" sẽ cài đầy đủ tất cả mọi thứ, do đó cần phải lựa chọn các gói phần mềm phù hợp nếu ổ Linux Native / không đủ.

Các packages có thể được chọn cài đặt bao gồm:

Printer Support: Nếu được cài Linux sẽ cố gắng nhận máy in hoặc cho phép người dùng thiết lập các thông số về máy in của hệ thống.

X Window System: Môi trường đồ họa nguyên thuỷ trong Linux, được gọi tắt là X. X không phải là một giao diện đồ họa người dùng thực sự mà chỉ là một hệ cửa sổ cùng với các công cụ để một giao diện đồ họa người dùng có thể được xây dựng từ đó.

GNOME: Là một giao diện đồ họa người dùng đẹp hơn, tiện lợi và thân thiện hơn X, GNOME cũng cung cấp một giao diện lập trình ở mức cao hơn để tạo ra các ứng dụng với giao diện kiểu GNOME.

KDE: KDE là một giao diện đồ họa người dùng được phát triển dựa trên thư viện giao diện đồ họa C++ Qt. Thư viện giao diện này hỗ trợ UNIX, Windows và cả Mac. Do được phát triển trước GNOME nên KDE có nhiều điểm tốt hơn.

Mail/ WWW/ News tools: Bộ công cụ dùng để gửi, nhận thư tín điện tử, duyệt Web và đọc các bản tin.

Dos/ Windows Connectivity: Bộ giả lập DOS và Windows cho phép người dùng chạy các ứng dụng DOS và Windows ngay trong Linux.

Graphic Manupulation: Các tiện ích đồ họa như Gview, Imgedit... cho phép trình diễn và xử lý các file hình ảnh.

Games: Một số chương trình trò chơi giải trí trong chế độ text hoặc X Window.

Multimedia Support: Các chương trình hỗ trợ đa phương tiện, như nghe midi, wave, mp3... điều khiển joystick, thu âm ...

Dialup WorkStation: Cho phép người dùng sử dụng modem và quay số để sử dụng các dịch vụ qua đường điện thoại như gửi nhận mail, web...

News Server: Cung cấp dịch vụ máy chủ news, cho phép máy tính trở thành một server bản tin.

NFS Server: Máy chủ hệ thống file mạng NFS (Network File System).

SMB Server: Máy chủ hệ thống file mạng Samba.

IPX/ Netware Connectivity: Giúp hệ thống chạy Linux giao tiếp với các máy tính khác qua mạng Netware sử dụng giao thức IPX.

Anonymous FTP Server: Cài đặt package này giúp máy tính trở thành một máy chủ FTP (File Transfer Protocol), có thể cung cấp dịch vụ truyền file cho các máy khác trong mạng.

Web Server: Cài đặt package này giúp máy tính trở thành một máy chủ Web, có thể cung cấp dịch vụ web cho các máy khác trong mạng.

DNS Name Server: Cài đặt package này giúp máy tính trở thành một máy chủ DNS, có thể cung cấp dịch vụ đặt tên vùng cho các máy khác trong mạng.

Postgres (SQL) Server: Cài đặt package này giúp máy tính trở thành một máy chủ SQL, có thể cung cấp dịch vụ truy vấn dữ liệu cho các máy khác trong mạng.

Network Management Workstation: Giúp máy tính trạm làm việc điều hành trở thành một máy chủ SQL, có thể cung cấp dịch vụ truy vấn dữ liệu cho các máy khác trong mạng.

TeX Document Formatting: Hệ soạn thảo và định dạng văn bản dưới dạng Tex.

Emacs: Hệ soản thảo văn bản đơn giản.

Development: Các bộ biên dịch, gỡ rối, công cụ phát triển phần mềm ... dưới các ngôn ngữ như Perl, C, C++. Mã nguồn của các chương trình trong Linux.

Kernel Development: Mã nguồn nhân Linux và bộ công cụ phát triển dành cho phát triển nhân Linux.

Extra Documentation: Mọi tài liệu về Linux có trong đĩa CD, dưới những ngôn ngữ như Anh, Pháp, Italia, Tây Ban Nha...

Utilities: Các tiện ích cho Linux.

Thiết đặt cấu hình X (X Configuration)

Phần này sẽ đặt cấu hình card màn hình để thể hiện khi chúng ta sử dụng các hệ thống X Window. Ví dụ sau đây là một số thông số hiển thị khi chúng ta cài đặt:

Video Card: ATI Mach64

Video Ram: 4096KB

X server: Mach64

Monitor: 14" COLOR

Độ quét ngang: 30 - 54 Khz

Độ quét dọc: 50 - 120 Hz

Đây là phần khá quan trọng, nếu card màn hình của không phải là dạng chuẩn thì sẽ phải cài bằng tay, một công việc khá mệt mỏi.

Test this Configuration: ấn vào đây để kiểm tra tính năng đồ hoạ có hoạt động tốt không, nếu màn hình của qua khỏi cuộc kiểm tra này thì có thể coi là đã thành công tới 90% quá trình cài đặt Redhat.

Customize X Configuration: Nếu chọn mục này thì ghi gõ Next, Redhat sẽ cho một bảng cho thiết lập bằng tay các chế độ đồ họa 8 bits, 16 bits, 32 bits; các chế độ phân giải 640x480, 800x600, 1024x786, ... Chú ý là khi chọn độ phân giải nào thì luôn luôn gõ phím "Test this Configuration" để đảm bảo màn hình của luôn hiển thị đúng.

Bắt đầu quá trình copy từ đĩa CD vào ổ cứng

Bước cuối cùng này sẽ thực hiện các bước format ổ Linux Native / (nếu đã chọn ở phần trước), format ổ Linux Swap, và sau đó là giải nén tất cả các gói phần mềm mà đã lựa chọn trong bước thứ 13 vào ổ cứng. Quá trình này mất khoảng 10 phút đến 20 phút tuỳ theo số lượng các gói chọn.

Kết thúc quá trình này, gõ "Exit", khởi động lại máy, nhớ tháo đĩa CD Redhat ra khỏi ổ để bắt đầu khởi động từ ổ cứng.

AA.4. Các hạn chế về phần cứng đối với Linux

Các bộ vi xử lý mà Linux hỗ trợ

Linux chủ yếu chạy trên các máy PC thế hệ 386, 486, 586 sử dụng các phần cứng họ vi xử lý 80386. Việc cài đặt trên các phần cứng khác thì vẫn đang ở trong giai đoạn thiết kế.

Có thể chạy thử Linux bằng một máy với phần cứng tối thiểu là: bộ vi xử lý Intel 386, 486, 586, 4MB RAM và một ổ mềm. Dĩ nhiên là càng nhiều RAM thì càng tốt.

Linux hỗ trợ VESA Local Bus và PCI.

Linux cũng hỗ trợ phần lớn cho các ổ cứng chuẩn ESDI và MCA (bus độc quyền của IBM).

Linux cũng có thể chạy trên các laptop họ 386.

Có một cách cài đặt Linux trên 8086 được biết đến dưới tên gọi ELKS (Embeddable Linux Kernel Subset). Đây là một nhân Linux 16 bit được chủ yếu sử dụng trong các hệ thống nhúng. Thực ra phiên bản Linux hiện nay không thể chạy được đầy đủ trên 8086 hay 286 bởi vì những bộ vi xử lý này không hỗ trợ cho việc chuyển đổi tác vụ cũng như quản lý bộ nhớ.

Linux hỗ trợ đa quá trình cho các kiến trúc Intel MP.

Dưới đây là danh sách các bộ VXL mà Linux hỗ trợ:

- Dòng 68000 của Amigas và Ataris hiện đang được triển khai nghiên cứu.
- Các phiên bản GNU/Linux cũng được thử cài đặt cho các nền Alpha, Sparc, PowerPC, ARM.
- Một dự án về Linux trên PPC cũng đã được tiến hành.
- Apple đã hỗ trợ MkLinux trên các Power Macs dựa trên OSF của Mach vi nhân.
- Linux cho máy 64 bit DEC Alpha/AXP.
- Đang nghiên cứu về Linux cho MIPS, bắt đầu đối với R4 trên các máy Deskstation Type.
- Hiện có 2 bản Linux cho dòng máy dùng họ vi xử lý ARM. Một là của vi xử lý ARM3 trên các máy Acorn A5000 và nó còn bao gồm cả các thiết bị I/O cho 82710. Còn lại là cho họ vi xử lý ARM610 của máy Acorn RISC PC.
- Linux cho SPARC đang được tiến hành.
- Có bản ***Hardhat*** cho các máy SGI/Indy.

Các yêu cầu về không gian ổ cứng

Đối việc cài đặt tối thiểu là 10 MB, chủ yếu là để thử chứ không có nhiều các tính năng khác.

Ta có thể cài đặt thêm X với khoảng 80 MB. Nếu cài đặt cả bộ GNU/Linux sẽ cần khoảng 500MB-1GB bao gồm cả mã nguồn và nhiều thứ khác nữa.

Các yêu cầu về bộ nhớ

Tối thiểu là 4MB. Ta có thể sử dụng swapping để chạy thêm các cài đặt khác. Linux nói chung là chạy tương đối "thoải mái" với 4MB Ram nhưng các ứng dụng X Windows sẽ chạy chậm bởi vì chúng cần phải thực việc *swap* vào ra trên đĩa.

Một vài ứng dụng hiện tại lại chỉ chạy bình thường với 128MB bộ nhớ vật lý chẳng hạn như Netscape.

Sự tương thích với các hệ điều hành khác: DOS, OS/2, 386BSD, Win95

Linux sử dụng sắp xếp phân dạng giống như MS-DOS do đó nó có thể chia sẻ đĩa với các hệ điều hành khác. Tuy vậy, điều này cũng có nghĩa là các hệ điều hành khác cũng có thể không hẳn là hoàn toàn tương thích. Các trình *Fdisk* và *Format* của DOS thỉnh thoảng lại có thể viết đè lên dữ liệu trong phân vùng của Linux bởi vì chúng có thể sử dụng các thông tin phân vùng sai lệnh từ boot sector của phân vùng chứ không phải là từ bảng phân vùng. Để tránh hiện tượng này, một ý tưởng là đưa về 0 địa chỉ bắt đầu của một phân vùng

vừa mới tạo lập trong Linux trước khi sử dụng các lệnh format của MS-DOS. Sử dụng lệnh sau:

$ dd if=/dev/zero of=/dev/hdXY bs=512 count=1

Với hdXY là phân vùng liên quan, chẳng hạn /dev/hda1 là phân vùng đầu tiên trên đĩa IDE đầu tiên.

Linux có thể đọc và ghi các file trên các phân vùng FAT của DOS và OS/2 và các đĩa mềm bằng cách sử dụng hệ thống file DOS được tích hợp vào nhân hoặc các công cụ **mtool**. Nhân cũng cung cấp hỗ trợ cho hệ thống file VFAT của Windows 9x và Windows NT. Hiện tại các đĩa phân vùng theo NTFS cũng đang được nghiên cứu hỗ trợ cùng với việc hỗ trợ nén đĩa như là một tính năng chuẩn.

Linux cũng có thể truy cập được tới hệ thống file HPFS của OS/2 nhưng chỉ ở chế độ *read-only*. Người ta có thể thực hiện điều này như một lựa chọn khi biên dịch nhân.

Linux cũng hỗ trợ cho việc thao tác trên các định dạng AFFS (Amiga Fast File System) từ bản 1.3 trở về sau bằng cách như một lựa chọn lúc biên dịch hay như một mô đun riêng. Tuy vậy, điều này cũng chỉ dừng ở mức độ chỉ đọc. Các truy cập đĩa mềm thì chưa có hỗ trợ bởi vì sự khác biệt giữa các điều khiển đĩa của PC và Amiga.

Đối với các máy chạy các hệ điều hành của Unix như BSD, System V... thì các nhân hiện tại cũng mới chỉ có thể đọc hệ thống file UFS trên System V, Xenix, BSD, một số sản phẩm thừa kế khác như SunOS, FreeBSD, NetBSD, NeXTStep. Hỗ trợ UFS cũng được coi như một lựa chọn lúc biên dịch nhân hay như một mô đun.

Linux cho phép đọc/viết trên các ổ đĩa SMB của các nhóm Windows và WinNT. Có một chương trình tên là Samba cho phép truy cập và hệ thống file mạng WfW (miễn là dùng giao thức TCP/IP) .

Đối với các máy Macintosh thì có một tập hợp các chương trình ở cấp độ người dùng có thể đọc, ghi trên HFS (Macintosh Hierarchical File System).

Câu hỏi đặt ra là có thể chạy một chương trình Windows trong Linux hay không ? Chương trình tên WINE đang được phát triển để mô phỏng môi trường Windows trong Linux. Hiện tại khi muốn dùng hai hệ điều hành cùng lúc với Linux thì ta đã có chương trình LILO boot. LILO boot bắt buộc ta phải lựa chọn hệ điều hành vào lúc khởi động. Ngoài ra, còn có một chương trình tên LOADLIN là một chương trình DOS cho phép nạp Linux (cũng như bất kỳ hệ điều hành khác) khiến cho Linux cùng tồn tại với DOS. LOADLIN đặc biệt hữu dụng khi ta muốn cài Linux trên các ổ đĩa thứ 3, 4 của hệ thống (hoặc khi ta thêm một ổ SCSI vào một hệ thống có chứa ổ IDE). Trong trường hợp này thì LILO boot sẽ không có khả năng tìm kiếm và nạp nhân. Do đó ta sẽ phải tạo một thư mục chẳng hạn C:ữLINUX, đặt LOADLIN vào trong đó cùng với một bản copy của nhân và rồi sử dụng nó.

Chú ý: Cần tạo ít nhất một phân vùng Linux dưới giới hạn 1024 cylinder logic.

PHỤ LỤC B. TRÌNH SOẠN THẢO VIM

UNIX có hai bộ soạn thảo là **ed** và **vi** trong đó **vi** được ưa chuộng hơn do **vi** được phát triển từ bộ soạn thảo dòng lệnh **ed**. Trong chế độ văn bản, Linux cho phép người dùng sử dụng trình soạn thảo **vim** mà **vim** chính là bộ soạn thảo tương thích với **vi**. **vim** được phần lớn người dùng sử dụng để soạn thảo các file văn bản ASCII, đặc biệt là tạo ra các văn bản chương trình nguồn.

vim có sáu chế độ cơ bản:

- Chế độ thường (**Normal mode**): trong chế độ thường người dùng được phép nhập tất cả các lệnh soạn thảo thông thường. Nếu không thiết lập tuỳ chọn **insertmode**, ngầm định vào ngay chế độ thường khi khởi động **vim**. Chế độ thường còn được gọi là ***chế độ lệnh***.
- Chế độ ảo (**Visual mode**): chế độ này cũng gần giống như chế độ thường, chỉ khác ở chỗ là lệnh di chuyển có tác dụng đánh dấu văn bản. Mặt khác, các lệnh khác (không là lệnh di chuyển) thực sự tác dụng trong phạm vi những đoạn văn bản đã được đánh dấu.
- Chế độ chọn lựa (**Select mode**): chế độ này tương tự như chế độ lựa chọn của MS-Windows. Người dùng có thể nhập một ký tự thuộc loại in ấn được để xoá một sự lựa chọn và chạy chế độ chèn.
- Chế độ chèn (**Insert mode**): Trong chế độ này, có thể soạn thảo văn bản bình thường như các bộ soạn thảo quen biết khác. Văn bản đó sẽ được chèn vào trong bộ đệm.
- Chế độ dòng lệnh (**Command-line mode** hay **cmdline mode**): Trong chế độ này, một dòng lệnh được nhập tại đáy cửa sổ soạn thảo. Đó có thể là các lệnh **Ex** (**:**), các lệnh tìm kiếm (**/** hay **?**), và các lệnh lọc (**!**).
- Chế độ **Ex** (**Ex mode**): giống như chế độ dòng lệnh, nhưng sau khi nhập một lệnh, vẫn ở trong chế độ **Ex**. Tuy nhiên còn rất nhiều hạn chế đối với các lệnh ở chế độ này.

Ngoài ra còn có năm chế độ phụ sau:

- Chế độ chờ thực hiện (**Operator-pending mode**): chế độ này giống chế độ thường, nhưng sau khi gọi một lệnh, **vim** sẽ chờ cho đến khi đoạn văn bản chịu tác động của lệnh được đưa ra.
- Chế độ thay thế (**Replace mode**): chế độ thay thế là một trường hợp đặc biệt của chế độ chèn. Người dùng có thể nhập mọi ký tự như trong chế độ chèn, chỉ khác ở chỗ: mỗi ký tự nhập sẽ thay thế cho một ký tự đã tồn tại (có thể gọi là chế độ đè - overwrite).
- Chế độ chèn-lệnh (**Insert Normal mode**): gõ CTRL-O trong chế độ chèn để chuyển sang chế độ chèn-lệnh. Chế độ này cũng giống như chế độ thường, nhưng sau khi thực hiện một lệnh, **vim** sẽ trở lại chế độ chèn.
- Chế độ chèn-ảo (**Insert Visual mode**): chế độ này được sinh ra khi trong chế độ chèn thực hiện một sự lựa chọn ảo. **vim** sẽ trở về chế độ chèn sau khi sự lựa chọn ảo đó kết thúc.

- Chế độ chèn-lựa chọn (**Insert Select mode**): chế độ này được khởi tạo khi chạy chế độ lựa chọn trong chế độ chèn. Khi chế độ lựa chọn kết thúc, **vim** sẽ trở về chế độ chèn.

Việc chuyển đổi giữa các chế độ trong **vim** được thực hiện nhờ các ***lệnh*** (phím lệnh hoặc xâu lệnh) của **vim** và được tập hợp trong bảng dưới đây. Trong bảng này, cột đầu tiên là chế độ nguồn, hàng đầu tiên là chế độ đích, ô giao giữa hàng và cột chứa các phím lệnh chuyển chế độ (ký hiệu ***1, *2, *3, *4, *5, *6** là cách viết tắt danh sách lệnh được giải thích ở sau):

Chế độ hiện thời	***Chế độ cần chuyển tới***						
	Thường	**ảo**	**Lựa chọn**	**Chèn**	**Thay thế**	**Dòng lệnh**	**Ex**
Thường		v, V, ^V	***4**	***1**	R	:, /, ?, !	Q
ảo	***2**		^G	c, C	--	:	--
Lựa chọn	***5**	^O, ^G		***6**	--	:	--
Chèn	<Esc>	--	--		<Insert>	--	--
Thay thế	<Esc>	--	--	<Insert>		--	--
Dòng lệnh	***3**	--	--	:start	--		--
Ex	:vi	--	--	--	--	--	

Giải thích các lệnh viết tắt:

- ***1** Để chuyển sang chế độ chèn từ chế độ thường, sử dụng một trong các phím: i, I, a, A, o, O, c, C, s, S.
- ***2** Để chuyển sang chế độ thường từ chế độ ảo: ngoài <Esc>, v, V, CTRL-V có thể gõ một phím lệnh thông thường (ngoại trừ phím lệnh di chuyển con trỏ).
- ***3** Để chuyển sang chế độ thường từ chế độ dòng lệnh:
  - Thực hiện lệnh <Enter>
  - Gõ CTRL-C hoặc <Esc>
- ***4** Để chuyển sang chế độ lựa chọn từ chế độ thường:
  - Sử dụng chuột để lựa chọn văn bản
  - Sử dụng các phím không in được để di chuyển dấu nhắc trỏ trong khi ấn giữ phím SHIFT
- ***5** Để chuyển sang chế độ thường từ chế độ lựa chọn: sử dụng các phím không in được để di chuyển dấu nhắc trỏ mà không nhấn phím SHIFT.
- ***6** Để chuyển sang chế độ chèn từ chế độ lựa chọn: nhập một ký tự có thể in được.

Dưới đây trình bày nội dung một số các lệnh cơ bản trong **vim**.

B.1 Khởi động vim

B.1.1 Mở chương trình soạn thảo vim

Cách đơn giản nhất bắt đầu dùng **vim** để soạn thảo một file văn bản, là gõ một trong ba lệnh sau:

vim [**tuỳ-chọn**]	bắt đầu soạn thảo hay hiệu chỉnh một file
vim [**tuỳ-chọn]** <danh sách các file>	bắt đầu soạn thảo một hoặc nhiều file
vim [tuỳ chọn] **-**	soạn thảo một file từ thiết bị vào chuẩn

Nếu tham số danh sách các file không có thì **vim** sẽ thao tác với một file mới (vùng đệm soạn thảo rỗng). Ngược lại, file đầu tiên trong danh sách trở thành file hiện hành và được đọc vào trong vùng soạn thảo. Con trỏ sẽ xuất hiện ở đầu dòng đầu tiên của vùng này. Để hướng đến file kế tiếp, ta đánh lệnh "**:next**" ở chế độ lệnh. Để soạn thảo một file có tên bắt đầu bằng "**-**" thì phải điền vào tên file dấu "**--**".

Ví dụ:

```
# vim vdvim`
~       
~       
~
~
~
~
~
"vdvim"[New File] 0,0-1 All
```

Lệnh trên mở một cửa sổ cho người dùng soạn thảo một file mới có tên là **"vdvim"**

Một số các tuỳ chọn cơ bản:

+ [n]	đặt dấu nhắc trỏ tại dòng thứ n (ngầm định là dòng cuối)
+ <lệnh>	thực hiện lệnh sau khi nạp file
+/<mẫu> <file>	đặt dấu nhắc trỏ tại dòng đầu tiên có chứa mẫu trong file
-o[n]	mở n cửa sổ (ngầm định có một cửa sổ cho một file: n=1)
--help	hiển thị danh sách các tham số và thoát

B.1.2. Tính năng mở nhiều cửa sổ

Trong **vim**, có thể chia cửa sổ soạn thảo hiện thời thành nhiều phần hay mở nhiều cửa sổ cùng lúc để soạn thảo các file khác nhau.

Ví dụ lệnh sau sẽ mở hai file **vd1** và **vd2** trên hai cửa sổ soạn thảo:

```
# vim -o2 vd1 vd2
~       
~
~
vd1 0,0-1 All
~
```

~
~
vd2 0,0-1 All
"vd2" [New File]

Sau đây là một số các lệnh hay dùng:

CTRL-W	chia cửa sổ hiện tại thành hai phần
:split <file>	chia cửa sổ và soạn thảo <file> trên một phần chia của cửa sổ
:sf <file>	chia cửa sổ, tìm file trên đường dẫn và soạn thảo nó
CTRL-W CTRL-^	chia cửa sổ và edit alternate file
CTRL-W **n**	tạo một cửa sổ trống mới (giống :new)
CTRL-W **q**	dừng việc soạn thảo và đóng cửa sổ (giống **:q**)
CTRL-W **o**	phóng to cửa sổ hiện hành trên toàn màn hình
CTRL-W **j**	di chuyển trỏ soạn thảo xuống cửa sổ dưới
CTRL-W **k**	di chuyển trỏ soạn thảo lên cửa sổ trên
CTRL-W **t**	di chuyển trỏ soạn thảo lên đỉnh cửa sổ
CTRL-W **b**	di chuyển trỏ soạn thảo xuống đáy cửa sổ
CTRL-W **p**	di chuyển trỏ soạn thảo đến cửa sổ được kích hoạt lúc trước
CTRL-W **x**	di chuyển trỏ soạn thảo đến cửa sổ tiếp theo
CTRL-W **=**	tạo tất cả các cửa sổ có chiều cao như nhau
CTRL-W **-**	giảm chiều cao của cửa sổ hiện thời
CTRL-W **+**	tăng chiều cao của cửa sổ hiện thời
CTRL-W **Ỵ**	thiết đặt chiều cao của cửa sổ hiện thời

B.1.3. Ghi và thoát trong vim

Bảng dưới đây giới thiệu các lệnh để ghi nội dung file lên hệ thống file và thoát khỏi **vim** sau khi đã soạn thảo xong nội dung của file (tham số ửn, mứ nếu có mang ý nghĩa "*từ dòng n tới dòng m")*.

:[n,m] w [!]	ghi file hiện thời.
:[n,m] w <file>	ghi nội dung ra <file>, trừ khi file đó đã thực sự tồn tại
:[n,m] w! <file>	ghi nội dung ra <file>, nếu file đã tồn tại thì ghi đè lên nội dung cũ
:[n,m] w[!] >> [<file>]	chèn thêm vào <file>, nếu không có file, mặc định là file hiện thời
:[n,m] w !<lệnh>	thực hiện <lệnh> trên các dòng từ dòng thứ n đến dòng thứ m như thiết bị vào chuẩn
:[n,m] up [thời gian] [!]	ghi file hiện thời nếu nó được sửa đổi
:q [!]	thoát khỏi **vim**
:wq [!] [<file>]	ghi nội dung <file> (mặc định là file hiện thời) và thoát khỏi **vim**
:x [!] <file>	giống :wq nhưng chỉ ghi khi thực sự có sự thay đổi trong nội dung file (giống ZZ)
:st [!]	dừng **vim** và khởi tạo một shell (giống CTRL-Z)

B.2. Di chuyển trỏ soạn thảo trong Vim

B.2.1. Di chuyển trong văn bản

Di chuyển trỏ soạn thảo trong văn bản là một tính năng rất quan trọng trong một trình soạn thảo văn bản **vim**. Dưới đây là một số các lệnh để thực hiện việc trên (cột đầu tiên có n chỉ một số là số lượng):

N	**l**	di chuyển trỏ soạn thảo về bên phải n ký tự
N	**h**	di chuyển trỏ soạn thảo về bên trái n ký tự
n	**k**	di chuyển trỏ soạn thảo lên n dòng
n	**j**	di chuyển trỏ soạn thảo xuống n dòng
	0	di chuyển về đầu dòng
	^	di chuyển đến từ đầu tiên của dòng hiện tại
	$	di chuyển đến cuối dòng
	<Enter>	di chuyển đến đầu dòng tiếp theo
n	-	di chuyển đến đầu dòng trước dòng hiện tại n dòng
n	+	di chuyển đến đầu dòng sau dòng hiện tại n dòng
n	_	di chuyển đến đầu dòng sau dòng hiện tại n-1 dòng
	G	di chuyển đến dòng cuối cùng trong file
n	G	di chuyển đến dòng thứ n trong file (giống **:n**)
	H	di chuyển đến dòng đầu tiên trên màn hình
	M	di chuyển đến dòng ở giữa màn hình
n	gg	di chuyển đến đầu dòng thứ n (mặc định là dòng đầu tiên)
n	gk	di chuyển lên n dòng màn hình
n	gj	di chuyển xuống n dòng màn hình

B.2.2. Di chuyển theo các đối tượng văn bản

vim cung cấp các lệnh dưới đây cho phép di chuyển trỏ soạn thảo nhanh theo các đối tượng văn bản và điều đó tạo nhiều thuận tiện khi biên tập, chẳng hạn, trong các trường hợp người dùng cần xoá bỏ hay thay đổi một từ, một câu ...

N	W	di chuyển n từ tiếp theo
N	E	di chuyển đến cuối của từ thứ n
N	B	di chuyển ngược lại n từ
N	ge	di chuyển ngược lại n từ và đặt dấu nhắc trỏ tại chữ cái cuối từ
N	>	di chuyển đến n câu tiếp theo
N	<	di chuyển ngược lại n câu
N	|	di chuyển đến n đoạn tiếp theo
N	|	di chuyển ngược lại n đoạn
N	]]	di chuyển đến n phần tiếp theo và đặt dấu nhắc trỏ tại đầu phần
N	[[	di chuyển ngược lại n phần và đặt dấu nhắc trỏ tại đầu phần
n	][	di chuyển đến n phần tiếp theo và đặt dấu nhắc trỏ tại cuối phần
n	[]	di chuyển ngược lại n phần và đặt dấu nhắc trỏ tại cuối phần

B.2.3. Cuộn màn hình

Màn hình sẽ tự động cuộn khi di trỏ soạn thảo đến đáy hoặc lên đỉnh màn hình. Tuy nhiên các lệnh sau đây giúp người dùng cuộn màn hình theo ý muốn:

N	<CTRL-f>	cuộn lên n màn hình (mặc định là 1 màn hình)
N	<CTRL-b>	cuộn xuống n màn hình (mặc định là 1 màn hình)
N	<CTRL-d>	cuộn xuống n dòng (mặc định là 1/2 màn hình)
N	<CTRL-u>	cuộn lên n dòng (mặc định là 1/2 màn hình)
N	<CTRL-e>	cuộn xuống n dòng (mặc định là 1 dòng)
N	<CTRL-y>	cuộn lên n dòng (mặc định là 1 dòng)
	z<Enter>	vẽ lại cửa sổ soạn thảo, dòng hiện tại sẽ là dòng trên cùng của cửa sổ (giống zt)
	z.	vẽ lại cửa sổ soạn thảo, dòng hiện tại sẽ là dòng ở giữa của cửa sổ (giống zz)
	z-	vẽ lại cửa sổ soạn thảo, dòng hiện tại sẽ là dòng ở đáy của cửa sổ (giống zb)

B.3. Các thao tác trong văn bản

vim có rất nhiều các lệnh hỗ trợ thao tác soạn thảo hay hiệu chỉnh một file. Phần dưới đây giới thiệu chi tiết về các cách để thêm văn bản, hiệu chỉnh văn bản hay xoá một văn bản.

Khi soạn thảo văn bản, nhiều dòng có thể được nhập bằng cách sử dụng phím **Enter**. Nếu có một lỗi cần phải sửa, có thể sử dụng các phím mũi tên để di chuyển trỏ soạn thảo trong văn bản và sử dụng các phím **Backspace** hoặc **Delete** để hiệu chỉnh.

B.3.1. Các lệnh chèn văn bản trong vim

	A	chèn văn bản vào vị trí dấu nhắc trỏ hiện thời (n lần)
N	A	chèn văn bản vào cuối một dòng (n lần)
n	i	chèn văn bản vào bên trái dấu nhắc trỏ (n lần)
n	I	chèn văn bản vào bên trái ký tự đầu tiên khác trống trên dòng hiện tại (n lần)
n	gI	chèn văn bản vào cột đầu tiên (n lần)
n	o	chèn n dòng trống vào dưới dòng hiện tại
n	O	chèn n dòng trống vào trên dòng hiện tại
	:r file	chèn vào vị trí con trỏ nội dung của file
	:r! lệnh	chèn vào vị trí con trỏ kết quả của lệnh lệnh

B.3.2. Các lệnh xoá văn bản trong vim

Bên cạnh các lệnh tạo hay chèn văn bản, **vim** cũng có một số lệnh cho phép người dùng có thể xoá văn bản. Dưới đây là bảng liệt kê một số lệnh cơ bản:

N	x	xoá n ký tự bên phải dấu nhắc trỏ
N	X	xoá n ký tự bên trái dấu nhắc trỏ
N	dd	xoá n dòng kể từ dòng hiện thời

	D hoặc d$	xoá từ vị trí hiện thời đến hết dòng
N	dw	xoá n từ kể từ vị trí hiện thời
	dG	xoá từ vị trí hiện thời đến cuối file
	d1G	xoá ngược từ vị trí hiện thời đến đầu file
	dn$	xoá từ dòng hiện thời đến hết dòng thứ n
N,m	d	xoá từ dòng thứ n đến dòng thứ m
N	cc	xoá n dòng, kể cả dòng hiện thời rồi khởi tạo chế độ chèn (Insert)
N	C	xoá n dòng kể từ vị trí hiện thời rồi khởi tạo chế độ chèn (Insert)
	cn$	xoá từ dòng hiện thời đến hết dòng thứ n rồi khởi tạo chế độ chèn (Insert)
N	s	xoá n ký tự và chạy chế độ chèn (Insert)
N	S	xoá n dòng và chạy chế độ chèn (Insert)

B.3.3. Các lệnh khôi phục văn bản trong vim

Các lệnh sau cho phép khôi phục lại văn bản sau một thao tác hiệu chỉnh nào đó:

N	u	khôi phục lại văn bản như trước khi thực hiện n lần thay đổi
	U	khôi phục lại hoàn toàn dòng văn bản hiện thời như trước khi thực hiện bất kỳ sự hiệu chỉnh nào trên dòng đó
	: e!	hiệu chỉnh lại. Lưu trữ trạng thái của lần ghi trước
N	CTRL-R	làm lại (redo) n lần khôi phục (undo) trước đó !

6.3.4. Các lệnh thay thế văn bản trong vim

vim còn có các lệnh cho phép thay đổi văn bản mà không cần phải xoá văn bản rồi sau đó đánh mới.

n	r <ký tự>	thay thế n ký tự bên phải dấu trỏ bởi <ký tự>
	R	ghi đè văn bản bởi một văn bản mới (hay chuyển sang chế độ thay thế - Replace trong Vim)
n	~	chuyển n chữ hoa thành chữ thường và ngược lại
n	gUU	chuyển các ký tự trên n dòng, kể từ dòng hiện tại, từ chữ thường thành chữ hoa
n	guu	chuyển các ký tự trên n dòng, kể từ dòng hiện tại, từ chữ hoa thành chữ thường
n	CTRL-A	cộng thêm n đơn vị vào số hiện có
n	CTRL-X	bớt đi n đơn vị từ số hiện có
n	> [> ...]	chuyển dòng thứ n sang bên phải x khoảng trống (giống như phím TAB trong Win), nếu không có n mặc định là dòng hiện tại, x là số dấu ' > ' (ví dụ: >>> thì x bằng 3)
n	< [< ...]	chuyển dòng thứ n sang bên trái x khoảng trống (giống như phím SHIFT+TAB trong Win), nếu không có n mặc định là dòng

		hiện tại, x là số dấu ' < '
n	J	kết hợp n dòng, kể từ dòng hiện tại, thành một dòng
n	gJ	giống như J nhưng không chèn các khoảng trống
	:[n,m] ce [width]	căn giữa từ dòng thứ n đến dòng thứ m với độ rộng là width, nếu không có width, mặc định độ rộng là 80
	:[n,m] ri [width]	căn phải từ dòng thứ n đến dòng thứ m với độ rộng là width, nếu không có width, mặc định độ rộng là 80
	:[n,m] le [width]	căn trái từ dòng thứ n đến dòng thứ m với độ rộng là width, nếu không có width, mặc định độ rộng là 80
	:[n,m]s/<mẫu1>/<mẫu2>/[g][c]	tìm từ dòng thứ n đến dòng thứ m và thay thế mẫu1 bởi mẫu2. Với [g], thay thế cho mọi mẫu tìm được. Với [c], yêu cầu xác nhận đối với mỗi mẫu tìm được
	:[n,m]s[g][c]	lặp lại lệnh tìm và thay thế trước (:s) với phạm Vim mới từ dòng n đến dòng m kèm theo là các tuỳ chọn
	&	lặp lại việc tìm kiếm và thay thế trên dòng hiện thời mà không có các tuỳ chọn

B.3.5. Sao chép và di chuyển văn bản trong vim

Phần này giới thiệu với các các lệnh cơ bản để cắt và dán văn bản trong **vim**.

Để sao chép văn bản phải thực hiện ba bước sau:

- Sao chép văn bản vào một bộ nhớ đệm (**`Yanking`**)
- Di chuyển dấu nhắc trỏ đến vị trí cần sao chép (**Moving**)
- Dán văn bản (**Pasting**)

Sau đây là các lệnh cụ thể của từng bước:

*** <u>Sao chép văn bản vào bộ nhớ đệm</u>**

n	yw	sao chép n ký tự
n	Y	sao chép n dòng văn bản, kể từ dòng hiện tại, vào bộ nhớ đệm (giống yy)
	: [n] co [m]	sao chép dòng thứ n vào dưới dòng thứ m

*** <u>Dán văn bản:</u>**

n	P	dán đoạn văn bản được sao chép vào bên phải vị trí hiện thời (n lần)
n	P	dán n đoạn văn bản được sao chép vào bên trái vị trí hiện thời (n lần)
n	Gp	giống như p, nhưng đưa dấu nhắc trỏ về sau đoạn văn bản mới dán
n	gP	giống như P, nhưng đưa dấu nhắc trỏ về sau đoạn văn bản mới dán

: [n] put m	dán m dòng văn bản vào sau dòng thứ n (nếu không có n ngầm định là dòng hiện tại)
: [n] put! m	dán m dòng văn bản vào trước dòng thứ n (nếu không có n ngầm định là dòng hiện tại)

Ngoài các lệnh trên, khi sử dụng **vim** trong **xterm**, người dùng có thể sử dụng chuột để thực hiện các thao tác cho việc sao chép văn bản. Việc này chỉ thực hiện được khi đang ở trong chế độ soạn thảo của **vim**. Nhấn phím trái chuột và kéo từ điểm bắt đầu đến điểm kết thúc của đoạn văn bản cần sao chép. Đoạn văn bản đó sẽ được tự động sao vào bộ nhớ đệm. Sau đó di trỏ soạn thảo đến vị trí cần dán và nháy nút chuột giữa, văn bản sẽ được dán vào vị trí muốn.

Để di chuyển văn bản trong **vim**, cũng phải thực hiện qua ba bước sau:

- Cắt đoạn văn bản và dán vào bộ đệm
- Di chuyển dấu nhắc trỏ tới vị trí mới của đoạn văn bản
- Dán đoạn văn bản vào vị trí mới

Di chuyển văn bản chỉ khác sao chép ở bước đầu tiên là bước cắt đoạn văn bản. hãy sử dụng các lệnh xoá trong **vim** để cắt đoạn văn bản. Ví dụ, khi dùng lệnh **dd**, dòng bị xoá sẽ được lưu vào trong bộ đệm, khi đó có thể sử dụng các lệnh dán để dán văn bản vào vị trí mới.

Ngoài ra còn có thể sử dụng một số lệnh sau:

: [n] m [x]	di chuyển dòng thứ n vào dưới dòng thứ x
' '	dịch chuyển đến vị trí lúc trước
' "	dịch chuyển đến vị trí lúc trước thực hiện việc hiệu chỉnh file

B.3.6. Tìm kiếm và thay thế văn bản trong vim

vim có một số các lệnh tìm kiếm như sau:

/ <xâu>	tìm xâu từ dòng hiện tại đến dòng cuối trong file
? <xâu>	tìm xâu từ dòng hiện tại ngược lên dòng đầu trong file
N	tìm tiếp xâu được đưa ra trong lệnh / hoặc ? (từ trên xuống dưới)
N	tìm tiếp xâu được đưa ra trong lệnh / hoặc ? (từ dưới lên trên)

Xâu được tìm kiếm trong lệnh **/** hay **?** có thể là một biểu thức. Một biểu thức thông thường là một tập các ký tự. Tập ký tự này được xây dựng bằng cách kết hợp giữa các ký tự thông thường và các ký tự đặc biệt. Các ký tự đặc biệt trong biểu thức thường là:

.	thay thế cho một ký tự đơn ngoại trừ ký tự xuống dòng
\	để hiển thị các ký tự đặc biệt
*	thay thế cho 0 hoặc nhiều ký tự
\+	thay thế cho 1 hoặc nhiều ký tự
\=	thay thế cho 0 hoặc một ký tự
^	thay thế cho ký tự đầu dòng
$	thay thế cho ký tự cuối dòng
\ <	thay thế cho chữ bắt đầu của từ

\ >	thay thế cho chữ cuối của từ
[]	thay thế cho một ký tự nằm trong cặp dấu []
[^]	thay thế cho ký tự không thuộc trong cặp dấu [] và đứng sau dấu ^
[-]	thay thế cho một tập có thứ tự các ký tự
\p	thay thế cho một ký tự có thể in được
\s	thay thế cho một ký tự trống
\e	thay thế cho phím Esc
\t	thay thế cho phím Tab

vim sử dụng chế độ lệnh **Ex** để thực hiện các việc tìm kiếm và thay thế. Tất cả các lệnh trong chế độ này được bắt đầu bằng dấu '**:**'. có thể kết hợp lệnh tìm kiếm và thay thay thế để đưa ra được các lệnh phức tạp theo dạng tổng quát sau:

:<điểm bắt đầu>,<điểm kết thúc> s/<mẫu cần thay thế>/<mẫu được thay thế>/[g][c]

Ví dụ, lệnh sau đây:

:1,$s/the/The/g

tìm trong file đang soạn thảo các từ **the** và thay chúng bởi các từ **The**.

B.3.7. Đánh dấu trong vim

m |a-zA-Z|	đánh dấu văn bản tại vị trí hiện thời với dấu là các chữ cái |a-zA-Z|
'|a-z|	dịch chuyển con trỏ tới vị trí đã được đánh dấu bởi các chữ cái |a-z| trong phạm Vim file hiện thời
'|A-Z|	dịch chuyển con trỏ tới vị trí đã được đánh dấu bởi các chữ cái |A-Z| trong một file bất kỳ
:marks	hiển thị các đánh dấu hiện thời

B.3.8. Các phím sử dụng trong chế độ chèn

	Chuyển đổi chế độ
<Insert>	chuyển vào chế độ chèn hoặc chế độ thay thế
<Esc>	thoát khỏi chế độ chèn, trở lại chế độ thông thường
CTRL-C	giống như <Esc>, nhưng ???
CTRL-O <lệnh>	thực hiện <lệnh> và trở về chế độ chèn
	Di chuyển
Các phím mũi tên	di chuyển trỏ soạn thảo sang trái/phải/lên/xuống một ký tự
SHIFT-left/right	di chuyển trỏ soạn thảo sang trái/phải một từ
<Home>	di chuyển trỏ soạn thảo về đầu dòng
<End>	di chuyển trỏ soạn thảo về cuối dòng
	Các phím đặc biệt
<Enter>, CTRL-M, CTRL-J	bắt đầu một dòng mới
CTRL-E	chèn ký tự vào bên phải dấu nhắc trỏ
CTRL-Y	chèn một ký tự vào bên trái dấu nhắc trỏ
CTRL-A	chèn vào trước đoạn văn bản được chèn
CTRL-ừ	chèn vào trước đoạn văn bản được chèn và dừng chế độ chèn

CTRL-R <thanh ghi>	chèn nội dung của một thanh ghi
CTRL-N	chèn từ tiếp theo vào trước dấu nhắc trỏ
CTRL-P	chèn từ trước đó vào trước dấu nhắc trỏ
CTRL-X ...	hoàn thành từ trước dấu nhắc trỏ theo nhiều cách khác nhau
<Backspace>, CTRL-H	xoá một ký tự trước dấu nhắc trỏ
<Del>	xoá một ký tự sau dấu nhắc trỏ
CTRL-W	xoá từ trước dấu nhắc trỏ
CTRL-U	xoá tất cả các ký tự trên dòng hiện tại
CTRL-T	chèn một khoảng trống trước dòng hiện thời
CTRL-D	xoá một khoảng trống trước dòng hiện thời

B.3.9. Một số lệnh trong chế độ ảo

	v	khi nhấn phím này, có thể sử dụng các phím di chuyển để đánh dấu đoạn văn bản hoặc bỏ đánh dấu (văn bản được đánh dấu có màu trắng)
	V	khi nhấn phím này, một dòng văn bản sẽ được đánh dấu và có thể sử dụng các phím di chuyển để đánh dấu đoạn văn bản hoặc bỏ đánh dấu
	CTRL-V	nhấn phím này sẽ đánh dấu một khối văn bản và có thể sử dụng các phím di chuyển để đánh dấu đoạn văn bản hoặc bỏ đánh dấu
	o	di chuyển vị trí dấu nhắc trỏ trên khối được đánh dấu hoặc bỏ đánh dấu
	gv	đánh dấu lại đoạn văn bản được đánh dấu lúc trước
n	aw	chọn đánh dấu n từ
n	as	chọn đánh dấu n câu
n	ap	chọn đánh dấu n đoạn
n	ab	chọn đánh dấu n khối

B.3.10. Các lệnh lặp

n	.	lặp lại n lần thay đổi cuối
	q |a-z|	ghi các ký tự được nhập vào trong thanh ghi |a-z|
n	@|a-z|	thực hiện nội dung có trong thanh ghi |a-z| n lần
n	@@	lặp lại n lần sự thực hiện của lệnh @|a-z| trước
	:@|a-z|	thực hiện nội dung của thanh ghi |a-z| như một lệnh Ex
	:@@	lặp lại sự thực hiện của lệnh :@|a-z| trước
	:[n,m]g/mẫu/[lệnh]	thực hiện lệnh (mặc định là :p) trên các dòng có chứa mẫu nằm trong khoảng từ dòng thứ n đến dòng thứ m
	:[n,m]g!/<mẫu>/[lệnh]	thực hiện lệnh (mặc định là :p) trên các dòng không chứa mẫu nằm trong khoảng từ dòng thứ n đến dòng thứ m
	:sl [n]	tạm dừng trong n giây
n	gs	tiếp tục dừng trong n giây

B.4. Các lệnh khác

B.4.1. Cách thực hiện các lệnh bên trong Vim

:sh	khởi tạo một shell

:! <lệnh>	thực hiện một lệnh shell trong Vim
:!!	lặp lại lệnh ':! <lệnh>' lúc trước
K	mở trang man của lệnh trùng với nội dung từ tại dấu nhắc trỏ
q	thoát khỏi lệnh đang thực hiện trở lại Vim

B.4.2. Các lệnh liên quan đến file

Ngoài các lệnh cơ bản như sao chép hay cắt dán, trong **vim** còn có một số lệnh cho phép có thể có được những thông tin cần thiết về file.

	CTRL-G	hiển thị tên file hiện thời kèm theo trạng thái file và vị trí dấu nhắc trỏ (trạng thái có thể là: chỉ đọc, được sửa, lỗi khi đọc, file mới) (giống **:f**)
n	CTRL-G	hiển thị thông tin như CTRL -G và có thêm đường dẫn đầy đủ của file (nếu n>1, tên buffer hiện thời sẽ được đưa ra)
g	CTRL-G	đưa ra vị trí dấu nhắc trỏ theo dạng: cột/tổng số cột, dòng/tổng số dòng và ký tự/tổng số ký tự
	:f <tên mới>	đổi tên file hiện thời thành tên mới
	:ls	liệt kê tất cả các file hiện thời đang được sử dụng trong Vim (giống **:buffer** và **:files**)
	:cd	đưa thêm đường dẫn vào tên file
	:w <tên file>	tạo một bản sao của file hiện thời với tên mới là tên file (giống như save as trong Win)
		Xác định file cần soạn thảo
	:e[n, /mẫu] <file>	soạn thảo file, từ dòng thứ n hoặc từ dòng có chứa mẫu, trừ khi có sự thay đổi thực sự trong file
	:e[n, /mẫu]! <file>	luôn soạn thảo file, từ dòng thứ n hoặc từ dòng có chứa mẫu, bỏ qua mọi sự thay đổi trong file
	:e	nạp lại file hiện thời, trừ khi có sự thay đổi thực sự trong file
	:e!	luôn nạp lại file hiện thời, bỏ qua mọi sự thay đổi thực sự trong file
	:fin [!] <file>	tìm file trên đường dẫn và soạn thảo
	:e #n	soạn thảo file thứ n (giống n CTRL-^)
		Các lệnh khác
	:pw	đưa ra tên thư mục hiện thời
	:conf <lệnh trong **vim**>	thực hiện lệnh trong **vim** và đưa ra hộp thoại yêu cầu xác nhận khi có thao tác đòi hỏi sự xác nhận

PHỤ LỤC C. MIDNIGHT COMMANDER

C.1. Giới thiệu về Midnight Commander (MC)

Người sử dụng hệ điều hành MS-DOS đều biết tính năng tiện ích Norton Commander (NC) rất mạnh trong quản lý, điều khiển các thao tác về file, thư mục, đĩa cũng như là môi trường trực quan trong chế độ văn bản (text). Dù trong hệ điều hành Windows sau này đã có sự hỗ trợ của tiện ích Explorer nhưng không vì thế mà vai trò của NC giảm đi: Nhiều người dùng vẫn thích dùng NC trong các thao tác với file và thư mục. Linux cũng có một tiện ích mang tên Midnight Commander (viết tắt là MC) có chức năng và giao diện gần giống với NC của MS-DOS và sử dụng MC trong Linux tương tự như sử dụng NC trong MS-DOS.

C.2. Khởi động MC

Lệnh khởi động MC:

 # mc [Tùy-chọn]

Có một số tuỳ chọn khi dùng tiện ích này theo một số dạng thông dụng sau:

-a	Không sử dụng các ký tự đồ hoạ để vẽ các đường thẳng khung.
-b	Khởi động trong chế độ màn hình đen trắng.
-c	Khởi động trong chế độ màn hình màu.
-d	Không hỗ trợ chuột
-P	Với tham số này, Midnight Commander sẽ tự động chuyển thư mục hiện hành tới thư mục đang làm việc. Như vậy, sau khi kết thúc, thư mục hiện hành sẽ là thư mục cuối cùng thao tác.
-v file	Sử dụng chức năng View của MC để xem nội dung của file được chỉ ra.
-V	Cho biết phiên bản chương trình đang sử dụng.

Nếu chỉ ra đường dẫn (path), đường dẫn đầu tiên là thư mục được hiển thị trong panel chọn (selected panel), đường dẫn thứ hai được hiển thị panel còn lại.

C.3. Giao diện của MC

Giao diện của MC được chia ra làm bốn phần. Phần lớn màn hình là không gian hiển thị của hai panel. Panel là một khung cửa sổ hiển thị các file thư mục cùng các thuộc tính của nó hoặc một số nội dung khác. Theo mặc định, dòng thứ hai từ dưới lên sẽ là dòng lệnh còn dòng dưới cùng hiển thị các phím chức năng. Dòng đầu tiên trên đỉnh màn hình là thực đơn ngang (menu bar) của MC. Thanh thực đơn này có thể không xuất hiện nhưng nếu kích hoạt bằng cả hai chuột tại dòng đầu tiên hoặc nhấn phím <F9> thì nó sẽ hiện ra và được kích hoạt.

Midnight Commander cho phép hiển thị cùng một lúc cả hai panel. Một trong hai panel là panel hiện hành (panel chọn). Thanh sáng chọn nằm trên panel hiện hành. Hầu hết các thao tác đều diễn ra trên Panel này. Một số các thao tác khác về file như Rename hay Copy sẽ mặc định sử dụng thư mục ở Panel còn lại làm thư mục đích. Tuy nhiên ta vẫn có thể sửa được thư mục này trước khi thao tác vì các thao tác này đầu tiên bao giờ cũng yêu cầu nhập đường dẫn. Trên panel sẽ hiển thị hầu hết các file và thư mục con của thư mục hiện hành. Midnight Commander có cơ chế hiển thị các kiểu file khác nhau bằng các ký hiệu và màu

sắc khác nhau, ví dụ như các file biểu tượng liên kết sẽ có ký hiệu '@' ở đầu, các file thiết bị sẽ có màu đỏ tím, các file đường ống có màu đen, các thư mục có ký hiệu '/' ở đầu, các thư mục liên kết có ký hiệu '~'...

Cho phép thi hành một lệnh hệ thống từ MC bằng cách gõ chúng lên màn hình. Tất cả những gì có gõ vào đều được hiển thị ở dòng lệnh phía dưới trừ một số ký tự điều khiển và khi nhấn Enter, Midnight Commander sẽ thi hành lệnh gõ vào.

C.4. Dùng chuột trong MC

Midnight Commander sẽ hỗ trợ chuột trong trường hợp không gọi với tham số **-d**. Khi kích chuột vào một file trên Panel, file đó sẽ được chọn, có nghĩa là thanh sáng chọn sẽ nằm tại vị trí file đó và panel chứa file đó sẽ trở thành panel hiện hành. Còn nếu kích chuột phải vào một file, file đó sẽ được đánh dấu hoặc xoá dấu tuỳ thuộc vào trạng thái kích trước đó.

Nếu kích đôi chuột tại một file, file đó sẽ được thi hành nếu đó là file thi hành được (executable program) hoặc nếu có một chương trình đặc trưng cho riêng phần mở rộng đó thì chương trình đặc trưng này sẽ được thực hiện.

Người dùng cũng có thể thực hiện các lệnh của các phím chức năng bằng cách nháy chuột lên phím chức năng đó.

Nếu kích chuột tại dòng đầu tiên trên khung panel, toàn bộ panel sẽ bị kéo lên. Tương tự kích chuột tại dòng cuối cùng trên khung panel, toàn bộ panel sẽ bị kéo xuống.

Có thể bỏ qua các thao tác chuột của MC và sử dụng các thao tác chuột chuẩn bằng cách giữ phím <Shift>

C.5. Các thao tác bàn phím

Một số thao tác của Midnight Commander cho phép sử dụng nhanh bằng cách gõ các phím tắt (hot key). Để tương thích với một số hệ thống khác, trong các bảng dưới đây về Midnight Commander, viết tắt phím CTRL là "C", phím ALT là "M" (Meta), phím SHIFT là "S". Các ký hiệu tổ hợp phím có dạng như sau:

C-<chr>	Có nghĩa là giữ phím CTRL trong khi gõ phím <char>. Ví dụ C -f có nghĩa là giữ CTRL và nhấn <f>.
C-<chr1><char2>	Có nghĩa là giữ phím CTRL trong khi gõ phím <char1> sau đó nhả tất cả ra và gõ phím <char2>.
M-<chr>	Có nghĩa là giữ phím ALT trong khi gõ phím <char>. Nếu không có hiệu lực thì có thể thực hiện bằng cách gõ phím <Esc> nhả ra rồi gõ phím <char>.
S-<chr>	Có nghĩa là giữ phím SHIFT trong khi gõ phím <char>.

Sau đây là chức năng một số phím thông dụng.
Các phím thực hiện lệnh:

Enter	Nếu có dòng lệnh, lệnh đó sẽ được thi hành. Còn nếu không thì sẽ tuỳ vào vị trí của thanh sáng trên panel hiện hành là file hay thư mục mà hoặc việc chuyển đổi thư mục hoặc thi hành file hay thi hành một chương trình tương ứng sẽ diễn ra.
C-l	Cập nhật lại các thông tin trên Panel.

Các phím thao tác trên dòng lệnh:

M-Enter hay C-Enter	chép tên file ở vị trí thanh sáng chọn xuống dòng lệnh
M-Tab	hoàn thành tên file, lệnh, biến, tên người dùng hoặc tên máy giúp
C-x t, C-x C-t	sao các file được đánh dấu (mặc định là file hiện thời) trên panel chọn (C-x t) hoặc trên panel kia (C-x C-t) xuống dòng lệnh
C-x p, C-x C-p	đưa tên đường dẫn hiện thời trên panel chọn (C-x p) hoặc trên panel kia (C-x C-p) xuống dòng lệnh
M-p, M-n	sử dụng để hiện lại trên dòng lệnh các lệnh đã được gọi trước đó. M-p sẽ hiện lại dòng lệnh được thi hành gần nhất, M-n hiện lại lệnh được gọi trước lệnh đó
C-a	đưa dấu nhắc trỏ về đầu dòng
C-e	đưa dấu nhắc trỏ về cuối dòng
C-b, Left	đưa dấu nhắc trỏ di chuyển sang trái một ký tự
C-f, Right	đưa dấu nhắc trỏ di chuyển sang phải một ký tự
M-f	đưa dấu nhắc trỏ đến từ tiếp theo
M-b	đưa dấu nhắc trỏ ngược lại một từ
C-h, Space	xoá ký tự trước đó
C-d, Delete	xoá ký tự tại vị trí dấu nhắc trỏ
C-@	đánh dấu để cắt
C-k	xoá các ký tự từ vị trí dấu nhắc trỏ đến cuối dòng
M-C-h, M-Backspace	xoá ngược lại một từ

Các phím thao tác trên panel:

Up,Down, PgUp, PgDown, Home, End	sử dụng các phím này để di chuyển trong một panel
b, C-b, C-h, Backspace, Delete	di chuyển ngược lại một trang màn hình
Space	di chuyển tiếp một trang màn hình
u, d	di chuyển lên/ xuống 1/2 trang màn hình
g, G	di chuyển đến điểm đầu hoặc cuối của một màn hình
Tab, C-i	hoán đổi panel hiện hành. Thanh sáng chọn sẽ chuyển từ panel cũ sang panel hiện hành
Insert, C-t	chọn đánh dấu một file hoặc thư mục
M-g, M-h, M-j	lần lượt chọn file đầu tiên, file giữa và file cuối trên panel hiển thị

	tìm kiếm file trong thư mục. Khi kích hoạt chế độ này, những ký tự gõ vào sẽ được thêm vào xâu tìm kiếm thay vì hiển thị trên dòng lệnh. Nếu tuỳ chọn Show mini-status trong option được đặt thì xâu tìm kiếm sẽ được hiển thị ở dòng trạng thái.
C-s, M-s	Khi gõ các kí tự, thanh sáng chọn sẽ di chuyển đến file đầu tiên có những ký tự đầu giống những ký tự gõ vào. Sử dụng phím Backspace hoặc Del để hiệu chỉnh sai sót. Nếu nhấn C-s lần nữa, việc tìm kiếm sẽ được tiếp tục
M-t	chuyển đổi kiểu hiển thị thông tin về file hoặc thư mục
C-\	thay đổi thư mục hiện thời
+	sử dụng dấu cộng để lựa chọn đánh dấu một nhóm file. Có thể sử dụng các ký tự đại diện như '*', '?'... để biểu diễn các file sẽ chọn
ữ	sử dụng dấu trừ để xoá đánh dấu một nhóm file. Có thể sử dụng các ký tự dại diện như '*', '?' để biểu diễn các file sẽ xoá
*	sử dụng dấu * để đánh dấu hoặc xoá đánh dấu tất cả các file trong panel
M-o	một panel sẽ hiển thị nội dung thư mục hiện thời hoặc thư mục cha của thư mục hiện thời của panel kia
M-y	di chuyển đến thư mục lúc trước đã được sử dụng
M-u	di chuyển đến thư mục tiếp theo đã được sử dụng

C.6. Thực đơn thanh ngang (menu bar)

Thực đơn thanh ngang trong Midnight Commander được hiển thị ở dòng đầu tiên trên màn hình. Mỗi khi nhấn <F9> hoặc kích chuột tại dòng dầu tiên trên màn hình thực đơn ngang sẽ được kích hoạt. Thực đơn ngang của MC có năm mục "Left", "File", "Command", "Option" và "Right".

Thực đơn Left và Right giúp ta thiết lập cũng như thay đổi kiểu hiển thị của hai panel left và right. Các thực đơn mức con của chúng gồm:

Listing Mode ...		thực đơn này được dùng khi muốn thiết lập kiểu hiển thị của các file. Có bốn kiểu hiển thị: * Full - hiển thị thông tin về tên , kích thước, và thời gian sử đổi của file; * Brief - chỉ hiển thị tên của file; * Long - hiển thị thông tin đầy đủ về file (tương tự lệnh ls -l); * User - hiển thị các thông tin do tự chọn về file;
Quick view	C-x q	xem nhanh nội dung của một file
Info	C-x i	xem các thông tin về một thư mục hoặc file
Tree		hiển thị dưới dạng cây thư mục
Sort order...		thực hiện sắp xếp nội dung hiển thị theo tên, theo tên mở rộng, thời gian sửa chữa, thời gian truy nhập, thời gian thay đổi, kích thước, inode
Filter ...		thực hiện việc lọc file theo tên
Network link ...		thực hiện liên kết đến một máy tính
FTP link ...		thực hiện việc lấy các file trên các máy từ xa

Rescan	C-r	quét lại

Thực đơn File chứa một danh sách các lệnh mà có thể thi hành trên các file đã được đánh dấu hoặc file tại vị trí thanh chọn. Các thực đơn mức con:

User menu	F2	thực đơn dành cho người dùng
View	F3	xem nội dung của file hiện thời
View file ...		mở và xem nội dung của một file bất kì
Filtered view	M-!	thực hiện một lệnh lọc với tham số là tên file và hiển thị nội dung của file đó
Edit	F4	soạn thảo file hiện thời với trình soạn thảo mặc định trên hệ thống
Copy	F5	thực hiện copy
cHmod	C-x c	thay đổi quyền truy nhập đối với một thư mục hay một file
Link	C-x l	tạo một liên kết cứng đến file hiện thời
Symlink	C-x s	tạo một liên kết tượng trưng đến file hiện thời
edit sYimlink	C-x C-s	hiệu chỉnh lại một liên kết tượng trưng
chOwn	C-x o	thay đổi quyền sở hữu đối với thư mục hay file
Advanced chown		thay đổi quyền sở hữu cũng như quyền truy nhập của file hay thư mục
Rename/Move	F6	thực hiện việc đổi tên hay di chuyển đối với một file
Mkdir	F7	tạo một thư mục
Delete	F8	xoá một hoặc nhiều file
Quick cd	M-c	chuyển nhanh đến một thư mục
select Group	M-+	thực hiện việc chọn một nhóm các file
Unselect group	M-ữ	ngược với lệnh trên
reverse selecTion	M-*	chọn các file trong thư mục hiện thời
Exit	F10	thoát khỏi MC

Thực đơn Command cũng chứa một danh sách các lệnh.

Directory tree		hiển thị thư mục dưới dạng cây thư mục
Find file	M-?	tìm một file
Swap panels	C-u	thực hiện tráo đổi nội dung giữa hai panel hiển thị
Switch panels on/of	C-o	đưa ra lệnh shell được thực hiện lần cuối (chỉ sử dụng trên xterm, trên console SCO và Linux)
Compare directories	C-x d	thực hiện so sánh thư mục hiện tại trên panel chọn với các thư mục khác
Command history		đưa ra danh sách các lệnh đã thực hiện
Directory hotlist	C-\	thay đổi thư mục hiện thời
External panelize	C-x !	thực hiện một lệnh trong MC và hiển thị kết quả trên panel chọn (ví dụ: nếu muốn trên panel chọn hiển thị tất cả các file liên kết trong thư mục hiện thời, hãy chọn mục thực đơn này và nhập lệnh find . -type l -print

		sẽ thấy kết quả thật tuyệt vời)
Show directory size		hiển thị kích thước của thư mục
Command history		hiển thị danh sách các lệnh đã thực hiện
Directory hotlist	C-\	chuyển nhanh đến một thư mục
Background	C-x j	thực hiện một số lệnh liên quan đến các quá trình nền
Extension file edit		cho phép hiệu chỉnh file ~/.mc/ext để xác định chương trình sẽ thực hiện khi xem, soạn thảo hay làm bất cứ điều gì trên các file có tên mở rộng

Thực đơn Options cho phép thiết lập, huỷ bỏ một số tuỳ chọn có liên quan đến hoạt động của chương trình MC.

Configuration ...	thiết lập các tuỳ chọn cấu hình cho MC
Lay-out ...	xác lập cách hiển thị của MC trên màn hình
Confirmation ...	thiết lập các hộp thoại xác nhận khi thực hiện một thao tác nào đó
Display bits ...	thiết lập cách hiển thị của các ký tự
Learn keys ...	xác định các phím không được kích hoạt
Virtual FS ...	thiết lập hệ thống file ảo
Save setup	ghi mọi sự thiết lập được thay đổi

C.7. Các phím chức năng

Các phím chức năng của Midnight Commander được hiển thị tại dòng cuối cùng của màn hình. Có thể thực hiện các chức năng đó bằng cách kích chuột lên nhãn của các chức năng tương ứng hoặc nhấn trên bàn phím chức năng đó.

F1	hiển thị trang trợ giúp
F2	đưa ra thực đơn người dùng
F3	xem nội dung một file
F4	soạn thảo nội dung một file
F5	thực hiện sao chép file
F6	thực hiện di chuyển hoặc đổi tên file
F7	tạo thư mục mới
F8	xoá thư mục hoặc file
F9	đưa trỏ soạn thảo lên thanh thực đơn nằm ngang
F10	thoát khỏi MC

C.8. Bộ soạn thảo của Midnight Commander

Midnight Commander cung cấp một bộ soạn thảo khá tiện dụng trong việc soạn thảo các văn bản ASCII. Bộ soạn thảo này có giao diện và thao tác khá giống với tiện ích Edit của DOS hay NcEdit của Norton Commander. Để hiệu chỉnh một số file văn bản, hãy di chuyển thanh sáng chọn đến vị trí file đó rồi nhấn F4, nội dung của file đó sẽ hiện ra trong vùng soạn thảo. Sau khi hiệu chỉnh xong, nhấn F2 để ghi lại. Bộ soạn thảo này có một thực đơn ngang cung cấp các chức năng đầy đủ như một bộ soạn thảo thông thường. Nếu đã từng là người dùng DOS và mới dùng Linux thì nên dùng bộ soạn thảo này để hiệu chỉnh và soạn thảo văn bản thay vì bộ soạn thảo Vim. Sau đây là bảng liệt kê các phím chức năng cũng như các mức thực đơn trong bộ soạn thảo này:

*** Thanh thực đơn**

Thực đơn File:

các thao tác liên quan đến file

Open/load	C-o	mở hoặc nạp một file
New	C-n	tạo một file mới
Save	F2	ghi nội dung file đã được soạn thảo
Save as ...	F12	tạo một file khác tên nhưng có nội dung trùng với nội dung file hiện thời
Insert file ...	F15	chèn nội dung một file vào file hiện thời
Copy to file ...	C-f	sao đoạn văn bản được đánh dấu đến một file khác
About ..		thông tin về bộ soạn thảo
Quit	F10	thoát khỏi bộ soạn thảo

Thực đơn Edit:

các thao tác liên quan đến việc soạn thảo nội dung file

Toggle Mark	F3	thực hiện đánh dấu một đoạn văn bản
Mark Columns	S-F3	đánh dấu theo cột
Toggle Ins/overw	Ins	chuyển đổi giữa hai chế độ chèn/đè
Copy	F5	thực hiện sao chép file
Move	F6	thực hiện di chuyển file
Delete	F8	xoá file
Undo	C-u	trở về trạng thái trước khi thực hiện một sự thay đổi
Beginning	C-PgUp	di chuyển đến đầu màn hình
End	C-PgDn	di chuyển đến cuối màn hình

Thực đơn Sear/Repl:

các thao tác liên quan đến việc tìm kiếm và thay thế

Search ..	F7	thực hiện tìm kiếm một xâu văn bản
Search again	F17	tìm kiếm tiếp
Replace ...	F4	tìm và thay thế xâu văn bản

Thực đơn Command:

Các lệnh có thể được thực hiện trong khi soạn thảo

Goto line ...	M-l	di chuyển trỏ soạn thảo đến một dòng
Insert Literal ...	C-q	chèn vào trước dấu nhắc trỏ một ký tự
Refresh screen	C-l	làm tươi lại màn hình
Insert Date/time		chèn ngày giờ hiện tại vào vị trí dấu nắhc trỏ
Format paragraph	M-p	định dạng lại đoạn văn bản
Sort	M-t	thực hiện sắp xếp

Thực đơn Options:

Các tuỳ chọn có thể thiết lập cho bộ soạn thảo

General ...	thiết lập các tuỳ chọn cho bộ soạn thảo
Save mode ...	ghi lại mọi sự thiết lập được thay đổi

*** Các phím chức năng**

F1	hiển thị trang trợ giúp
F2	ghi nội dung file
F3	thực hiện việc đánh dấu đoạn văn bản
F4	tìm và thay thế xâu văn bản
F5	thực hiện việc sao chép
F6	di chuyển file
F7	tìm kiếm xâu văn bản
F8	xoá đoạn văn bản được đánh dấu
F9	hiển thị thanh thực đơn ngang
F10	thoát khỏi bộ soạn thảo

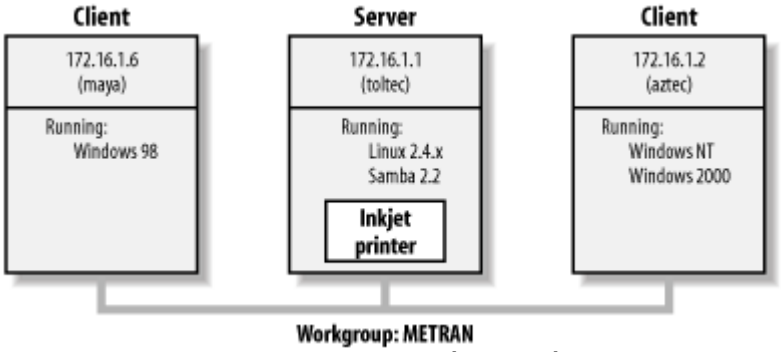
PHỤ LỤC D. SAMBA

D.1 Cài đặt Samba

Nếu các dịch vụ chia sẻ file giữa các máy Windows với nhau và giữa các máy Linux với nhau đã được giải quyết thì còn lại là vấn đề chia sẻ file giữa các máy Linux và Windows lại là một nhu cầu quan trọng. Samba hiện đã trở thành cây cầu nối giữa Linux và Windows. Samba cho phép các máy tính chạy Linux có thể hoạt động và giao tiếp trên cùng một giao thức mạng với máy Windows. Samba server thực hiện các dịch vụ sau:

- Chia sẻ một hay nhiều hệ thống file.
- Chia sẻ các máy in đã được cài đặt ở cả hai phía server và client của nó.
- Hỗ trợ các client duyệt Network Neighborhood trong các máy windows.
- Kiểm tra xác nhận các client đăng nhập vào vùng của Windows.
- Cung cấp hoặc hỗ trợ việc phân giải địa chỉ của server bằng WINS (Windows Internet Name Server).

Để dễ hình dung hơn, ta lấy ví dụ mạng đơn giản có dùng Samba. Giả sử ta có một cấu hình mạng cơ bản sau: một máy chủ Linux có sử dụng Samba với tên được đặt là Toltec, và hai client trên MS Windows, với tên là Maya và Aztec, được nối với nhau trong một mạng Lan và tham gia vào nhóm làm việc có tên METRAN. Ta còn giả sử máy chủ Toltec còn có một máy in phun tên là lp được nối tại chỗ, đồng thời việc chia sẻ đĩa cứng cho phép các máy "nhìn" thấy nhau. Mô tả mạng trên bằng hình vẽ D.1 được biểu diễn dưới đây:



Hình D.1 Một mạng có nhiều hệ điều hành

Các vai trò của Samba đối với mạng Windows NT (phiên bản Samba 2.0.4b.)

Vai Trò	*Có thể thực hiện được*
File Server	Có
Printer Server	Có
Primary Domain Controller	Có (Samba 2.1 hay muộn hơn)
Backup Domain Controller	Không
Window 95/98 Authentication	Có
Local Master Browser	Có
Local Backup Browser	Không
Domain Master Browser	Có
Primary WINS Server	Có
Secondary WINS Server	Không

D.2 Các thành phần của Samba

Samba thực chất chứa một số chương trình phục vụ cho những mục đích khác nhau nh-ng có liên quan với nhau. Hạt nhân của Samba là hai deamon có những nhiệm vụ sau:

smbd Daemon: smbd chịu trách nhiệm điều khiển các tài nguyên được chia sẻ giữa máy chủ Samba và các máy trạm của nó. Nó cung cấp các dịch vụ về file, in, và trình duyệt cho các máy trạm SMB thông qua một hay nhiều mạng. smdb xử lý tất cả các trao đổi giữa máy chủ Samba và các client mạng của nó. Ngoài ra, deamon này còn chịu trách nhiệm kiểm tra xác nhận người dùng, khoá tài nguyên, và chia sẻ dữ liệu thông qua giao thức SMB.

nmbd Deamon: nmbd là một máy chủ dịch vụ tên đơn giản bắt chước các chức năng máy chủ dịch vụ tên, chạy với các giao thức WINS và NetBIOS. Daemon này ỏlắng ngheÕ các yêu cầu của máy chủ dịch vụ tên và cung cấp các thông tin thích hợp khi được gọi tới. Nó còn cung cấp danh sách duyệt Network Neighborhood và tham gia vào lựa chọn các đối tượng mạng trong đó.

Bộ cài đặt Samba còn có một tập hợp nhỏ các công cụ dòng lệnh Linux:

smbclient: Một client Linux theo kiêu ftp có thể dùng tiện ích này để kết nối với tài nguyên được Samba chia sẻ.

Smbtar: Chương trình để lưu trữ các tài nguyên được chia sẻ, tương tự như lệnh tar của Linux.

nmblookup: Chương trình cung cấp NetBIOS thông qua việc tìm tên bằng TCP/IP.

smbpasswd: Chương trình cho phép ngươời quản trị thay đổi mật khẩu đã mã hóa của Samba.

testparm: Chương trình đơn giản để làm cho file cấu hình Samba có hiệu lực.

testprns: Chương trình kiểm tra liệu các máy in khác nhau có được deamon smbd nhận ra hay không.

Nếu muốn xem từng deamon thực hiện những gì, Samba có chương trình với tên smbstatus sẽ đa tất cả các thông tin đó lên màn hình như sau:

```
Samba version 2.2.7-security-rollup-fix
Service uid gid pid machine
----------------------------------------------
IPC$ root root 21608 httt-09 (10.10.16.5) Fri Nov 28 09:42:52 2003
No locked files
```

Việc cài đặt samba cũng khá đơn giản, ta cần chuẩn bị các package sau:

```
samba-client-xxx.rpm
samba-xxx.rpm
samba-common-xxx.rpm
```

Trong đó xxx là số hiệu phiên bản của samba. Đăng nhập với quyền root và sau đó ra lệnh:

```
#rpm –ivh samba-client-xxx.rpm samba-xxx.rpm samba-
common-xxx.rpm
```

Nếu ta không nhận lời thông báo lỗi nào cả thì quá trình cài đặt đã hoàn tất.

D.3 File cấu hình Samba

Những tên được bao trong các ngoặc vuông dùng để ký hiệu cho các phần của file cấu hình smb.conf, mô tả các chia sẻ hay dịch vụ mà Samba cung cấp. Ví dụ, các phần "test" và "homes" là các chia sẻ riêng rẽ đối với đĩa cứng; chúng chứa các tuỳ chọn được ánh xạ tới các thư mục cụ thể trên server Samba. Phần chia sẻ "printers" chứa các tuỳ chọn ánh xạ tới các máy in khác nhau của servers. Tất cả các phần được xác định trong file smb.conf, trừ phần [global, sẽ được coi như các chia sẻ đĩa cứng hoặc máy in cho những dùng kết nối với server Samba.

Các dòng vào còn lại là các tuỳ chọn riêng được quy định cụ thể cho sự chia sẻ đã được đề cập tới. Các tuỳ chọn đó có tác dụng cho tới khi bắt đầu một phần mới được ký hiệu trong cặp ngoặc vuông, hoặc cho tới điểm cuối của file smb.conf được thiết lập bằng cách gán giá trị cho chúng. Mỗi một tuỳ chọn cấu hình đều có cú pháp đơn giản: `option = value`

Cuối cùng, ta có thể dùng khoảng trống dể ngăn cách chuỗi các giá trị trong danh sách, hoặc có thể dùng các dấu phảy, Hai cách trên là tơng đơng nhau nhng ta chỉ nên dùng một cách hoặc dấu phảy hoặc khoảng trống.

Chữ viết hoa không có ý nghĩa gì đối với file cấu hính Samba, ngoại trừ trong các vị trí mà hệ điều hành được chỉ tới không cho phép viết, bởi vì hệ điều hành Linux phân biệt chữ viết thường và viết hoa.

Trong trường hợp dòng vào quá dài không thể gói gọn trong không gian mà cửa sổ dòng lệnh cho phép, ta có thể viết tiếp dòng trong file cấu hình Samba bằng cách dùng ký hiệu dấu gạch ngược "\", ví dụ:

```
comment = Su chia se dau tien la ban sao chinh cua s\
an pham phan mem Teamworks moi
```

Có thể thay đổi file smb.conf và bất kỳ tuỳ chọn nào của nó vào thời điểm bất kỳ trong khi các daemon Samba đang chạy. Theo mặc định, Samba kiểm tra file cấu hình của mình cứ 60 giây một lần để tiếp nhận cá thay đổi mới. Nếu không muốn chờ đợi lâu như vậy, bạn có thể bắt các daemon đó nạp lại bằng cách gửi tín hiệu SIGHUP tới chúng, hoặc chỉ đơn giản là khởi động lại. Ví dụ, nếu tiến trình **smbd** có PID là 893, ta có thể bắt nó đọc lại file cấu hình bằng lệnh sau đây:

```
# kill –SIGHUP 893
```

Không phải tất cả các thay đổi đều được cá máy client chấp nhận ngay. Ví dụ, các tài nhuyên chia sẻ hiện thời đang được sử dụng sẽ không được đăng ký cho đến khi các máy client cắt các nối kết rồi kết nối lại tới các tài nguyên đó. Thêm nữa, cũng sẽ không được đăng ký ngay lập tức. Điều đó giúp cho các máy client tích cực không bị ngắt nối kết một cách bất ngờ hoặc gặp phải các vấn đề không được chờ đợi về truy cập tài nguyên khi phiên làm việc vẫn đang được mở.

***Các biến*:** Samba có một tập hợp đầy đủ cá biến xác định các đặc trưng của server Samba và của cá máy client nối với nó. Mỗi một biến được bắt đầu bằng dấu phần trăm "%", tiếp theo là một ký tự đơn viết hoa hoặc viết thường và chỉ có thể được dùng bên vế phải của dòng liịnh tuỳ chọn cấu hình theo cú pháp potion = variable nh trong câu li lệnh ví dụ sau:

```
[pub]
path = /home/ftp/pub/%a
```

Ký hiệu biến %a có nghĩa đại diện cho kiến trúc của máy tính client, như WinNT để chỉ các máy tính chạy trên Windows NT, Win95 – cho máy Windows 95 hoặc 98, hay như WfWg – cho Windows for Worrgroups (Windows 3.11). Theo cách viết trên, Samba sẽ gán đường dẫn chung chỉ tới tài nguyên được chia sẻ trong phần [pub] cho các máy client chạy trên Windows NT, gán đường dẫn khác cho các máy Windows 9x, và một đường dẫn nữa cho các máy với Windows for Worrkgroups. Nói cách khác, các đường dẫn mà theo đó mỗi máy client nhận thấy tài nhuyên chia sẻ sẽ khác nhau, tuỳ thuộc vào kiến trúc của client.

Biến	Định nghĩa
Các biến của máy client	
%a	Kiến trúc của các máy client(ví dụ, Samba, wfwg, winNT, win95, hoặc UNKNOWN).
%I	Địa chỉ IP của client(ví dụ, 192.168.220.100).
%m	Tên NetBIOS của client.
%M	Tên DNS của client.
Các biến về người dùng	
%g	Nhóm chính của %u
%G	Nhóm chính của %U
%u	Thư mục home hiện thời của %u.
%U	Tên người dùng được yêu cầu trên máy client
Các biến về tài nguyên được chia sẻ	
%p	Đường dẫn cho automounter tới thư mục gốc của tài nguyên được chia sẻ, nếu thư mục đó khác với %P
%P	Th mục gốc hiện thời của tài nguyên được chia sẻ.
%S	Tên hiện thời của tài nguyên được chia sẻ.
Các biến của server	
%d	Định danh tiến trình (PID) server hiện thời.
%h	Tên host DNS của server Samba.
%L	Tên host NetBIOS của server Samba
%N	Th mục home của server Samba, lấy từ file ánh xạ (map) của **automount.**
%v	Phiên bản Samba.
Các biến khác	
%R	Mức của giao thức SMB được thoả thuận thiết lập.
%T	Ngày giờ hiện tại.

Danh sách các biến của Samba

D.4 Các phần đặc biệt của file cấu hình Samba

***Phần [global]*:** Phần [global] xuất hiện hầu như trong mọi file cấu hình Samba, thậm chí khi trong đó không có dòng lệnh bắt buộc nào. Mọi tuỳ chọn được thiết lập trong phần này đều được áp dụng đối với tất cả các tài nguyên chia sẻ khác, vì nội dung của phần này sẽ được sao chép vào các phần khác. Nhưng các phần khác, nếu cũng có các tuỳ chọn giống như trong phần [global], thì trong các phần đó, các tuỳ chọn sẽ được xác định với các giá trị mới được ghi đè lên các lên các giá trị cũ của `[global]`. Ta có thể cấu hình server

Samba, đầu tiên ta phải chú ý đến ba tuỳ chọn cấu hình cơ bản xuất hiện trong phần ửglobalứ của file cấu hình smb.conf:

```
[global]
Server configuration parameters
netbios name = HYDRA
server string = Samba %v on (%L)
workgroup = SIMPLE
```

Tuỳ chọn netbios name: cho phép đặt tên NetBIOS cho server. Ví dụ:
netbios name = DHQGHN

Giá trị mặc định cho tuỳ chọn này là tên máy của server (phần bên trái cùng của tên DNS đầy đủ). Ví dụ, tên NetBIOS mặc định của máy hut.edu.vn sẽ là HUT. Thông thường, người ta đặt tên NetBIOS khác với tên DNS hiện thời.

Việc thay đổi tên NetBIOS của server không được khuyến khích nếu không có lý do chính đáng, nếu như tên đó không phải là duy nhất vì mạng LAN được chia ra thành hai hay nhiều vùng DNS. Ví dụ, khi mạng `hut.edu.vn` bị chia thành hai vùng với các server là `hut.lythuyet.edu.vn` và hut.thuchanh.edu.vn thì tên NetBIOS cũ là `HUT` bây giờ có thể thành các tên `HUTLYTHUYET` và `HUTTHUCHANH`.

Tùy chọn server string: Thông số của server string xác định nội dung dòng chú thích sẽ xuất hiện cạnh tên của server Samba trong cả cửa sổ Network Neighborhood (khi ở chế độ Details) lẫn cửa sổ quản lý in của Microsoft Windows. Bạn có thể dùng các biến chuẩn để cung cấp thông tin cho dòng mô tả đó, ví dụ trên ta đã sử dụng hai biến là %v và %L.

Tùy chọn workgroup: Thông số của tuỳ chọn workgroup thiết lập nhóm làm biệc hiện thời, nơi mà server Samba tự thông báo cho các thành biên của mạng về mình. Các clients muốn truy cập được tài nguyên được chia sẻ trên server Samba phải cùng thuộc về một nhóm làm việc NetBIOS. Nên nhớ rằng các nhóm làm việc phải có các tên nhóm NetBIOS thực thụ, tuân theo quy tắc đặt tên NetBIOS.

Cấu hình chia sẻ đĩa cứng

Trong ví dụ ở phần trước ta đã nhắc đến rằng do chưa có tài nguyên được chia sẻ nên cửa sổ chi tiết của server hydra đang còn trống. Bây giờ ta tiếp tục làm việc cới file cấu hình Samba và tạo ra một đĩa cứng được chia sẻ còn rỗng có tên là `[data]`. Đây là các động tác cần thêm vào để đạt được kết quả vừa nêu:

```
SampleDataDrive]
comment=Data Drive
path = /export/samba/data
writable = yes
guest ok = yes
```

Tài nguyên được chia sẻ ửSampleDataDriveứ thường là đĩa cứng được Samba chia sẻ và ánh xạ tới thư mục `/export/samba/data` trên server Samba. Ta đã cho thêm một dòng vào để có chú thích mô tả tài nguyên được chia sẻ đó là Data Drive, cũng như gán cho bản thân tài nguyên đó một cái tên `SampleDataDrive`.

Tài nguyên được chia sẻ được thiết lập có quyền ghi cho các người dùng. Giá trị mặc định của tuỳ chọn này là chỉ đọc. Trong phương án không cần nức độ bảo mật chặt chẽ như ở đây, ta đặt giá trị yes cho tuỳ chọn `guest ok`, để cho bất kỳ ai cũng có thể kết nối được tới tài nguyên vừa được chia sẻ. Trên máy UNIX có cài đặt Samba ta tạo thư mục `/export/samba/data` với quyền root bằng các lệnh sau:

```
# mkdir /export/samba/data
#chmod 777 /export/samba/data
```

Bây giờ, nếu ta lại kết nối với server hydra (bằng cách kích phím chuột vào biểu tượng của server trong cửa sổ Network Neighborhood của Windows), một thư mục được chia sẻ với tên data đã xuất hiện.

Tuỳ chọn	Thông số	Chức năng	Mặc định	Phạm vi
´ Path (directory)	String (đường dẫn đến thư mục)	Đặt thư mục UNIX dùng cho chia sẻ đĩa cứng hoặc cho việc xếp hàng chờ bởi máy in được chia sẻ.	/tmp	Share
´ Guest ok (public)	Nhị phân (yes/no)	Nếu đặt là ***yes***, sẽ không dần kiểm tra xác nhận người dùng đẻ truy cập tài nguyên được chia sẻ này.	***no***	Share
´ Comment	String (xâu ký tự)	Đặt chú thích sẽ xuất hiện cùng tài nguyên được chia sẻ.	Không có	Share
´ volume	String	Đặt tên cho ổ đĩa, theo dạng của DOS.	Tên của tài nguyên được chia sẻ	Share
Read only	Nhị phân (yes/no)	Nếu là ***yes***, cho phép truy cập chỉ đọc tới tài nguyên được chia sẻ.	***yes***	Share
Writeable (write ok)	nhị phân (yes/no)	Nếu là ***no***, cho phép truy cập chỉ đọc tới tài nguyên được chia sẻ.	***no***	Share

Các tuỳ chọn cơ bản chia sẻ đĩa cứng

Các tuỳ chọn về mạng của Samba

Tuỳ chọn	Thông số	Chức năng	Mặc định	Phạm vi
hosts allow (allow hosts)	String (danh sách tên máy)	Xác định các máy có thể kết nối với Samba.	Không có	´Share
Hosts deny (deny hosts)	String (danh sách tên máy)	Xác định các máy không thể kết nối với Samba.	Không có	Share
Bind interfaces only	Nhị phân (yes/no)	Nếu đặt là ***yes***, Samba sẽ chỉ liên kết tới các giao diện xác định bởi tuỳ chọn interfaces.	´***no***	Global
socket	String (địa chỉ	Đặt địa chỉ IP để ôngheÕ,	Không có	Global

address	IP)	dùng cho trờng hợp có nhiều giao thức ảo trên servers.		

Các tuỳ chọn cấu hình mạng

Tùy chọn hosts allow: Tuỳ chọn này xác định các máy có quyền truy cập các tài nguyên được chia sẻ trên servers Samba, được viết như danh sách các máy hay địa chỉ IP của chúng, cách nhau bằng dấu phảy hoặc khoảng trống. Ta có thể đặt được một chút ít mức độ bảo mật, chỉ đơn giản bằng cách đặt địa chỉ mạng con LAN của mình vào chỗ giá trị của tuỳ chọn này. Ví dụ:

```
hosts allow = 192.168.200. localhost
```

Chú ý rằng ta đặt localhost (hoặc địa chỉ 127.0.0.1) ở vị trí sau địa chỉ của mạng con. Một trong số các lỗi thường thấy khi dùng tuỳ chọn hosts allow là cấm luôn servers Samba liên hệ với chính nó. Chương trình smbpasswd sẽ cần được kết nối với servers Samba như là một client để thay đổi mật khẩu mã hóa của người dùng. Thêm nữa, việc duyệt tại chỗ cũng đòi hỏi có được đăng nhập tại chỗ.

Sau đây là các quy tắc của Samba quy định cho việc dùng các tuỳ chọn `hosts allow` và `hosts deny`:

- Nếu không có tuỳ chọn allow hoặc deny nào được xác định trong file cấu hình smb.conf Samba sẽ cho phép các kết nối từ bất kỳ máy nào mà hệ thống Unix chấp nhận
- Nếu có các tuỳ chọn allow hoăc deny được xác trong phần ửglobalứ của file cấu hình smb.conf, chúng sẽ được áp dụng cho tất cả các tài nguyên được chia sẻ, thậm chí khi một tài nguyên nào đó có tuỳ chọn ghi đè lên được xác định.
- Nếu chỉ có tuỳ chọn allow được xác định cho một tài nguyên được chia sẻ, chỉ có các máy được liệt kê mới có quyền truy cập tài nguyên đó. Các máy khác đều bị cấm.
- Nếu chỉ có tuỳ chọn deny được xác định cho một tài nguyên được chia sẻ, mọi máy không có trong danh sách đều có quyền sử dụng tài nguyên đó.
- Nếu cả hai tuỳ chọn allow và deny được xác định, một máy đã xuất hiện trong danh sách được phép thì không có mặt trong danh sách bị cấm. Nếu không máy đó sẽ bị cấm truy cập vào tài nguyên được chia sẻ.

Chý ý: Cẩn thận tránh trờng hợp ta cho phép một máy nào đó, nhưng sau đấy lại cấm cả mạng con mà máy ấy tham gia.

hosts deny: Tuỳ chọn hosts deny xác định các máy không có quyền truy nhập tài nguyên được chia sẻ, được viết như danh sách các tên máy hoặc địa chỉ IP của chúng, cách nhau bằng dấu phẩy hay khoảng trống với cú pháp giống như đới với tuỳ chọn `hosts allow` ở trên. Ví dụ, để hạn chế truy cập tới servers từ một máy, trừ từ vùng example.com, ta có thể viết:

```
hosts deny = ALL EXCEPT.example.com
```

Giống như hosts allow, không có giá trị mặc định cho tuỳ chọn hosts deny. Nếu muốn cho phép hay cấm truy cập tới tài nguyên được chia sẻ cụ thể ta phải qua cả hai tuỳ chọn `hosts allow` và `hosts deny` trong phần ửglobalứ hay nếu có dụng thì phải ghi đè giá trị mới trong phần cấu hình cho tài nguyên được chia sẻ đó.

interfaces: Tuỳ chọn interfaces liệt kê các địa chỉ mạng mà ta muốn servers Samba nhận biết và đáp ứng. Tuỳ chọn này rất tiện lợi nếu ta muốn máy tính tham gia đồng thời nhiều mạng con. Nếu không được dùng Samba tìm giao diện mạng chính của servers (thường là card Ethernet đầu tiên) khi khởi động và tự cấu hình để hoạt động chỉ trong mạng con có giao diện mạng đó. Ta phải dùng tuỳ chọn này để bắt buộc Samba phải thực hiện mạng con khác nữa trong mạng của ta.

Giá trị của tuỳ chọn là một hay nhiều bộ gồm các đôi địa chỉ IP/ mặt nạ mạng, giống như trong ví dụ sau:

```
interfaces = 192.168.220.100/255.255.255.0 192.168.210.30/255.255.0
```

Có thể dùng định dạng mặt nạ bít CIDR nh sau:

```
interfaces = 192.168.220.100/24 192.168.210.30/24
```

Số của mặt nạ bít chỉ số đầu tiên được bật trong mặt nạ mạng, ví dụ số 24 nghĩa là 24 bít đầu tiên (trong số tất cả 32 bit) sẽ được kích hoạt, hay đông nghĩa với giá trị mặt nạ mạng 255.255.255.0. Tương tự nh vậy, số 16 tương đương với mặt nạ 255.255.0.0, và 8- với 255.0.0.0. Tuy nhiên, tuỳ chọn này có thể hoạt động không đúng nếu ta dùng DHCP (phân phối địa chỉ IP động).

Bind interfaces only: Tuỳ chọn này có thể được dùng để bắt buộc các tiến trình smbd và nmbd phục vụ các yêu cầu SMB chỉ cho các địa chỉ được xác định bởi tuỳ chọn interfaces mà thôi. Tiến trình nmbd bình thường liên kết giao diện (0.0.0.0.) trên các cổng 137 và 138 tới tất cả các địa chỉ, cho phép chúng nhận các thông báo phân phối công cộng từ khắp mọi nơi. Tuy nhiên, nếu ta ghi đè lên giá trị đó bằng:

```
bind interfaces only = yes
```

Thì chỉ các gói đi từ các địa chỉ nguồn xác thông qua tuỳ chọn interfaces mới được chấp nhận. Với smbd, tuỳ chọn này cũng bắt Samba không phục vụ các yêu cầu về file của các mạng con ngoài danh sách của tuỳ chọn interfaces. Nếu muốn cho phép có các nối kết mạng tạm thời, như dụng SLIP hoặc ppp, ta không được dụng tuỳ chọn này. Nói chung, tuỳ chọn này ít được dùng, và thường chỉ có nhng người quản trị đầy kinh nghiệm mới để ý tới nó.

Nếu đặt giá trị cho bind intefaces only là yes, ta phải thêm địa chỉ của máy tại chỗ (127.0.0.1) vào danh sách của interfaces, nếu không smbpasswd sẽ không thể hoạt động được.

socket address: Tuỳ chọn socket address quy định địa chỉ nào trong số được xác định bởi interfaces sẽ "ghe" tức là chờ các kết nối. Samba theo mặc định chấp nhận tất cả các nối kết với tất cả các địa chỉ. Khi được dùng trong file smb.conf, tuỳ chọn này hạn chế số địa chỉ mà Samba sẽ dùng để chờ các nối kết. Vi dụ:

```
Interfaces = 192.168.220.100/24 192.168.210.30/24
Socket address = 192.168.210.30
```

Bình thường, tuỳ chọn này không được khuyên dùng.

Nếu như có dùng các mật khẩu đã mã hoá, ta phải thêm vào một dòng có nội dung `encrypt passwords=yes` vào file cấu hình trên.

Sau khi đã soạn thảo nội dung như trên của file smb.conf và đặt nó vào đúng vị trí cần thiết, ta khởi động lại server Samba và dùng các máy client Windows để kiểm tra kết quả.

Tất nhiên các máy client Windows đó cũng phải thuộc về nhóm SIMPLE – trong ví dụ ta vẫn dùng từ đầu chơng – đó là các máy phoenix và chimaera.

Mọi tuỳ chọn xuất hiện trớc phần được đánh dấu bằng ngoặc vuông "[]"ầu tiên, tức là bên ngoài phần [lobal] cũng được coi là những tuỳ chọn chung.

Phần [mes] Nếu một client nào đó cố gắng kết nối tới tài nguyên được chia sẻ không được nêu trong file cấu hình smb.conf, Samba sẽ tìm tài nguyên được chia sẻ ửhomesứ trong file cấu hình. Nếu phần này tồn tại, tên của tài nguyên được chia sẻ không xác định kia sẽ được coi như tên người dùng của Linux và được yêu cầu tìm trong cơ sở dữ liệu mật khẩu của server Samba. Nếu như có tên ngưười dùng đó, Samba coi máy được nói tới ở trên là một ngưười dùng Linux đang cố kết nối tới th mục home của mình trong server.

Ví dụ, giả sử một máy client kết nối với server Samba hydra lần đầu tiên, và cố truy cập tới tài nguyên được chia sẻ có tên là [dung]. Trong file smb.conf, không có tài nguyên được chia sẻ nào tên là ửdungứ được xác định, nhng lại có phần [homes], vì thế Samba tìm file cơ sở dữ liệu mật khẩu và tìm xem có tài khoản ngưười dùng dung trong hệ thống hay không, Sau đó Samba kiểm tra mầy khẩu được client cung cấp và so sánh với mật khẩu của ngưười dùng Linux dung - hoặc trong file cơ sở dữ liệu mật khẩu nếu dùng mật khẩu mã hoá. Nếu các mật khẩu đó trùng nhau, Samba nhận biết chắc là ngưười dùng dung có quyền và đang muốn kết nối tới th mục home của mình trong máy Linux. Sau đó Samba sẽ tự tạo tài nguyên được chia sẻ được gọi là ửdungứ cho người dùng dung.

Người ta cũng áp dụng phơng pháp được thực hiện với phần [homes] để tạo tài khoản người dùng mới, kèm theo mật khẩu.

Phần [printers]: Phần đặc biệt thứ ba gọi là [printers] tương tự như phần [homes]. Nếu một client cố kết nối tới tài nguyên được chia sẻ không có mặt trong file cấu hình smb.conf file, và nếu tên của nó không thể tìm được trong file mật khẩu, Samba sẽ kiểm tra xem nó có phải sự chia sẻ máy in cho client đó. Samba thực hiện điều đó thông qua việc đọc file dữ liệu máy in (thường là /etc/printcap hay /etc/terminfo) để xem có tên của tài nguyên được chia sẻ đó hay không. Nếu có, Samba tạo ra tài nguyên được chia sẻ với tên liên quan tới việc chia sẻ máy in. Để có thể in được trong Samba ta phải thêm các tuỳ chọn printer driver, printer driver file, và printer driver location vào file cấu hình smb.conf của Samba. Tuỳ chọn chung printer driver file chỉ đến file printers.def phải được đặt vào phần [global]. Các tuỳ chọn còn lại được đặt vào phần tài nguyên máy in được chia sẻ mà ta muốn cấu hình một cách tự động các trình điều khiển máy in. Giá tri cho printer driver phải trùng với xâu được hiện ra trong *Printer Winzard* trên hệ thống *Windows*. Giá trị của printer driver location là đ-ờng dẫn của tài nguyên PRINTER$ mà ta thiết lập, chứ không phải là đờng dẫn UNIX trên server. Do đó, ta có thể dùng các dòng mã sau đây trong file cấu hình Samba:

```
[global]
printer driver file = /usr/local/samba/print/printers.def
[hpdeskjet]
path = /var/spool/samba/printers
printable = yes
printer driver = HP DeskJet 560C Printer
printer driver location = \\%L\PRINTER$
```

Giống như đối với phần [home], ta không cần phải bảo trì tài nguyên được chia sẻ cho mỗi một may in của hệ thống trong file cấu hình smb.conf. Thực vậy, Samba luôn dựa vào

việc đăng ký máy in của Linux nếu ta cần đến, và cung cấp các máy in đã đăng ký cho các client. Tuy nhiên, có một hạn chế nhỏ: nếu tài khoản ngưười dùng và máy in đều có tên là hai, Samba bao giờ cũng tìm tài khoản ngưười dùng trớc tiên, bất kể là client thực ra là cần kết nối với máy in.

Các chi tiết vè việc thiết lập tài nguyên được chia sẻ ửprintersứ được trình bày trong phần liên quan tới việc in và phân giải tên.

Các tuỳ chọn cấu hình: Các tuỳ chọn trong file cấu hình Samba được chia sẻ làm hai loại: global (toàn cục) và share (chia sẻ). Mỗi một loại quy định một tuỳ chọn sẽ được xuất hiện ở đâu trong file cấu hình.

Global (toàn cục): Các tuỳ chọn `global` phải có mặt chỉ trong phần `[global]` mà thôi. Đây là các tuỳ chọn thường chỉ ps dụng để xác định hoạt động của chính server Samba.

Share: Các tuỳ chọn `share` có thể xuất hiện trong các tài nguyên được chia sẻ cụ thể, hoặc cả trong phần `[global]`. Nếu có mặt trong phần `[global]`, chúng sẽ xác định các giá trị mặc định cho tất cả các tài nguyên được chia sẻ, chừng nào cha bị các tuỳ chọn cùng tên tại các phần tài nguyên được chia sẻ cụ thể ghi đè những giá trị mới.

D.5 Quản lý người dùng trong Samba

Samba có khả năng quản lý người dùng có khả năng truy cập vào máy chủ Samba. Nó có khả năng quản lý người dùng khá độc lập với hệ thống người dùng hệ thống. Thông thường các thông tin về người dùng sẽ được lưu trong file smbpasswd, file này nằm trong thư mục /etc/samba. Để thêm một người dùng cho samba quản lý, người dùng đó phải là một người dùng trong hệ thống. Sau đó, để thao tác với những người dùng của samba, ta có công cụ smbpasswd.

`smbpasswd [-a][-x][-d][-e][-h][-s] [` `tên người dùng` **`]`**

Trong đó,

- a : tuỳ chọn này cho phép ta thêm một người dùng mới vào trong danh sách người dùng của samba.
- x : tuỳ chọn này cho phép xoá bỏ một người dùng trong danh sách người dùng của samba.
- d : tuỳ chọn này cho phép ta khoá (disable) một người dùng trong danh sách người dùng của samba.
- e : tuỳ chọn này cho phép ta mở khoá (enable) một người dùng trong danh sách người dùng của samba mà người dùng đó đã bị khoá bằng tham số -d.
- <tên người dùng>: tên của người dùng ta muốn xử lý.

Chẳng hạn, muốn thêm một người dùng vào trong danh sách người dùng của samba, ta dùng lệnh (sử dụng lệnh này với quyền root):

`#smbpasswd –a thanhnt`

Trong đó người dùng thanhnt phải là một người dùng hệ thống. Sau khi đánh lệnh này, máy sẽ hỏi ta đánh vào mật khẩu cho người dùng mới này, và samba cho phép người dùng do nó quản lý có thể có mật khẩu khác với mật khẩu hệ thống của người dùng đó.

New SMB password:
Retype new SMB password:
Password changed for user thanhnt.

Lưu ý là mật khẩu sẽ được hỏi hai lần để đảm bảo tính chính xác và mật khẩu sẽ không được hiển thị ra màn hình. Nếu thành công thì ta sẽ nhận được thông báo như trên. Ta cũng có thể dùng lệnh này để thay đổi mật khẩu của một người dùng bằng lệnh (thực hiện bằng quyền root):

```
#smbpasswd thanhnt
```

Khi đó nó sẽ thông báo cho ta nhập mật khẩu hai lần giống như trên. Còn trong trường hợp là một người dùng bình thường thì muốn thay đổi mật khẩu samba cho chính người dùng đó ta chỉ cần đánh:

```
#smbpasswd
Old SMB password:
New SMB password:
Retype new SMB password:
Mismatch - password unchanged.
Unable to get new password.
```

Trong trường hợp trên, máy sẽ yêu cầu ta nhập mật khẩu cũ trước khi nhập mật khẩu mới, nếu có sai sót (mật khẩu cũ không đúng hoặc mật khẩu mới không khớp nhau) thì ta sẽ nhận được thông báo lỗi.

Nếu muốn xoá người dùng ra khỏi danh sách người dùng thì sử dụng lệnh (với quyền root):

```
#smbpasswd –x thanhnt
```

Còn nếu muốn một người dùng trong danh sách vẫn tồn tại nhưng không có hiệu lực, thì ta có thể khoá người dùng đó bằng lệnh:

```
#smbpasswd –d thanhnt
```

Khi đó người dùng thanhnt tuy vẫn còn nằm trong danh sách nhưng không được samba coi là người dùng hợp lệ nữa. Khi muốn khôi phục người dùng này có các quyền như ban đầu thì ta có thể khôi phục bằng lệnh:

```
#smbpasswd –e thanhnt
```

D.6 Cách sử dụng Samba từ các máy trạm

D.6.1 Cách sử dụng từ các máy trạm là Linux

Samba có cung cấp một công cụ nhằm sử dụng các thư mục chia sẻ theo giao thức SMB trong mạng LAN, đó chính là smbclient. Với công cụ này ta có thể thao tác với tài nguyên được chia sẻ trên mạng, chẳng hạn như kết nối vào một thư mục chia sẻ trên một máy nào đó để thao tác, sao chép file từ thư mục đó. smbclient cũng giống như một chương trình client ftp.

```
smbclient <tên dịch vụ> [-U <tên người dùng> ] [ -W <tên
miền hoặc group> ] –L [<tên netbios>]
```

Trong đó:

- <tên dịch vụ> : là tên của dịch vụ muốn sử dụng, có dạng //Maychu/dichvu. ***Maychu*** là tên netbios của máy chủ cung cấp dịch vụ, còn ***dichvu*** là tên của dịch vụ muốn sử dụng. Chẳng hạn như //dulieu/setups, thì tên máy chủ cần truy nhập là dulieu, còn

setups là tên thư mục muốn tham chiếu đến. Ta cũng có thể sử dụng địa chỉ IP thay cho tên netbios dưới dạng //192.168.0.12/setups.

- U <tên người dùng> : là tên người dùng muốn sử dụng tài nguyên đó.
- W <tên miền hoặc group> : là tên miền hoặc group mà máy chủ đó thuộc vào.
- L <tên netbios> : là tên netbios của máy chủ ta muốn xem các dịch vụ mà máy chủ đó đang cung cấp.

Ví dụ, để xem thông tin về các thư mục chia sẻ của một máy đồng thời cùng với các thông tin về các máy trong miền, các máy miền khác ta dùng lệnh:

```
# smbclient -L 10.10.16.5 -U thanhnt -W httt
```

Thì máy sẽ hỏi ta mật khẩu ứng với người dùng trên, sau khi đánh đúng mật khẩu ta sẽ thu được kết quả:

```
added interface ip=10.10.16.23 bcast=10.10.255.255 nmask=255.255.0.0
Password:
Domain=ửCHTTTMPIứ OS=ửUnixứ Server=ửSamba 2.2.3aứ
 Sharename       Type  Comment
 ---------       ----  -------
 netlogon        Disk  Network Logon Service
 public    Disk  Public Stuff
 Source    Disk  Source and documents for vietseek
 IPC$            IPC   IPC Service (Samba Server)
 ADMIN$ Disk   IPC Service (Samba Server)
 thanhnt   Disk  Home Directories

 Server   Comment
 ---------   -------
 HTTT-23 Samba Server

 Workgroup            Master
 ---------            -------
 BCNK.FOTECH      VINHTQ
 BMVT             NGUYENHONG
 CHTTTMPI         HTTT-23
 ECC                  HUNGTN
 FOTECH               ANHNV
 FOTECH-CTSV      MAIPT
```

Để sử dụng một dịch vụ (một thư mục chia sẻ chẳng hạn) ta có thể dùng lệnh như sau:

```
# smbclient //10.10.16.5/setup -U thanhnt -W httt
```

Trong trường hợp này ta sẽ được máy hỏi mật khẩu, nếu thành công thì nó sẽ cho ta một phiên làm việc với dịch vụ đó, cụ thể ta sẽ được một phiên làm việc với thư mục, ta có thể sao chép file ở trên thư mục này vào máy hiện tại và ngược lại.

```
added interface ip=10.10.16.23 bcast=10.10.255.255
nmask=255.255.0.0
Password:
Domain=[CHTTTMPI] OS=[Unix] Server=[Samba 2.2.3a]
```

```
smb: \>
smb: \> ls
  .           D    0    Tue Sep 11 12:03:53 2001
  ..          D    0    Tue Sep 11 12:03:53 2001
  ee120-ta    D    0    Wed Aug 29 09:37:14 2001
  fa01        D    0    Fri Sep 21 09:47:34 2001

  60472 blocks of size 2097152. 52606 blocks available
  smb: \> cd fa01\ee120-kmm
  smb: \> put hello.p [send files from local to remote]
  smb: \> get interruptq.doc [receive files to local from
remote]
  smb: \> quit
```

Khi dấu nhắc hiện ra, để xem các lệnh thao tác, ta có thể đánh lệnh help. Sau khi kết thúc phiên làm việc, ta dùng lệnh ***quit*** để thoát.

Kết gắn một thư mục chia sẻ vào một thư mục trong hệ thống file hiện tại: Trong trường hợp ta không muốn dùng các lệnh smbclient cho từng phiên làm việc khi mà ta sẽ có nhiều thao tác với thư mục được chia sẻ đó, giải pháp tốt nhất là kết gắn thư mục chia sẻ đó vào thành một thư mục ở trên máy cục bộ. Khi đó thư mục được kết gắn sẽ trở thành một thư mục bình đẳng như các thư mục trên máy cục bộ. Mọi việc thao tác sẽ trở nên thuận tiện hơn rất nhiều. Để làm điều đó ta dùng lệnh (với quyền root):

#smbmount //10.10.16.5/setup /mnt/smb –o username=thanhnt

hoặc

#mount –t smbfs //10.10.16.5/setup /mnt/smb –o username=thanhnt

Khi đó máy sẽ hỏi mật khẩu, khi thành công thì ta sẽ ánh xạ được thư mục chia sẻ setup trên máy 10.10.16.5 thành một thư mục /mnt/smb trên máy của mình. Khi nào xong ta có thể bỏ kết gắn đó bằng lệnh:

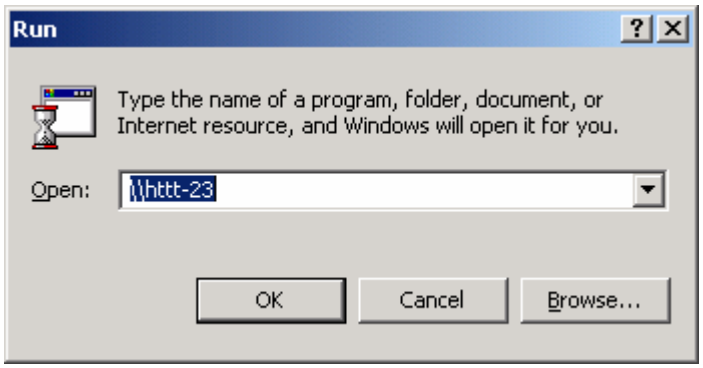
#smbmount /mnt/smb

hoặc

#umount /mnt/smb

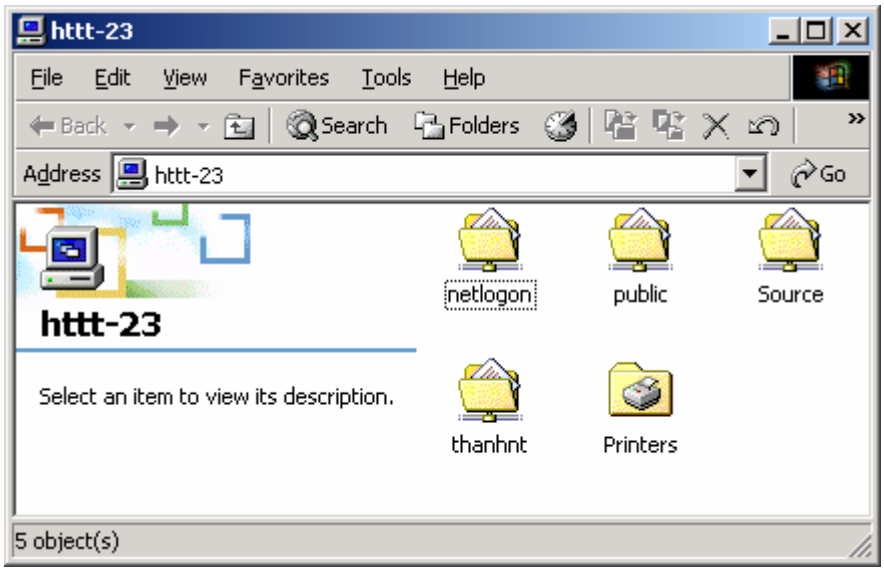
D.6.2 Cách sử dụng từ các máy trạm là Windows

Ta chọn menu start, ta chọn run, sau đó đánh vào tên máy mà ta muốn sử dụng một dịch vụ nào đó như trên hình D.2.



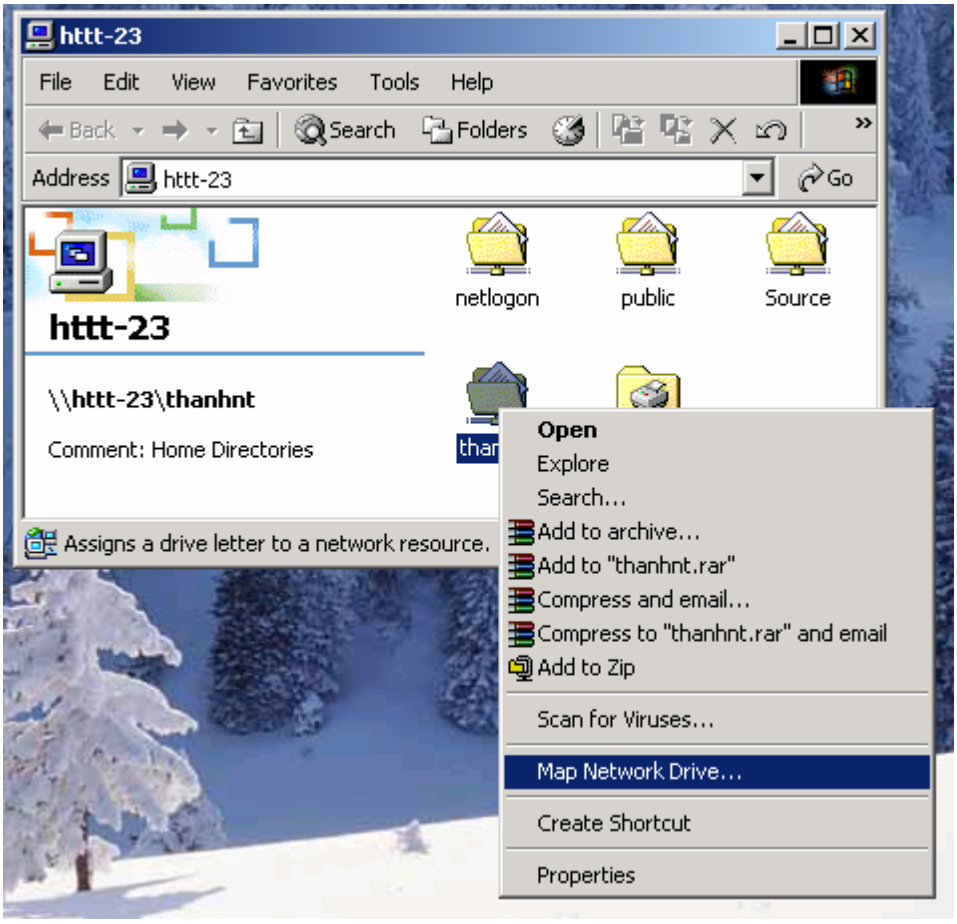
Hình D.2 Sử dụng dịch vụ samba từ máy trạm Windows

Sau đó máy sẽ hỏi ta tên người dùng và mật khẩu dùng để truy cập. Sau khi nhập đủ các thông tin, nếu thành công thì ta sẽ được một cửa sổ hiển thị danh sách các dịch vụ của máy chủ samba đó cung cấp như hình D.3.



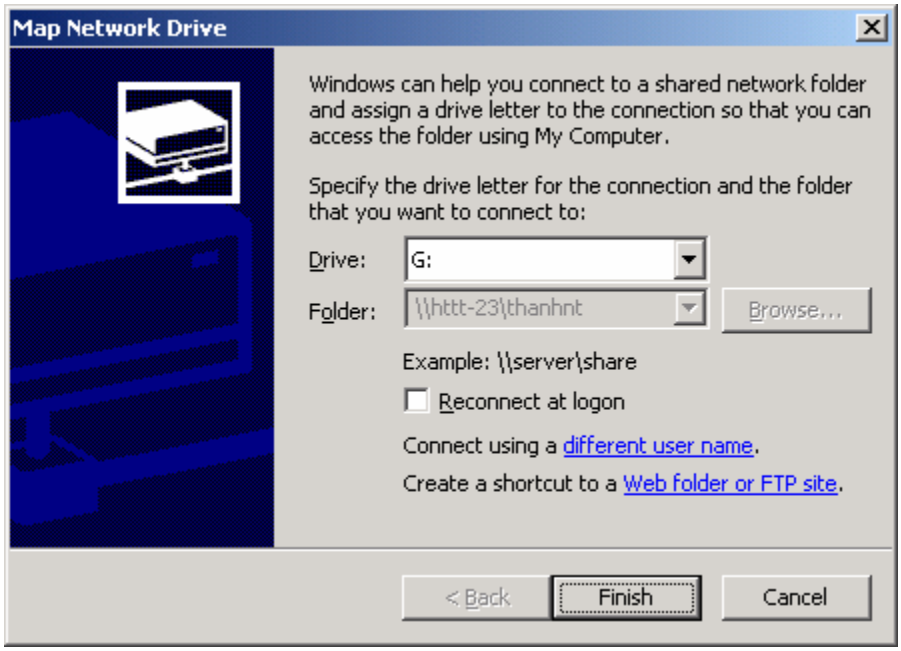
Hình D.3 Danh sách các dịch vụ trên một samba server

Cho phép ánh xạ một thư mục trên một samba server thành một ổ đĩa trêm máy trạm Windows bằng cách trên cửa sổ hiện danh sách các tài nguyên trên ta nháy phải chuột vào thư mục ta muốn ánh xạ, sau đó chọn "Map network drive" như trên hình D.4.



Hình D.4 Tạo một ánh xạ ổ đĩa trên máy trạm Windows

Sau đó máy sẽ hỏi tên ổ đĩa mà ta muốn đặt cho ổ mới này như hình D.5:



Hình D.5 Đặt tên ổ đĩa cho một ánh xạ ổ đĩa