BAN CHỈ ĐẠO CÔNG NGHỆ THÔNG TIN CỦA CƠ QUAN ĐẢNG

***

TÀI LIỆU THAM KHẢO

HỆ ĐIỀU HÀNH LINUX

HÀ NỘI, 2003

Chương 5. Các lệnh quản lý thiết bị ngoại vi

Chương 8. Midnight Commander

Chương 9. Mtools - tiện ích truy cập ổ đĩa DOS trong Linux

CHƯƠNG 1. GIỚI THIỆU CHUNG VỀ LỆNH TRONG LINUX

1.1. Giới thiệu về UNIX và Linux

1.1.1. Sơ bô về hê điều hành đa người dùng

UNIX (và Linux) là hệ điều hành đa người dùng (multi-users). Hệ điều hành đa người dùng thuộc vào loại hệ điều hành đa chương trình định hướng "thân thiện với người dùng". Tại cùng một thời điểm, có nhiều người dùng cùng sử dụng máy tính và đối với mỗi người dùng như vậy đều có cảm giác như được sử dụng máy tính một cách "độc quyền" vì họ được trực tiếp liên kết với chương trình của mình đang thực hiện trong máy tính. Điều này tương ứng với một chức năng của hệ điều hành là "hệ điều hành như một máy tính ảo" theo góc độ của người sử dụng.

Như vậy, trong máy tính đồng thời xuất hiện nhiều chương trình người dùng, các chương trình này chia nhau sử dụng các tài nguyên của hệ thống, trong đó có các tài nguyên quan trọng nhất là CPU, bộ nhớ trong và hệ thống tập tin (hệ thống File).

Mỗi người dùng hướng đến tài nguyên chung qua trạm cuối (terminal) của mình (các trạm cuối này được đặt tên và được hệ thống quản lý). Trong trường hợp đơn giản, trạm cuối chỉ bao gồm hai thiết bị là màn hình (để hiện thông tin cho người dùng) và bàn phím (để người dùng đưa yêu cầu đối với hệ điều hành). Trong nhiều trường hợp khác, có thể sử dụng một máy tính cá nhân đóng vai trò của một trạm cuối và như vậy mỗi người dùng vừa được phép sử dụng tài nguyên riêng vừa được phép sử dụng tài nguyên chung.

Điển hình nhất trong công việc phân chia tài nguyên của hệ thống máy tính trong hệ điều hành đa người dùng là việc phân chia CPU theo một chu kỳ thời gian mà mỗi người dùng được sử dụng CPU trong một khoảng thời gian nhất định (được gọi là lượng tử thời gian) và sau khi mỗi người đã được phân chia CPU thì lại chuyển đến lượt phân chia tiếp theo. Như vậy, phân chia thời gian (Time shared system) là cách thức của hệ đa người dùng khi điều phối CPU.

Là một hệ điều hành đa người dùng, UNIX đã khá phổ biến trong các lĩnh vực hoạt động CNTT, có thể được sử dụng từ máy vi tính cho tới máy tính mainframe. Nó đặc biệt thích hợp đối với các hệ Client-Server và mạng máy tính diện rộng.

1.1.2. Xuất xứ, sư phát triển và môt số đăc trưng của hê 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 tập tin 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

Trong thời gian gần đây (khoảng năm 1997) một số phiên bản mới của UNIX được giới thiệu và phổ biến trên Internet. Có thể tìm thấy các nội dung liên quan 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: một tổ chức chủ trương phát hành một dòng của UNIX mã nguồn mở, miễn phí: đó là hệ điều hành Linux.

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 tiến trình, trong đó mỗi người dùng có thể thực hiện các tiến 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.

1.1.3. Giới thiệu sơ bộ về Linux

Linus Tovalds (một sinh viên Phần lan) đã đưa ra phiên bản đầu tiên của hệ điều hành Linux vào tháng 8 năm 1991. Linus Tovalds đã xây dựng hệ điều hành Linux dựa trên một phiên bản nhỏ của UNIX có tên Minix (Minix do một chuyên gia hàng đầu về hệ điều hành là Giáo sư Andrew S. Tanenbaum xây dựng) theo hướng hoạt động trên máy tính cá nhân. Từ thời điểm đó, theo tư tưởng GNU, hàng nghìn chuyên gia trên toàn thế giới đã tham gia vào quá trình phát triển Linux và vì vậy Linux ngày càng đáp ứng nhu cầu của người dùng. Có thể kể ra một số đặc điểm sau đây của hệ điều hành Linux hiện tại:

- Linux tương thích với nhiều hệ điều hành như DOS, MicroSoft Windows ...:
- Có thể cài 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: đa người dùng, đa chương trình và đa xử lý.
- Linux có giao diện đồ hoạ (GUI) qua X-Window. Linux hỗ trợ nhiều giao thức mạng. Linux còn hỗ trợ tính 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: PC, mini và việc cài đặt khá thuận lợi song chưa xuất hiện trên máy tính lớn (mainframe).
- Linux ngày càng được hỗ trợ bởi các phần mềm ứng dụng: soạn thảo, quản lý mạng, quản trị cơ sở dữ liệu, bảng tính v.v.
- 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 Free SoftWare Foundation nên Linux phát triển nhanh và là một 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 Unicode, ISO10646) cho nên việc bản địa hóa trên Linux là triệt để nhất trong các hệ điều hành.

Tuy nhiên cũng còn một số khó khăn làm cho Linux chưa thực sự trở thành một hệ điều hành phổ dụng, trong đó có thể kể đến một số khó khăn như sau:

- Khó khăn khi cài đặt Linux và 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 các hệ điều hành khác như MS Windows thì vẫn còn có khoảng cách.

Với sự hỗ trợ của nhiều công ty Tin học hàng đầu (IBM, SUN, HP ...) và sự tham gia phát triển của hàng vạn chuyên gia trên toàn thế giới, 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 và Red Flag Linux. 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 điều hành Linux được chia thành 4 thành phần như sau:

- Nhân (system kernel),
- Shell,
- Hệ thống tập tin (File system),
- Các tiện ích (utilities) hay là hệ thống lệnh của Linux.

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

1.2.1. Sơ bô 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 tiến 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 tiến trình hiện thời cùng tồn tại. Đối với Linux, số lượng tiến trình có thể lên tới con số hàng nghìn. Với số lượng tiến 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 tập tin. Linux có hai cách thức

làm việc với các tập tin: làm việc theo byte (kí tự) và làm việc theo khối. Một đặc điểm đáng chú ý là việc tập tin trong Linux có thể được nhiều người cùng truy nhập tới nên các lời gọi hệ thống làm việc với tập tin cần đảm bảo việc tập tin được truy nhập theo quyền và được chia xẻ cho người dùng.

1.2.2. Sơ bô 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 cho chúng ta 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 tương ứng 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 tiến trình mới (khởi tạo tiến trình) và sau đó, shell chờ đợi tiến 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 ở 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 cuốn sách 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 cuốn sách này là lệnh thường trực. Chúng bao gồm các lệnh được chứa sẵn trong shell và các lệnh thường trực khác.
- Tập tin 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.
- Tập tin 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 việc 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 (năm 2000) Linux có khoảng hơn một nghìn lệnh.

Tuy nhiên chỉ có khoảng vài chục lệnh là thông dụng nhất đối với người dùng. Cuốn sách 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 và tham số vị trí. Tham số vị trí thường là tên tập tin, 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 các mô tả lệnh, phổ biến xuất hiện các ***tùy chọn lệnh*** mà được viết tắt là ***tùy-chọn***. Các tùy chọn lệnh (hầu hết là các tham số khóa) cho phép điều chỉnh hoạt động của lệnh trong Linux, làm cho lệnh có tính phổ dụng cao. Các 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 tập tin:

```
# ls -l g*↵
```

trong lệnh này:

- **ls** là tên lệnh thực hiện việc đưa danh sách các tên tập tin/ 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 tập tin có tên gọi bắt đầu là chữ cái "g".

Trong cuốn sách này, chúng ta 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 cuốn sách này, tại những dòng văn bản diễn giải, chúng tôi 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 ***người dùng cao cấp***) 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
```

Chương về lập trình shell sẽ giới thiệu chi tiết hơn về các cách thức nói trên.

- 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 lệnh được thực hiện; ngược lại, trong trường hợp có sai sót khi gõ lệnh thì shell sẽ 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 cuốn sách này.

1.3.1. Các quy ước khi viết lệnh

Trong cuốn sách 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 tập tin, 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 tập tin là

```
# more <tập-tin>
```

Thì từ **more** là tên lệnh, còn <tập-tin> là tham số trong đó ***tập-tin*** 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 tập tin với tên đã chỉ trong lệnh.

Để xem nội dung tập tin 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 "***tập-tin***" đã được thay thế bằng từ "***temp***" là tên tập tin 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, mô tả 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 tập tin, 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 tập tin.

Ví dụ, mô tả lệnh **chown** như sau:

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

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

- 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 <tập-tin>
```

có nghĩa là thay <tập-tin> bằng tên tập tin 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 [<tập-tin>]
```

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

☞ ***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).

- 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.

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

***Việc 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 xẩy 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 $\rightarrow$ để di chuyển con trỏ sang bên phải một vị trí
- Nhấn phím $\leftarrow$ 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 tập tin 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 tập tin mà lại liên quan đến một nhóm các tập tin mà tên gọi của các tập tin 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 tập tin (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 tập tin. Các kí tự này mang ý nghĩa như sau khi viết vào tham số tên tập tin 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 tập tin trong một thư mục nào đó, người dùng gõ * thay thế tham số ***tập-tin*** thì xác định được các tên tập tin sau (chúng ta viết bốn tên tập tin 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 ***tập-tin*** thì xác định được các tên tập tin 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 ***tập-tin*** thì xác định các tên tập tin 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 tập tin (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 tập tin (tên thư mục) và nhấn phím <**TAB**> để hoàn thành dòng lệnh.

1.3.2. 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 (Man Page)

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 các 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" trong "trang 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 nói chung là không quá khó hiểu, song để hiểu hết được nó cũng đòi hỏi không ít thời gian. Tuy nhiên, khi 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 tập tin 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 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. LỆNH THAO TÁC VỚI HỆ THỐNG

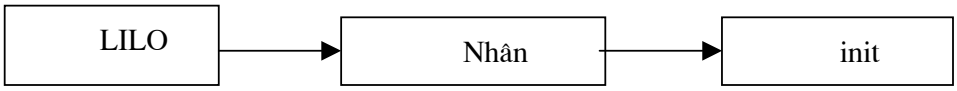
2.1. Quá trình khởi động Linux

Quá trình cài đặt Linux được trình bày trong phụ lục A của cuốn sách này. 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 tiến 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 tập tin 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 tập tin ***/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 nói trên được thực hiện xong, quá trình người dùng đăng nhập (login) vào hệ thống được tiến hành.

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 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.

---

*Chúng ta có thể thay đổi các dòng hiển thị như trình bày trên đây bằng cách sửa đổi tập tin **/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:

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

May1 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 2.8 sẽ cung cấp cách thức 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 thực hiện việc tắt hẳn máy.

Các tuỳ chọn của lệnh **halt**:

-w

không thực sự tắt máy nhưng vẫn ghi các thông tin lên tập tin **/var/log/wtmp** (đây là tập tin lưu trữ danh sách các người dùng đăng nhập thành công vào hệ thống).

-d

không ghi thông tin lên tập tin **/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 tập tin hay toàn bộ hệ thống tập tin 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 đông lai hê 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**.

2.3. 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** như sau:

```
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=tập-tin-văn-bản

giống như một tham số **--date** nhưng ứng với nhiều ngày cần xem: mỗi dòng của ***tập-tin-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= tập-tin

hiển thị thời gian sửa đổi ***tập-tin*** 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. 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] [tập-tin...]
```

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 tập-tin là tên tập tin chứa chương trình viết trên ngôn ngữ ***bc***, khi lệnh **bc** thực hiện sẽ tự động chạy các tập tin 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 cuốn sách 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).

2.6. 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.

2.7. 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 tuỳ 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.

2.8. Thay đổi nôi 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:

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 chúng ta 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.

CHƯƠNG 3. HỆ THỐNG TẬP TIN

3.1 Tổng quan về hệ thống tập tin

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: tập tin (File), thư mục, thư mục hiện thời ... Để cuốn sách mang 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 ***tập tin***. Tập tin 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 tập tin thuộc về chủ của nó là người dùng đã tạo ra tập tin. Tập tin 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 tập tin trên các thiết bị nhớ lâu dài (chẳng hạn đĩa từ) và đảm bảo các thao tác lên tập tin. Chính vì có hệ điều hành đảm bảo các chức năng liên quan đến tập tin nên người dùng không cần biết tập tin 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 tập tin.

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

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

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

❖ Tên tệp tin 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/tập tin trong Linux có thể có nhiều hơn một dấu chấm, ví dụ: **This_is.a.VERY_long.filename**. Nếu trong tên tập tin có dấu chấm "." thì xâu con của tên tập tin từ dấu chấm cuối cùng được gọi là phần mở rộng của tên tập tin (hoặc tập tin). Ví dụ, tên tập tin 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.

☞ ***Lưu ý:***

- Chúng ta nên lưu ý rằng, không phải ký tự nào cũng có nghĩa. Nếu có hai tập tin chỉ khác nhau ở ký tự cuối cùng, thì đối với Linux, đó là hai tập tin trùng tên. Bởi lẽ, Linux chỉ lấy 32 hay 64 ký tự đầu tiên trong tên tập tin mà thôi (tùy theo phiên bản Linux), phần tên tập tin còn lại dành cho chủ của tập tin, 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/tập tin, ví dụ hai tập tin **FILENAME.tar.gz** và **filename.tar.gz** là hai tập tin khác nhau.

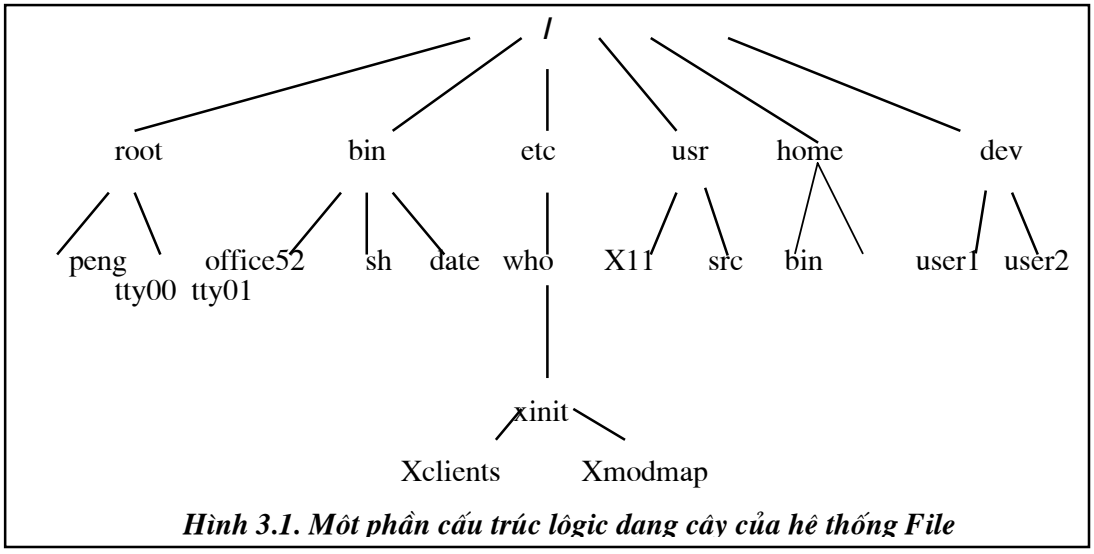
❖ Nếu trong tên thư mục/tập tin có chứa khoảng trống, sẽ phải đặt tên thư mục/tập tin vào trong cặp dấu nháy kép để sử dụng thư mục/tập tin đó. Ví dụ, để tạo thư mục có tên là "My document" chẳng hạn, hãy đánh dòng lệnh sau:

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

❖ Một số ký tự sau không được sử dụng trong tên thư mục/tập tin: !, *, $, &, # ...

❖ Khi sử dụng chương trình **mc** (Midnight Commander - chương 8), các tập tin khả thi trong Linux có dấu "*" được đặt trước tên tập tin, các tập tin sao lưu có dấu "~" và các tập tin có tên bắt đầu bởi dấu "**.**" là các tập tin ẩn, các tập tin có dấu "@" là các tập tin liên kết...

Tập hợp tất cả các tập tin có trong hệ điều hành được gọi là ***hệ thống tập tin*** là một hệ thống thống nhất. Bởi chính từ cách thức sử dụng thư mục, hệ thống tập tin được tổ chức lôgic theo dạng hình cây: Hệ thống tập tin 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 tập tin của nó. Hình 3.1. cho minh họa một phần trong cây lôgic của hệ thống tập tin.



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

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

/etc/X11/xinit/Xclients

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

Tên tập tin 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 tập tin mà hiện thời "người dùng đang ở đó".

Qua thư mục hiện thời, Linux cho phép người dùng chỉ một tập tin trong lệnh ngắn gọn hơn nhiều. Ví dụ, nếu thư mục hiện thời là thư mục xinit thì để chỉ tập tin 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 tập tin trên một thiết bị nhớ (đĩa mềm, vùng đĩa cứng chưa được đưa vào hệ thống tập tin) thành một thư mục con trong hệ thống tập tin của hệ thống bằng lệnh mount. Các hệ thống tập tin 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 tập tin Linux dành cho các bạn đọc muốn tìm hiểu thêm về hệ thống tập tin của một hệ điều hành UNIX điển hình.

3.1.2. Sơ bô kiến trúc nôi tai của hê thống tâp tin

Trên đĩa từ, hệ thống tập tin 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 tập tin. Trong hệ thống tập tin, 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 tập tin 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 tập tin.

***Siêu khối*:**

Siêu khối chứa nhiều thông tin liên quan đến trạng thái của hệ thống tập tin. 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 tập tin quản lý các inode.
- Kích thước của hệ thống tập tin.

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

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

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 tập tin.

- 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 tập tin nào đó.

Nhiều thao tác tạo tập tin mới, xoá tập tin, thay đổi nội dung tập tin v.v. cập nhật các thông tin này.

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

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

- 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 tập tin nào đó.

Hai tham số trên đây tạo thành cặp xác định được danh sách các inode rỗi trên hệ thống tập tin các thao tác tạo tập tin mới, xoá tập tin 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 tập tin 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 tập tin,

- Số lượng tổng cộng các khối rỗi trong hệ thống tập tin,

- Số lượng tổng cộng các inode rỗi trong hệ thống tập tin,

- Thông tin về thiết bị,

- Kích thước khối (đơn vị phân phối dữ liệu) của hệ thống tập tin. 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 tập tin.

Một trong khái niệm cốt lõi xuất hiện trong hệ thống tập tin 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 tập tin 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 tập tin dữ liệu và việc lưu trữ trên vật dẫn ngoài đối với Linux.

Nội dung của tập tin được chứa trong vùng dữ liệu của hệ thống tập tin và được phân chia các khối dữ liệu (nội dung tập tin) và hình ảnh phân bố nội dung tập tin 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 tập tin. Có thể quan niệm rằng, tập hợp bao gồm inode và tập các khối dữ liệu như vậy là một tập tin vật lý: inode có thông tin

về tập tin vật lý, trong đó có địa chỉ các khối dữ liệu chứa nội dung của tập tin 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 tập tin. 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 tập tin.

Như vậy, một tập tin chỉ có một inode song một tập tin lại có một hoặc một số tên tập tin. Người dùng tác động thông qua tên tập tin và tên tập tin lại tham chiếu đến inode (hai nội dung về tên tập tin 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 tập tin, mỗi tương ứng như vậy được gọi là một liên kết. Các inode được lưu trữ trên hệ thống tập tin 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 tập tin, hỗ trợ cho quá trình truy nhập dữ liệu trong hệ thống tập tin. 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 tập tin. Inode bao gồm các trường thông tin sau đây:

- Kiểu tập tin. Trong Linux phân loại các kiểu tập tin: tập tin thông thường (regular), thư mục, đặc tả kí tự, đặc tả khối và ống dẫn FIFO (pipes). Linux quy định trường kiểu tập tin có giá trị 0 tương ứng đó là inode chưa được sử dụng.
- Quyền truy nhập tập tin. Trong Linux, tập tin là một tài nguyên chung của hệ thống vì vậy quyền truy nhập tập tin 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 nhân của tập tin (đối tượng này được ký hiệu là ***u***: từ chữ user),
  - ❖ mức nhóm người dùng của chủ nhân của tập tin (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 nào đó tương ứng với việc cho phép tìm một tên tập tin nào đó có trong thư mục đó.

- Số lượng liên kết đối với inode: Đây chính là số lượng các tên tập tin 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 tập tin là một thành viên của nhóm này,
- Độ dài của tập tin tính theo byte,
- Thời gian truy nhập tập tin:
  - ❖ thời gian tập tin được sửa đổi muộn nhất,

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

• Bảng chứa các địa chỉ khối nhớ chứa nội dung tập tin. Bảng này chứa 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.

Nội dung của tập tin thay đổi khi có thao tác ghi lên nó; nội dung của một inode thay đổi khi nội dung của tập tin 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 tập tin,
- ❖ số lượng các tên tập tin nối với tập tin đang được mở,
- ❖ số hiệu thiết bị lôgic của hệ thống tập tin chứa tập tin 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 tập tin).

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

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

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

-f, --force

xóa bỏ các tập tin 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 tập tin bình thường được xem là đích liên kết từ một thư mục.

-i, interactive

vẫn tạo liên kết dù tập tin 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 tập tin trước khi tạo liên kết.

--help

hiển thị trang trợ giúp và thoát.

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

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

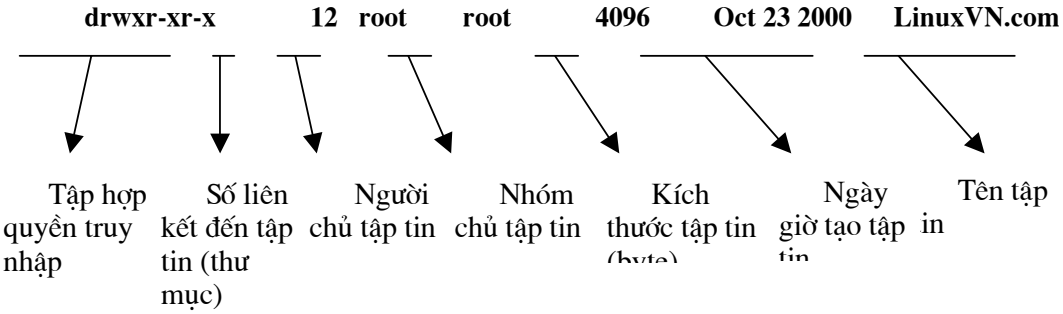
Khi chạy chương trình ***mc***, các tập tin liên kết có tên bắt đầu bởi dấu "@", và khi vệt sáng di chuyển đến tập tin liên kết thì tên tập tin được liên kết đến sẽ hiển thị ở bên dưới.

3.2 Quyền truy nhập thư mục và tập tin

3.2.1 Quyền truy nhập

Mỗi tập tin 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 tập tin và thư mục nhằm cung cấp truy nhập nhiều hơn hay ít hơn.

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



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

Theo mặc định, người dùng tạo một tập tin chính là người chủ (sở hữu) của tập tin đó và là người có quyền sở hữu nó. Người chủ của tập tin có đặc quyền thay đổi quyền truy nhập hay quyền sở hữu đối với tập tin 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.

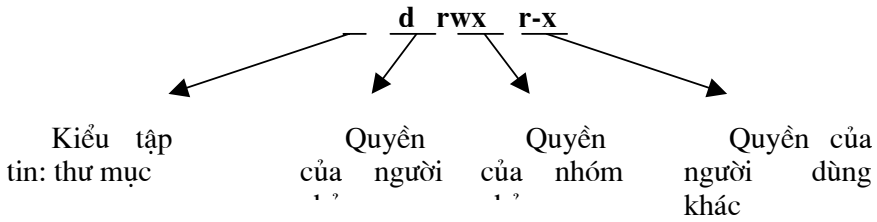
Tập hợp một chuỗi có 10 ký tự đã giới thiệu trên đây được chia ra làm 4 phần: kiểu tập tin, các quyền truy nhập đến tập tin của chủ sở hữu, của nhóm sở hữu và người dùng khác.

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

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

Chín ký tự tiếp theo trong chuỗi 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 tập tin**. 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 tập tin 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/tập tin, 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 tập tin 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 tập tin đó thành tập tin của họ và sửa chữa tập tin bản sao.

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

Loại quyền truy nhập thứ ba là quyền thực hiện, quyền này cho phép người dùng có thể chạy được tập tin, 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.

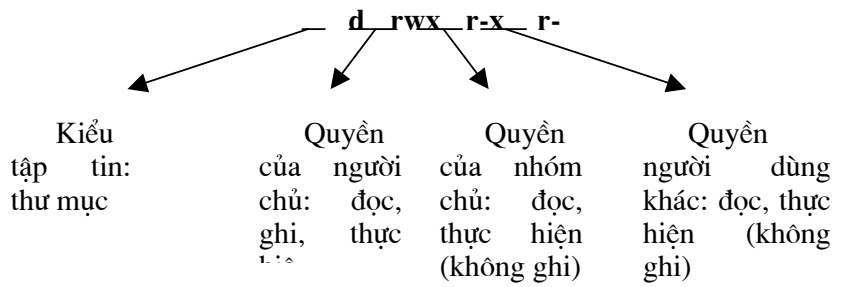
Bảng dưới đây giới thiệu cách ký hiệu của các quyền truy nhập:

Quyền truy nhập	Ý nghĩa
---	Không cho phép 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 tập tin và các thư mục

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 được xem nội dung của các tập tin 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 tập tin hay thư mục nào trong thư mục nhưng lại không thể xem cụ thể nội dung của một tập tin hay thư mục có trên thư mục đó.
- Hoặc giả sử có quyền thực hiện - quyền này sẽ cho phép thao tác với các tập tin 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 tập tin nào.

3.2.2. Các lệnh cơ bản

a. Thay đổi quyền sở hữu tập tin với lệnh chown

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

```
chown [tùy-chọn]... [chủ][.nhóm] <tập-tin ...>
```

Lệnh này cho phép thay chủ sở hữu **tập tin**. Nếu chỉ có tham số về ***chủ***, thì người dùng ***chủ*** sẽ có quyền sở hữu tập tin 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 tập tin. Nếu chỉ có dấu "**.**" và ***nhóm*** mà không có tên người chủ thì chỉ có quyền sở hữu nhóm của tập tin 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 tập tin 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 tập tin theo đệ quy.

-v, --verbose

hiển thị dòng thông báo với mọi tập tin 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 tập tin cho một người khác, người chủ cũ mất quyền sở hữu tập tin đó.

b. Thay đổi quyền sở hữu nhóm với lệnh chgrp

Các tập tin (và người dùng) còn thuộc vào các nhóm, đây là phương thức truy nhập tập tin 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 người dùng cao cấp 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 tập tin, hãy sử dụng lệnh **chgrp** với cú pháp như sau:

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

Lệnh này cho phép thay thuộc tính nhóm sở hữu của tập tin theo tên nhóm được chỉ ra trực tiếp theo tham số **nhóm** hoặc gián tiếp qua thuộc tính nhóm của tập tin 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 tập tin 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 tập tin theo đệ quy.

-v, --verbose

hiển thị dòng thông báo với mọi tập tin 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 tập tin theo nhóm chủ của một tập tin 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 tập tin với lệnh chmod

Cú pháp lệnh **chmod** có ba dạng:

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

Lệnh **chmod** cho phép xác lập quyền truy nhập theo kiểu (**mode**) trên **tập tin**. 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 tập tin **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

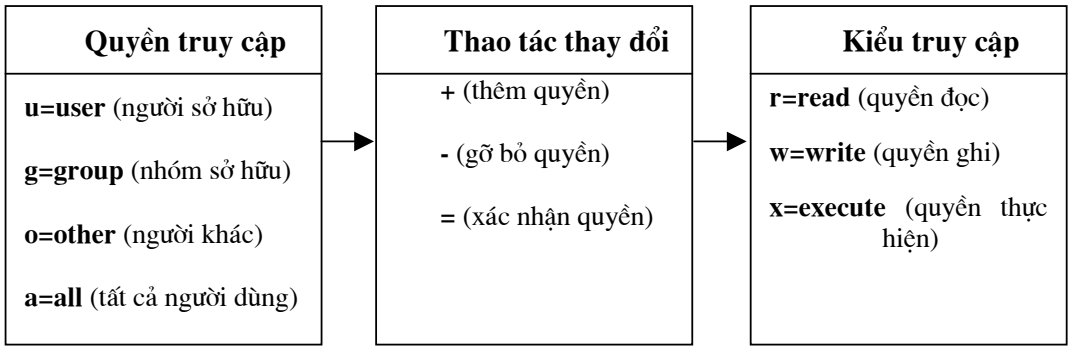
--help

- và tham số **--reference=RFILE** cũng ý nghĩa gián tiếp như trong lệnh **chgrp**.

Giải thích về hai cách xác lập quyền truy nhập tập tin 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 tập tin ***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 tập tin ***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 tập tin 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 tập tin 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 tập tin 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 tập tin thuộc quyền sở hữu của nhóm khởi động đó. Nếu muốn sử

dụng quyền sở hữu theo các nhóm khác đối với những tập tin 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 trên đây (trong mục 3.1.1.), Linux tổ chức hệ thống tập tin 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

Như đã được giới thiệu 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 tập tin tạm thời, nhân Linux và ảnh khởi động, các tập tin nhị phân quan trọng (những tập tin được sử dụng đến trước khi Linux có thể gắn kết đến phân vùng **/user**), các tập tin 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 tập tin 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 tập tin nhị phân. Như vậy, chương trình ứng dụng trong Linux là một tập tin 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 tập tin nhị phân khả thi. Nhưng theo thời gian, ngày càng có nhiều hơn các tập tin khả thi có

trong Linux, do đó, có thêm các thư mục như **/sbin**, **/usr/bin** được sử dụng để lưu trữ các tập tin 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 tập tin 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* /lost+found

Một tập tin 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* /mnt

Thư mục **/mnt** là nơi để kết nối các thiết bị (ví dụ đĩa cứng, đĩa mềm...) vào hệ thống tập tin chính nhờ lệnh **mount**. Thông thường các thư mục con của **/mnt** chính là gốc của các hệ thống tập tin 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* /tmp

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

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

* *Thư mục* /usr

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 tập tin 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* /home

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* /var

Thư mục **/var** được sử dụng để lưu trữ các tập tin 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 tiến trình, v.v..

* *Thư mục* /boot

Là thư mục chứa nhân của hệ thống (**Linux-*.*.**), **System.map** (tập tin ánh xạ đến các ***driver*** để nạp các hệ thống tập tin khác), ảnh (image) của hệ thống tập tin 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 tập tin khác.

* *Thư mục* /proc

Đây là thư mục dành cho nhân (**kernel**) của hệ điều hành và thực tế đây là một hệ thống tập tin 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 tập tin 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 như sau:

```
ls [tùy-chọn]... [tập-tin]...
```

Lệnh này sẽ đưa ra danh sách các tập tin liên quan đến tham số ***tập-tin*** trong lệnh. Trường hợp phổ biến tham số tập-tin là một thư mục, tuy nhiên trong một số trường

hợp khác, tham số ***tập-tin*** xác định nhóm (khi sử dụng các mô tả nhóm *, ? và cặp [và])**;** nếu không có tham số ***tập-tin***, mặc định danh sách các tập tin có trong thư mục hiện thời sẽ được hiển thị.

Các tùy chọn của lệnh là:

-a

liệt kê tất cả các tập tin, bao gồm cả tập tin ẩn.

-l

đưa ra thông tin đầy đủ nhất về các tập tin và thư mục.

-s

chỉ ra kích thước của tập tin, tính theo khối (1 khối = 1204 byte).

-F

xác định kiểu tập tin (**/** = thư mục, ***** = chương trình khả thi).

-m

liệt kê các tập tin được ngăn cách nhau bởi dấu "**,**".

-C

đưa ra danh sách các tập tin và thư mục theo dạng cột (hai thư mục gần nhau được xếp vào một cột).

-1

hiển thị mỗi tập tin hoặc thư mục trên một dòng.

-t

sắp xếp các tập tin và thư mục trong danh sách theo thứ tự về thời gian được sửa đổi gần đây nhất.

-x

đưa ra danh sách các tập tin 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
```

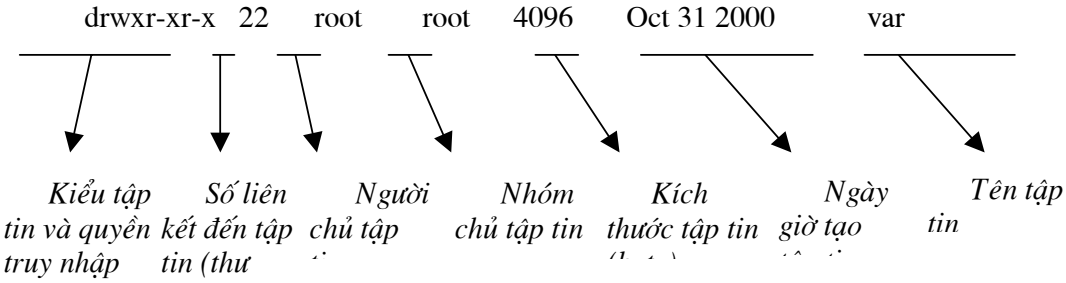
sẽ hiển thị danh sách đầy đủ nhất về các tập tin và thư mục có trong thư mục hiện thời.

```
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 tập tin trong danh sách (14*4+36+16=108).

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



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 tập tin 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 tao thư mục* mkdir

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

```
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 tập tin (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 như sau:

-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 tập tin 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 tập tin 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 tập tin

3.4.1 Các kiểu tập tin có trong Linux

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

- ***Tập tin người dùng (user data file):*** là các tập tin tạo ra do hoạt động của người dùng khi kích hoạt các chương trình ứng dụng tương ứng. Ví dụ như các tập tin thuần văn bản, các tập tin cơ sở dữ liệu hay các tập tin bảng tính.
- ***Tập tin hệ thống (system data file):*** là các tập tin 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.
- ***Tập tin thực hiện (executable file):*** là các tập tin chứa mã lệnh hay chỉ thị cho máy tính thực hiện. Tập tin 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 tập tin đó. Khi dùng trình ứng dụng **mc** (Midnight Commander, chương 8), tập tin thực hiện được bắt đầu bởi dấu (*) và thường có màu xanh lục.
- ***Thư mục hay còn gọi là tập tin bao hàm (directory):*** là tập tin bao hàm các tập tin khác và có cấu tạo hoàn toàn tương tự như tập tin thông thường khác nên có thể gọi là tập tin. Trong **mc**, tập tin bao hàm thường có màu trắng và bắt đầu bằng dấu ngã (**~**) hoặc dấu chia (**/**). Ví dụ: **/**, **/home**, **/bin**, **/usr**, **/usr/man**, **/dev** ...
- ***Tập tin thiết bị (device file):*** là tập tin mô tả thiết bị, dùng như là định danh để chỉ ra thiết bị cần thao tác. Theo quy ước, tập tin thiết bị được lưu trữ trong thư mục **/dev**. Các tập tin 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**, tập tin thiết bị có màu tím và bắt đầu bằng dấu cộng (+).

- ***9Tập tin liên kết (linked file):*** là những tập tin chứa tham chiếu đến các tập tin khác trong hệ thống tệp tin của Linux. Tham chiếu này cho phép người dùng tìm nhanh tới tập tin 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 tập tin này có tính năng trội hơn so với tính năng nguyên thủy của nó. Ta thấy loại tập tin 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 lênh tao tâp tin

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

** Tao tâp tin với lênh* touch

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

Cú pháp lệnh:

```
touch <tập-tin>
```

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

Ví dụ, dùng lệnh **touch** để tạo tập tin **newfile:**

```
# touch newfile
```

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

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

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

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

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

Ví dụ, lệnh

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

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

** Tao tập tin 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 tập tin hoặc các nguồn khác (màn hình ...), hay để xem nội dung của một tập tin ... Phần này trình bày tác dụng của lệnh **cat** đối với việc tạo tập tin.

Cú pháp lệnh:

```
cat > <tập-tin>
```

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 tập tin bằng lệnh cat tức là đã đổi hướng đầu ra của lệnh từ màn hình vào một tập tin. Người dùng gõ nội dung của tập tin ngay tại dấu nhắc màn hình và gõ **CTRL+d** để kết thúc việc soạn thảo. Nhược điểm của cách tạo tập tin 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 tập tin **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 tập tin vừa soạn thảo:

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

3.4.3 Các lênh thao tác trên tập tin

** Sao chép tập tin với lệnh* cp

Lệnh **cp** có hai dạng như sau:

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

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

Các tùy chọn của lệnh:

-a, --archive

giống như **-dpR** (tổ hợp ba tham số **-d**, **-p**, **-R**, như dưới đây).

-b, --backup[=CONTROL]

tạo tập tin lưu cho mỗi tập tin đích nếu như nó đang tồn tại.

-d, --no-dereference

duy trì các liên kết.

-f, --force

ghi đè tập tin 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 tập-tin-đích từ tập-tin-nguồn mà không sao chép.

-p, --preserve

duy trì các thuộc tính của tập-tin-nguồn sang tập-tin-đích.

-r

cho phép sao chép một cách đệ quy tập tin 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 tập tin.

-S, --suffix=<hậu-tố>

bỏ qua các hậu tố thông thường (hoặc được chỉ ra).

-u, --update

chỉ sao chép khi tập tin nguồn mới hơn tập tin đích hoặc khi tập tin đích chưa có.

-v, --verbose

đưa ra thông báo về quá trình sao chép.

--help

hiển thị trang trợ giúp và thoát.

Tập tin đích được tạo ra có cùng kích thước và các quyền truy nhập như tập tin nguồn, tuy nhiên tập tin 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 tập tin đích thì nội dung tập tin nguồn sẽ được sao chép sang tập tin đích. Trong trường hợp chỉ đưa ra vị trí tập tin đích được đặt trong thư mục nào thì tên của tập tin nguồn sẽ là tên của tập tin đích.

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

Trong ví dụ này, tên tập tin đích sẽ là **vd** nghĩa là tạo một tập tin 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 tập tin 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 tập tin, khi gõ lệnh có thể sử dụng kí hiệu mô tả nhóm để xác định một nhóm tập tin làm cho tăng hiệu lực của các lệnh đó. Ví dụ, lệnh:

  ```
  # cp * bak
  ```

  thực hiện việc sao chép mọi tập tin 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 tập tin 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 tập tin 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 cuốn sách. 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 tập tin với lệnh*** mv

Cú pháp lệnh đổi tên tập tin:

```
mv <tên-cũ> <tên-mới>
```

Lệnh này cho phép đổi tên tập tin từ **tên cũ** thành **tên mới**.

Ví dụ:

```
# mv vd newfile
```

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

** Xóa tập tin 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 như sau:

```
rm [tùy-chọn]... <tập-tin> ...
```

Lệnh **rm** cho phép xóa bỏ một tập tin hoặc nhiều tập tin.

Các tùy chọn của lệnh:

-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 tập tin (xác định qua tham số **tập-tin**) không tồn tại mà không cần nhắc nhở.

-i, --interactive

nhắc nhở trước khi xóa bỏ một tập tin.

-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 tập tin.

--help

hiển thị trang trợ giúp và thoát.

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

Ví dụ, dùng lệnh **ls** để xem danh sách các tập tin 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-           libnss_files.so.2
2.1.3.so                     libnss_hesiod-1.3.so
  libBrokenLocale.so.1       telex.o
  libNoVersion-2.1.3.so
  vd2.txt
```

Sử dụng lệnh xóa tập tin **vd2.txt** sau đây:

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

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

```
# 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-           libnss_files.so.2
2.1.3.so                     libnss_hesiod-1.3.so
  libBrokenLocale.so.1       telex.o
  libNoVersion-2.1.3.so
```

Dùng lệnh

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

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

** Lệnh đếm từ và dòng trong tập tin* wc

Linux có lệnh **wc** dùng để đếm số ký tự, số từ, hay số dòng trong một tập tin.

Cú pháp lệnh như sau:

```
wc [tùy-chọn]... [tập-tin]...
```

Lệnh hiện ra số lượng dòng, số lượng từ, số lượng ký tự có trong mỗi tập tin, và một dòng tính tổng nếu có nhiều hơn một tập tin 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 tập tin trong lệnh thì sẽ đọc và đếm trên thiết bị vào chuẩn.

Các tuỳ chọn:

-c, --byte, --chars

đưa ra số ký tự trong tập tin.

-l, --lines

đưa ra số dòng trong tập tin.

-L, --max-line-length

đưa ra chiều dài của dòng dài nhất trong tập tin.

-w, --words

đưa ra số từ trong tập tin.

--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 tập tin **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 tập tin 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 tập tin có trong một thư mục:

```
# ls | wc -l
37
```

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

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

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

* *Lệnh loại bỏ những dòng không quan trong* uniq

Trong một số trường hợp khi xem nội dung một tập tin, 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 tập tin, có thể sử dụng lệnh **uniq** để liệt kê ra nội dung tập tin 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 của lệnh:

-c, --count

đếm và hiển thị số lần xuất hiện của các dòng trong tập tin.

-d

hiển thị lên màn hình dòng bị trùng lặp.

-u

hiển thị nội dung tập tin sau khi xóa bỏ toàn bộ các dòng bị trùng lặp không giữ lại một dòng nào.

-i

hiển thị nội dung tập tin 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 tập tin không có các dòng trùng lặp thì lệnh không có tác dụng.

Ví dụ, người dùng sử dụng lệnh **cat** để xem nội dung tập tin **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 tập tin **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 tập tin **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 nôi dung tập tin với lênh* 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] ... [tập-tin] ...
```

Hiển thị nội dung sau khi sắp xếp của một hoặc nhiều tập tin 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 tập tin (từng chữ cái theo bảng chữ hệ thống (chẳng hạn ASCII) và kể từ vị trí đầu tiên trong các dòng).

Các tùy chon:

+<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 tập tin 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 tập tin.

-r

chuyển đổi thứ tự sắp xếp hiện thời.

Ví dụ, muốn sắp xếp tập tin **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 sẽ cho thứ tự sắp xếp của các tập tin theo kích thước trong thư mục hiện thời.

3.4.4 Các lệnh thao tác theo nội dung tập tin

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

Cú pháp lệnh **file**:

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

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

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

Một số tuỳ chọn sau đây:

-b

cho phép chỉ đưa ra kiểu tập tin mà không đưa kèm theo tên tập tin.

-f `tên-tập-tin`

cho phép hiển thị kiểu của các tập tin có tên trùng với nội dung trên mỗi dòng trong tập tin **`tên-tập-tin`**. Để kiểm tra trên thiết bị vào chuẩn, sử dụng dấu "**-**".

-z

xem kiểu của tập tin 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 tập tin **file.c**, **file** và thư mục **/dev/hda**.

Cần nhớ rằng kết quả của lệnh **file** không phải lúc nào cũng chính xác.

* *Xem nôi dung tâp tin với lênh* cat

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

Cú pháp lệnh:

```
cat [tùy-chọn] <tên tập tin>
```

Các tùy chọn như sau:

-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 tập tin 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 tập tin 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 tập tin lớn với lệnh* more

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

Cú pháp lệnh:

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

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

Các lựa chọn như sau:

-số

xác định số dòng nội dung của tập tin 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 tập tin 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 tập tin.

-c

không cuộn màn hình, thay vào đó xóa màn hình và hiển thị nội dung tập tin bắt đầu từ đỉnh màn hình.

-s

xóa bớt các dòng trống liền nhau trong nội dung tập tin 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 tập tin.

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

* *Thêm số thứ tư của các dòng trong tập tin với lệnh* nl

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

Cú pháp lệnh:

```
nl [tùy-chọn] ... <tập-tin>
```

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

Các tuỳ chọn lệnh:

-b, --body-numbering=STYLE

sử dụng kiểu STYLE cho việc đánh thứ tự các dòng trong nội dung tập tin. 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 tập tin (một câu có thể có hai dòng ...).

-h, --header-numbering=STYLE

sử dụng kiểu STYLE để đánh số các dòng trong nội dung tập tin.

-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 tập tin **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 tập tin được hiện ra thành hai dòng trên giấy).

* *Xem qua nội dung tập tin với lệnh head*

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

Cú pháp lệnh

```
head [tùy-chọn]... [tập-tin]...
```

Lệnh này mặc định sẽ đưa ra màn hình 10 dòng đầu tiên của mỗi tập tin. Nếu có nhiều hơn một tập tin, thì lần lượt tên của tập tin và 10 dòng nội dung đầu tiên sẽ được hiển thị. Nếu không có tham số **tập-tin**, hoặc **tập-tin** 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 tập tin (**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 tập tin ở dòng đầu.

-v, --verbose

luôn đưa ra tên tập tin 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 ~/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 tập tin **vdhead1** và **vdhead2**.

* *Xem qua nôi dung tâp tin với lênh* tail

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

```
tail [tùy-chọn]... [tập-tin]...
```

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

Các tùy chọn như sau:

--retry

cố gắng mở một tập tin 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 tập tin sẽ hiện thông tin về tập tin: **-f**, **--follow**, và **--follow=descriptor** là như nhau.

-n, --lines=n

hiển thị **n** (số) dòng cuối cùng của tập tin thay cho 10 dòng ngầm định.

--max-unchanged-stats=n

hiển thị tài liệu về tập tin (ngầm định **n** là 5).

--max-consecutive-size-changes=n

hiển thị tài liệu về tập tin (ngầm định **n** là 200).

--pid=PID

kết hợp với tùy chọn **-f**, chấm dứt sau khi tiến trình có chỉ số = **PID** lỗi.

-q, --quiet, --silent

không đưa ra tên tập tin 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 tập tin.

--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 tập tin **vdtail1** và **vdtail2**.

** Tìm sự khác nhau giữa hai tập tin (lệnh* diff*)*

Việc tìm ra sự khác nhau giữa hai tập tin 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] <tập-tin1> <tập-tin2>
```

Trong trường hợp đơn giản, lệnh **diff** sẽ so sánh nội dung của hai tập tin. Nếu **tập-tin1** là một thư mục còn **tập-tin2** là một tập tin bình thường, **diff** sẽ so sánh tập tin có tên trùng với **tập-tin2** trong thư mục **tập-tin1** với **tập-tin2**.

Nếu cả **tập-tin1** và **tập-tin2** đều là thư mục, **diff** sẽ thực hiện sự so sánh lần lượt các tập tin 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 tập tin.

Các tuỳ chọn:

-a

xem tất cả các tập tin 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=tập-tin

khi so sánh thư mục, bỏ qua các tập tin và các thư mục con có tên phù hợp với mẫu có trong **tập-tin.**

-i

so sánh không biệt chữ hoa chữ thường.

-r

thực hiện so sánh đệ qui trên thư mục.

-s

thông báo khi hai tập tin là giống nhau.

-y

hiển thị hai tập tin cạnh nhau để dễ phân biệt sự khác nhau.

3.4.5 Các lệnh tìm tập tin

** Tìm theo nội dung tập tin 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 tập tin được chỉ ra.

Cú pháp lệnh **grep**:

```
grep [tùy-chọn] ... <mẫu-lọc> [tập-tin]
```

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

Các tùy chọn là:

-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 tập tin đầu vào trước mỗi dòng được đưa ra

-c, --count

đếm số dòng tương ứng chứa mẫu trong tập tin 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 tập tin 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 tập tin bên trong thư mục (đệ quy); tùy chọn này tương đương với tùy chọn **-r**.

-f tập-tin, --file=tập-tin

lấy các mẫu từ **tập-tin**, một mẫu trên một dòng. Tập tin trống chứa đựng các mẫu rỗng, và các dòng đưa ra cũng là các dòng trống.

-H, --with-tập-tin

đưa ra tên tập tin trên mỗi dòng chứa mẫu tương ứng.

-h, --no-filename

không hiển thị tên tập tin kèm theo dòng chứa mẫu trong trường hợp tìm nhiều tập tin.

-i

hiển thị các dòng chứa mẫu không phân biệt chữ hoa chữ thường.

-l

đưa ra tên các tập tin trùng với mẫu lọc.

-n, --line-number

thêm số thứ tự của dòng chứa mẫu trong tập tin.

-r, --recursive

đọc tất cả các tập tin có trong thư mục (đệ quy).

-s, --no-messages

bỏ qua các thông báo lỗi tập tin 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 tập tin **text**:

```
# cat -n text
```

thì hiện ra nội dung tập tin đó như sau:

```
1    $ file file.c file /dev/hda
2    file.c: C program text
3    file:ELF 32-bit LSB executable, Intel 80386, version 1,
4    dynamically linked, not stripped
5    /dev/hda: block special
6
7    $ file -s /dev/hda{,1,2,3,4,5,6,7,8,9,10}
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 tập tin khác nhau. Bảng dưới đây sẽ liệt kê một số 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

```
# 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 thứ 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 tập tin **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

Người dùng gõ lệnh:

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

thì nhận được thông báo kết quả:

aglw:

: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 tập tin 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: tập tin **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 tập tin **mauloc** có nội dung là:

```
# cat mauloc
```

empty

test

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

```
# fgrep -i -f mauloc thu
```

/dev/hda9: empty

/dev/hda10: empty

thutest

toithutest

* *Tìm theo các đặc tính của tập tin với lệnh* find

Các đoạn trên đây đã giới thiệu cách thức tìm tập tin 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 tập tin theo các thuộc tính của tập tin. 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 tập tin 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 tập tin 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 tập tin 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 tập tin được truy nhập n phút trước.

-atime n

tìm tập tin được truy nhập n*24 giờ trước.

-cmin n

trạng thái của tập tin được thay đổi n phút trước đây.

-ctime n

trạng thái của tập tin được thay đổi n*24 giờ trước đây.

-empty

tập tin rỗng và hoặc là thư mục hoặc là tập tin bình thường.

-fstype `kiểu`

tập tin thuộc hệ thống tập tin với **`kiểu`.**

-gid n

chỉ số nhóm của tập tin là n.

-group `nhóm`

tập tin thuộc quyền sở hữu của **`nhóm`.**

-links n

tập tin có n liên kết.

-mmin n

dữ liệu của tập tin được sửa lần cuối vào n phút trước đây.

-mtime n

dữ liệu của tập tin được sửa vào n*24 giờ trước đây.

-name `mẫu`

tìm kiếm tập tin có tên là **`mẫu`**. Trong tên tập tin có thể chứa cả các ký tự đại diện như dấu "*", "?"...

-type kiểu

tìm các tập tin thuộc **kiểu** với kiểu nhận các giá trị:

- ❖ b: đặc tả theo khối
- ❖ c: đặc tả theo ký tự

- d: thư mục
- p: pipe
- f: tập tin bình thường
- l: liên kết tượng trưng
- s: socket

-uid n

chỉ số người sở hữu tập tin là n.

-user tên-người

tập tin 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 tập tin cần tìm. Tùy chọn **exec** phải sử dụng dấu {} - nó sẽ thay thế cho tên tập tin tương ứng, và dấu '\' tại cuối dòng lệnh, (phải có khoảng trống giữa {} và '\'). Kết thúc lệnh là dấu '**;**'

-fprint tập-tin

hiển thị đầy đủ tên tập tin vào trong **tập-tin**. Nếu **tập-tin** 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 tập tin trên thiết bị ra chuẩn.

-ls

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

Ví dụ:

```
# find -name 'what*'
```

./usr/bin/whatis

./usr/bin/whatnow

./usr/doc/AfterStep-1.8.0/TODO/1.0_to_1.5/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 tập tin

3.5.1 Sao lưu các tập tin (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 sau 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 chúng tôi muốn giới thiệu trong cuốn sách 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 tập tin, bao gồm tất cả các tập tin. Sao lưu tăng dần chỉ sao lưu những tập tin 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] [<tập-tin>, ...] [<thư-mục>, ...]
```

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

Lệnh **tar** có các lựa chọn sau đây:

-c, --create

tạo tập tin lưu trữ mới.

-d, --diff, --compare

tìm ra sự khác nhau giữa tập tin lưu trữ và tập tin hệ thống được lưu trữ.

--delete

xóa từ tập tin lưu trữ (không sử dụng cho băng từ).

-r, --append

chèn thêm tập tin vào cuối tập tin lưu trữ.

-t, --list

liệt kê nội dung của một tập tin lưu trữ.

-u, --update

chỉ thêm vào tập tin lưu trữ các tập tin mới hơn các tập tin đã có.

-x, --extract, --get

tách các tập tin ra khỏi tập tin 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 tập tin lưu trữ.

-f, --file [HOSTNAME:]`tập-tin`

tùy chọn này xác định tên tập tin lưu trữ hoặc thiết bị lưu trữ là **tập-tin** (nếu không có tùy chọn này, mặc định nơi lưu trữ là **/dev/rmt0**).

-h, --dereference

không hiện các tập tin liên kết mà hiện các tập tin mà chúng trỏ tới.

-k, --keep-old-files

giữ nguyên các tập tin lưu trữ đang tồn tại mà không ghi đè tập tin lưu trữ mới lên chúng.

-K, --starting-file `tập-tin`

bắt đầu tại **`tập-tin`** trong tập tin lưu trữ.

-l, --one-file-system

tạo tập tin lưu trữ trên hệ thống tập tin cục bộ.

-M, --multi-volume

tùy chọn này được sử dụng khi dung lượng của tập tin 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 tập tin mới hơn các tập tin được lưu trữ trong ngày DATE.

--remove-files

xóa tập tin gốc sau khi đã sao lưu chúng vào trong tập tin lưu trữ.

--totals

đưa ra tổng số byte được tạo bởi tùy chọn **--create**.

-v, --verbose

hiển thị danh sách các tập tin 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 sẽ tạo một tập tin 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 tập tin 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 tập tin bị thay đổi, ví dụ như thay đổi quyền truy nhập của tập tin, hay thay đổi tên tập tin 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 tập tin hệ thống với danh sách các tập tin được sao lưu từ trước.

3.5.2 Nén dữ liêu

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, nhưng trong cuốn sách này chúng tôi giới thiệu hai phương cách phổ biến là **gzip** và **compress**.

* *Nén, giải nén và xem nôi dung các tâp tin với lênh* gzip, gunzip *và* zcat

Cú pháp các lệnh này như sau:

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

Lệnh **gzip** sẽ làm giảm kích thước của tập tin và khi sử dụng lệnh này, tập tin gốc sẽ bị thay thế bởi tập tin nén với phần mở rộng là .gz, các thông tin khác liên quan đến tập tin không thay đổi. Nếu không có tên tập tin 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 tập tin nén quá dài so với tên tập tin gốc, **gzip** sẽ cắt bỏ bớt. **gzip** sẽ chỉ cắt phần tên tập tin vượt quá 3 ký tự (các phần được ngăn cách với nhau bởi dấu chấm). Nếu tên tập tin gồm nhiều phần nhỏ thì phần dài nhất sẽ bị cắt bỏ. Ví dụ, tên tập tin là **gzip.msdos.exe**, khi được nén sẽ có tên là **gzip.msd.exe.gz**.

Tập tin 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 tập tin có phần mở rộng là **.gz**, **-gz**, **.z**, **-z**, **_z** hoặc **.Z** ... **gunzip** dùng để giải nén các tập tin nén bằng lệnh **gzip**, **zip**, **compress**, **compress -H**.

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

Các tùy chọn như sau:

-c, --stdout --to-stdout

đưa ra trên thiết bị ra chuẩn; giữ nguyên tập tin gốc không có sự thay đổi. Nếu có nhiều hơn một tập tin đầu vào, đầu ra sẽ tuần tự là các tập tin được nén một cách độc lập.

-d, --decompress --uncompress

giải nén.

-f, --force

thực hiện nén hoặc giải nén thậm chí tập tin có nhiều liên kết hoặc tập tin 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 tập tin được nén:

- **compressed size:** kích thước của tập tin nén
- **uncompressed size**: kích thước của tập tin được giải nén
- **ratio**: tỷ lệ nén (0.0% nếu không biết)
- **uncompressed_name**: tên của tập tin 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 tập tin được giải nén

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

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

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

-n, --no-name

khi nén, tùy chọn này sẽ không lưu trữ tên tập tin gốc và thời gian nén, (tên tập tin 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 tập tin 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 tập tin hay khi thời gian nén bị mất sau khi chuyển đổi tập tin.

-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 tập tin được chuyển đến hệ thống khác.

-t, --test

tùy chọn này được sử dụng để kiểm tra tính toàn vẹn của tập tin được nén

-v, --verbose

hiển thị phần trăm thu gọn đối với mỗi tập tin 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 tập tin với các lệnh* compress, uncompress, zcat

Cú pháp các lệnh như sau:

```
compress [tùy-chọn] [<tập-tin>]
uncompress [tùy-chọn] [<tập-tin>]
zcat [tùy-chọn] [<tập-tin>]
```

Lệnh **compress** sẽ làm giảm kích thước của tập tin và khi sử dụng lệnh này, tập tin gốc sẽ bị thay thế bởi tập tin nén với phần mở rộng là **.Z**, các thông tin khác liên quan đến tập tin không thay đổi. Nếu không có tên tập tin nào được chỉ ra, thông tin từ thiết bị vào chuẩn sẽ được nén và gửi ra thiết bị ra chuẩn. Lệnh **compress** chỉ sử dụng cho các tập tin thông thường. Trong một vài trường hợp, nó sẽ bỏ qua liên kết tượng trưng. Nếu một tập tin có nhiều liên kết cứng, **compress** sẽ bỏ qua việc nén tập tin đó trừ khi có tùy chọn **-f**.

Các tùy chọn là:

-f

nếu tùy chọn này không được đưa ra và **compress** chạy trong chế độ nền trước, người dùng sẽ được nhắc khi các tập tin đã thực sự tồn tại và có thể bị ghi đè. Các tập tin 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 tập tin nào bị thay đổi.

Lệnh **zcat** tương đương với **uncompress -c**. **zcat** thực hiện việc giải nén hoặc là các tập tin 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 tập tin được nén.

3.6 Sử dụng rpm

3.6.1.Giới thiệu chung về rpm

rpm là tên viết tắt của RedHat Package Manager, là một chương trình cho phép người dùng quản lý các phần mềm được cài đặt trên Linux một cách mềm dẻo và hữu dụng. Với **rpm**, người dùng có thể đơn giản hóa các thao tác bằng tay. Dùng **rpm** để cài đặt các gói phần mềm một cách tự động, có thể gỡ bỏ các phần mềm đã được cài một cách an toàn, có thể kiểm tra tính toàn vẹn của các chương trình. Khi các chương trình được đóng gói bằng **rpm** thì có thể mang từ nơi này qua nơi khác mà không sợ bị sai lạc. Đặc biệt RPM có sẵn khả năng bảo mật tự xác thực chính nó bằng chữ ký số. Với đặc tính này có thể tải một phần mềm nào đó trên Internet mà không sợ bị bẫy vì do một người trung gian nào đó đã thay đổi phần mềm này. Nếu một gói tự chứng thực được nơi xuất xứ thì chúng ta có thể yên tâm sử dụng gói đó.

Với **rpm**, người dùng có thể tự đóng gói phần mềm riêng cho mình. Việc đóng gói có thể theo hai dạng là đóng gói từ source, hay tái tạo lại gói nhị phân từ gói đã có. Mặt khác, **rpm** cũng cung cấp các thư viện hàm API (Application Programming Interface) giúp cho mọi người dễ dàng tạo gói **rpm** từ các chương trình C, python... khác nhau.

Một trong những phương thức (mode) sau đây của **rpm** bắt buộc phải được chọn để sử dụng: duy trì cơ sở dữ liệu gói, dựng gói, đặt yêu cầu, duy trì gói đã được khởi tạo, đánh dấu gói, trộn gói ... Hình thái sử dụng rpm là hết sức phong phú vì vậy, các đoạn dưới đây chỉ hạn chế giới thiệu những nội dung cơ bản nhất về **rpm**.

3.6.2 RPM với người dùng

** Cài đăt gói:*

Cú pháp đầy đủ là:

```
rpm {-i|--install} [tuỳ-chọn]... <tập-tin-rpm>...
```

Khi một gói được cài đặt, chương trình **rpm** thực hiện các công việc sau:

- Kiểm tra tính phụ thuộc của gói.
- Kiểm tra tình trạng xung đột giữa các tập tin.
- Thực hiện các script trước lúc cài đặt.
- Sửa lại các tập tin cấu hình đang có trong hệ thống.
- Bung nén các tập tin từ gói cài đặt vào các vị trí thích hợp.
- Thực hiện các script sau cài đặt.
- Lưu lại tất cả các thông tin trên vào cơ sở dữ liệu.

** Xóa môt gói ra khỏi hê thống*

Để xóa một gói ra khỏi hệ thống, dùng lệnh:

```
rpm {-e|--erase} [tùy-chọn] <danh-sách-các-gói>
```

Khi một gói được xóa khỏi hệ thống, **rpm** thực hiện các công việc sau:

- Kiểm tra xem có một gói nào trong hệ thống phụ thuộc vào gói sẽ bị xóa không.
- Thực hiện **script pre-uninstall** nếu có.
- Kiểm tra các tập tin cấu hình có bị thay đổi không, nếu có sẽ lưu lại một bản sao.
- Tra cứu cơ sở dữ liệu rpm để xóa các tập tin của gói đó.
- Thực hiện các script **post-uninstall** nếu có.
- Xóa các thông tin liên quan đến các gói trong cơ sở dữ liệu.

** Nâng cấp môt gói*

Cách thức nâng cấp một gói cũng tương tự như cài đặt gói. Chỉ khác là tham số không phải **-i** mà là **-U.** Một điều chú ý ở đây là nếu cần phải cài lại một gói cũ hơn gói đang có trên hệ thống thì cần thêm tham số **--oldpackage**.

** Lấy thông tin về các gói phần mềm (package)*

Một trong những điểm thú vị nhất về **rpm** là **rpm** luôn sẵn sàng cung cấp những thông tin cần thiết về toàn bộ các gói có trong hệ thống.

Cú pháp đầy đủ như sau:

```
rpm {-q |--query} [tùy-chọn-q]
```

Tùy-chọn-q có hai loại là tùy chọn bó và tùy chọn thông tin.

- Tùy chọn bó xác định các gói cần cung cấp thông tin và có các dạng như sau:

pkg1 pkg2 .. pkgN

là một danh sách tên các gói đã cài đặ).

-p <tập-tin-rpm>

Hỏi thông tin về bó (một **tập-tin-rpm**) có thể chưa được cài.

-f tập-tin

Hỏi thông tin về gói chứa **tập-tin**.

-a

Hỏi thông tin về tất cả các gói đã được cài đặt.

--whatprovides <x>

Hỏi thông tin về gói cung cấp <x>.

-g <nhóm>

Hỏi thông tin về gói thuộc nhóm **<nhóm>**.

--whatrequires <x>

Hỏi thông tin về gói cần đến <x>.

** Dùng RPM để kiểm tra các gói đã cài đăt*

Để kiểm tra một gói đã cài đặt trên hệ thống ta có thể dùng lệnh **rpm -V**.

Cú pháp đầy đủ là:

```
rpm {-V|--verify |-y} [tùy-chọn-v]
```

Như đã nói ở các phần trên, toàn bộ thông tin về các gói đều được lưu trữ trong cơ sở dữ liệu của **rpm**. Kể cả là nội dung của từng tập tin (**rpm** sử dụng thuật toán mã hóa hàm băm MD5 để kiểm tra tính toàn vẹn nội dung của một tập tin). Do đó, mọi sự thay đổi của các gói rpm đều được lưu lại. Lệnh **rpm -V** sẽ kiểm tra lại tất cả các thông tin đó xem có khớp với ban đầu hay không. Khi có một lỗi nào đó xảy ra, **rpm -V** sẽ cung cấp các thông tin về lỗi phát hiện được.

CHƯƠNG 4. LỆNH 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.

4.1 Tài khoản người dùng

Khi một máy tính được nhiều người dùng sử dụng, nó rất cần phải có được sự phân biệt giữa các người dùng khác nhau, ví dụ, mỗi người dùng lại có những tập tin 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 sẽ có một cái tên duy nhất 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 tập tin, các tài nguyên, và các thông tin thuộc về người dùng đó.

Khi cài đặt hệ điều hành Linux, đăng nhập chính sẽ được tự động tạo ra. Đăng nhập này, gọi là **root**, được xem là thuộc về siêu người dùng (người dùng cấp cao, siêu người dùng), 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.

Vậy nếu muốn tạo một người dùng mới thì sẽ làm như thế nào? Các phần sau sẽ 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.

4.2 Các lệnh cơ bản quản lý người dùng

Người dùng được quản lý thông qua tên người dùng hoặc chỉ số người dùng. Thông thường, nhân hệ thống quản lý người dùng theo chỉ số, vì việc quản lý theo chỉ số sẽ dễ dàng và nhanh hơn so với quản lý theo tên, và có một cơ sở dữ liệu để 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 người dùng cấp cao.

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**).

4.2.1 Tập tin /etc/passwd

Danh sách người dùng cũng như các thông tin tương ứng được lưu trữ trong tập tin **/etc/passwd**.

Ví dụ dưới đây là nội dung của tập tin **/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 tập tin 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 tập tin **/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 tập tin.

4.2.2 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 tập tin hệ thống, thư mục cá nhân sẽ được tạo, hay các tập tin 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 tập tin **/etc/passwd** bằng **vipw**. Lệnh **vipw** mở trình soạn thảo trên hệ thống và hiệu chỉnh bản sao tạm của tập tin **/etc/passwd**. Việc sử dụng tập tin tạm và khóa tập tin sẽ có tác dụng như một cơ chế khóa để ngăn việc hai người dùng cùng soạn thảo tập tin 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 tập tin này được lưu, **vipw** sẽ kiểm tra sự đồng nhất trên tập tin bị thay đổi. Nếu tất cả mọi thứ dường như thích hợp thì có nghĩa là tập tin **/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

Tao thư mục cá nhân của người dùng mới với lệnh mkdir

```
# mkdir /home/new
```

- Sao chép các tập tin từ thư mục **/etc/skel/** (đây là thư mục lưu trữ các tập tin cần thiết cho người dùng) vào tập tin cá nhân vừa tạo
- Thay đổi quyền sở hữu và các quyền truy nhập tập tin **/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 tập tin.

4.2.3 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 tập tin **/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 tập tin 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 cá nhân 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 tiến 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
```

4.2.4 Xóa bỏ môt 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 tập tin 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 tập tin tồn tại trong thư mục cá nhân của người dùng cũng như các tập tin nằm trong các thư mục khác có liên quan đến người dùng sẽ 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 tiến 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 tập tin **/etc/passwd**.

4.3 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, tập tin, hệ thống tập tin hoặc toàn bộ máy tính mà những người dùng khác không thuộc nhóm đó không thể truy cập được.

4.3.1 Nhóm người dùng và tập tin /etc/group

Thông tin về nhóm người dùng được lưu trong tập tin **/etc/group**, tập tin này có cách bố trí tương tự như tập tin **/etc/passwd**. Ví dụ nội dung của tập tin **/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 tập tin 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)

4.3.2 Thêm nhóm người dùng

Cho phép hiệu chỉnh thông tin trong tập tin **/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 tập tin **/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 tập tin **/etc/group**.

4.3.3 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.

4.3.4 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 tập tin **/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 tập tin tài khoản hệ thống, xóa tất cả các thực thể liên quan đến nhóm. Tên nhóm phải thực sự tồn tại.

4.4 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.

4.4.1 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 tập tin 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.

4.4.2 Xác đinh người dùng đang đăng nhập (lênh **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.

** Có một cách khác để xác định thông tin người dùng với lệnh* id

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)
#
```

*4.4.3 Xác đinh các tiến trình đang đươc tiến hành (lênh **w**)*

Lệnh **w** cho phép xác định được thông tin về các tiến trình đang được thực hiện trên hệ thống và những người dùng tiến hành tiến trình đó.

Cú pháp lệnh:

```
w [người-dùng]
```

Lệnh **w** đưa ra thông tin về người dùng hiện thời trên hệ thống và tiến trình họ đang thực hiện. Nếu chỉ ra người dùng trong lệnh thì chỉ hiện ra các tiến 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 5. CÁC LỆNH QUẢN LÝ THIẾT BỊ NGOẠI VI

5.1 Giới thiệu về cách thức Linux quản lý thiết bị ngoại vi

Trong một hệ thống máy tính, CPU là thiết bị trung tâm nhưng không phải là thiết bị điều khiển duy nhất, mỗi thiết bị vật lý đều có bộ điều khiển riêng. Bàn phím, chuột, các cổng tuần tự được điều khiển bởi một **SuperIO chip**. Những ổ đĩa IDE, SCSI được điều khiển bởi bộ điều khiển IDE, SCSI tương ứng, v..v. Mỗi bộ điều khiển phần cứng đều có những thanh ghi trạng thái (CSRs) riêng và chúng khác nhau cho các thiết bị khác nhau. Các CSR dùng để khởi động, dừng và khởi sinh thiết bị, ngoài việc được nhúng vào các ứng dụng chúng còn được lưu trữ trong nhân Linux.

Trong phần này, chúng ta sẽ xem xét việc Linux điều khiển các thiết bị phần cứng như thế nào. Linux cho phép chúng ta có quyền điều khiển phần cứng của hệ thống (Tương tự như Contronl Panel của Windows). Tuy nhiên, việc truy cập và điều khiển các thiết bị phần cứng không dễ như trong Windows, mặc dù nó tỏ ra khá cơ động và không phải bảo trì nhiều một khi đã xác lập. Trong một số trường hợp phải biên dịch lại nhân nếu muốn bổ sung phần cứng mới vào hệ thống. Các CD-ROM, sound card bắt buộc phải làm vậy. Nhưng modem, thiết bị chuột hoặc các ổ đĩa cứng thì có thể không cần thiết. Mỗi thiết bị ngoại vi muốn được dùng thì cần phải có những trình điều khiển thiết bị đi kèm. Phần mềm dùng để điều khiển thiết bị gọi là **device driver**. Trong Linux, các **device driver** của nhân Linux thực chất là thư viện dùng chung, thường trú trong bộ nhớ hoặc là các trình điều khiển phần cứng ở mức thấp. Tất cả các thiết bị phần cứng đều được xem như là các tập tin thông thường, chúng có thể được mở, đóng, đọc, ghi bằng cách sử dụng các lời gọi hệ thống giống như các lời gọi hệ thống quản lý tập tin. Mỗi thiết bị được biểu diễn như là một thiết bị tệp đặc biệt (device special file). Ví dụ: như thiết bị đĩa IDE thứ nhất trong hệ thống được biểu diễn bởi **/dev/hda**. Đối với các thiết bị khối (disk) và thiết bị ký tự (character device) thì các thiết bị tệp đặc biệt của chúng được khởi tạo bởi lệnh **mknod** và chúng mô tả thiết bị bằng cách sử dụng các số hiệu chính (major device number) và số hiệu nhỏ (minor device number). Thiết bị mạng cũng được biểu diễn như là một tập tin thiết bị đặc biệt, nhưng chúng được Linux khởi tạo khi khởi sinh bộ điều khiển mạng trong hệ thống.

Các thiết bị được điều khiển bởi một bộ điều khiển chung (driver) sẽ được gán một số (định danh) chung gọi là số hiệu chính**.** Các thiết bị đó được phân biệt thông qua một số gọi là số hiệu nhỏ**.** Ví dụ, mỗi phân vùng (partition) trên một đĩa cứng có một số hiệu nhỏ của mình, vậy **/dev/hda2** (partition thứ hai trên đĩa cứng IDE thứ nhất) có số hiệu chính cho cả thiết bị là 3 và số hiệu nhỏ để phân biệt là 2. Linux ánh xạ một tập tin thiết bị lên một driver thiết bị nhờ sử dụng số hiệu chính của thiết bị và số hiệu của bảng hệ thống.

Linux hỗ trợ 3 loại thiết bị: Thiết bị kí tự, thiết bị khối và thiết bị mạng.

- Thiết bị kí tự, tương ứng với các tập tin đặc biệt trong chế độ kí tự (**`Character Mode`**): Các tập tin này tương ứng với các thiết bị ngoại vi không có cấu trúc, chẳng hạn như các cổng song song hoặc nối tiếp mà trên đó dữ liệu có thể được đọc và ghi theo từng byte hoặc dòng byte.

- Thiết bị kiểu khối, tương ứng với các tập tin đặc biệt trong chế độ khối (**`Block Mode`**): Các tập tin này tương ứng với các thiết bị ngoại vi có cấu trúc dạng khối như ổ đĩa, có kiểu truy cập bằng cách cung cấp một số khối đọc hoặc ghi. Các thao tác nhập/xuất này được thực hiện thông qua một vùng đệm

(**Buffer Cache**) và có thể truy nhập trực tiếp tới từng khối (**Block**) trên thiết bị.

- Thiết bị mạng có thể truy cập thông qua giao diện socket BSD.

Mỗi tập tin đặc biệt sẽ được Linux mô tả theo ba thuộc tính sau:

- Kiểu tập tin (khối hoặc kí tự).
- Số hiệu chính của tập tin, đại diện cho trình điều khiển đang điều khiển thiết bị.
- Số hiệu thứ cấp của tập tin, cho phép trình điều khiển nhận biết thiết bị vật lí mà nó sẽ hoạt động trên đó.

Thông thường các tập tin thiết bị được định vị trong thư mục **/dev**. Các thao tác nhập/xuất vào thiết bị được thực hiện thông qua những lời gọi hệ thống như những thao tác nhập/xuất tập tin thông thường. Mỗi thiết bị ngoại vi được mở bởi lời gọi **open** bằng cách chỉ định tên tập tin đặc biệt tương ứng. Nhân sẽ trả về một trình mô tả nhập/ xuất tương ứng với thiết bị, và tiến trình gọi có thể truy cập nó bằng các lệnh hệ thống **read, write**. Sau khi hoàn thành công việc thì lời gọi **close** sẽ được sử dụng để tắt thiết bị. Linux thường sử dụng hai bảng để lưu trữ danh sách các thiết bị hỗ trợ, đó là: **blkdevs** chứa các chương trình mô tả hay các thiết bị trong chế độ khối, **chrdevs** dành cho các thiết bị có thể truy cập trong chế độ ký tự. Tập tin nguồn **fs./devices.c** chứa các hàm quản lý các thiết bị hỗ trợ.

- ❖ Các hàm **register_blkdevs** và **register_chrdevs** cho phép đăng ký các trình điều khiển thiết bị vào các bảng tương ứng.
- ❖ Các hàm **unregister_blkdev** và **unregister_chrdevs** có nhiệm vụ xoá một đăng ký đã có trong các bảng tương ứng.
- ❖ Các hàm **blkdev_open** và **chrdev_open** sẽ đảm nhiệm việc mở một thiết bị đã được đăng ký.
- ❖ Các hàm **get_blkfops** và **get_chrfops** trả về một con trỏ trỏ vào các thao tác tập tin kết hợp với một thiết bị nhờ **get_blkfops**, sau đó gọi lời gọi **release.**
- ❖ ...

Chương này sẽ lần lượt giới thiệu đến các từng loại các thiết bị: ổ cứng, các cổng nối tiếp cùng modem, các cổng song song cùng máy in và một ít về thiết bị mạng, card sound ... Với mục đích là hướng dẫn phổ dụng nhất, các nội dung dưới đây chỉ dừng lại ở mức sử dụng và chưa đi sâu vào nghiên cứu các can thiệp thiết bị mở mức nhân.

5.2 Các cách quản lý thiết bị lưu trữ trong Linux

Linux có cách điều khiển các thiết bị rất khác biệt so với các hệ điều hành khác. Sẽ không có các tên thiết bị lưu trữ vật lý như ổ A hay ổ C ..., mà lúc đó, các thiết bị lưu trữ này sẽ trở thành một phần của hệ thống tập tin cục bộ thông qua một thao tác được gọi là "kết gắn - mounting". Khi đang sử dụng thiết bị lưu trữ đó, muốn tháo bỏ phải "tháo bỏ kết gắn - umount" thiết bị.

5.2.1 Lệnh mount *và lệnh* umount

* *Lệnh* mount

* Lệnh **mount** được dùng để thông báo cho nhân hệ thống biết là tồn tại một hệ thống tập tin nào đó (đang nằm riêng rẽ và không thể truy cập được) muốn kết nối vào hệ thống tập tin chính tại một điểm gắn nào đó (mount-point). Điểm gắn thường là một thư mục của hệ thống tập tin chính và có thể truy cập dễ dàng.

Để sử dụng bất kỳ một thiết bị lưu trữ vật lý nào trên Linux, đều cần đến lệnh **mount**. Điểm gắn kết là thư mục **/mnt**.

Ví dụ, nếu muốn sử dụng đĩa mềm và đĩa CD, hãy gắn chúng vào hai thư mục **/mnt/floppy** và **/mnt/cdrom** bằng hai lệnh sau:

```
# mount -t msdos /dev/fd0 /mnt/floppy
# mount /dev/cdrom /mnt/cdrom
```

Cú pháp lệnh **mount**:

```
mount [tùy-chọn] <tập-tin-thiết-bị> <thư-mục>
```

Lệnh này thông báo cho nhân hệ thống thực hiện việc kết gắn hệ thống tập tin có trên **tập-tin-thiết-bị** (thiết bị này có **kiểu** nào đó) vào thư mục (điểm kết gắn) là **thư-mục**.

Các tuỳ chọn của lệnh **mount**:

-t <kiểu>

xác định **kiểu** của thiết bị (chẳng hạn **msdos** như ví dụ trên): **kiểu** cũng được sử dụng để xác định kiểu hệ thống tập tin. Các kiểu hệ thống tập tin hiện thời được hỗ trợ có trong tập tin **Linux/fs/filesystems.c**

-h

đưa ra trang trợ giúp.

-a

gắn kết tất cả các tập tin hệ thống (thuộc kiểu được đưa ra) có trong tệp tin **fstab** (đây là tập tin lưu trữ thông tin về trạng thái của các tập tin hệ thống).

-n

gắn kết hệ thống tập tin mà không ghi vào tập tin **/etc/mtab** (đây là tập tin lưu trữ thông tin về các tập tin hệ thống hiện có trên hệ thống). Tùy chọn này cần thiết khi hệ thống tập tin **/etc** chỉ cung cấp quyền đọc.

-r

kết gắn hệ thống tập tin chỉ có quyền đọc.

-w

kết gắn hệ thống tập tin có quyền đọc ghi.

-L nhãn

kết gắn phân vùng được chỉ ra bởi **nhãn.**

-U uuid

kết gắn phân vùng được xác định bởi **uuid**. Hai tùy chọn này chỉ thực hiện được khi tập tin **/proc/partitions** tồn tại (đây là tập tin lưu trữ thông tin về các phân vùng trên hệ thống)

Trên đây là cách chính thức để kết gắn các thiết bị lưu trữ vật lý, nhưng có cách thuận tiện hơn. Bình thường chỉ có người dùng root mới có quyền gắn kết các thiết bị, để mọi người dùng khác có thể kết gắn đĩa mềm hoặc cdrom chẳng hạn, hãy thực hiện như sau:

- với tư cách người dùng root, hãy gõ các lệnh

cấp cho mọi người dùng các quyền truy nhập tới hai thư mục là điểm kết gắn với hai thiết bị đĩa mềm và cdrom

 # chmod a+rwx /mnt/floppy ; /mnt/cdrom

cấp cho mọi người dùng quyền đọc và ghi đối với hai thư mục lưu trữ thiết bị trên hệ thống

 # chmod a+rw /dev/fd0 ; /dev/cdrom

- thêm các dòng sau vào tập tin ***/etc/fstab***

/dev/cdrom /mnt/cdrom iso9660 ro, user, noauto 0 0

/dev/fd0 /mnt/floppy vfat user, noauto 0 0

- bây giờ mọi người dùng đều có thể kết gắn đến đĩa mềm và cdrom đó

 # mount /mnt/floppy

 # mount /mnt/cdrom

Cần ghi nhớ rằng, việc cho phép mọi người dùng có thể **mount** được thiết bị đĩa của mình là điều rất nguy hiểm, vì điều đó có liên quan đến vấn đề bảo mật.

**** Lệnh umount***

Lệnh **umount** cho phép tháo bỏ kết gắn của một hệ thống tập tin trên hệ thống tập tin chính bằng lệnh **umount** với tham số đi kèm là tên của thiết bị lưu trữ hệ thống tập tin đó.

Cú pháp lệnh:

```
umount <thiết-bị>
```

Lệnh này sẽ tháo bỏ kết gắn của hệ thống tập tin có trên **thiết-bị** ra khỏi hệ thống tập tin chính. Chú ý rằng, không thể tháo bỏ kết gắn của một hệ thống tập tin khi nó ”bận” - tức là khi có một tiến trình đang hoạt động truy cập đến các tập tin trên hệ thống tập tin đó.

Các tuỳ chọn lệnh:

-h

hiển thị thông báo trợ giúp và thoát.

-n

loại bỏ các gắn kết mà không ghi vào thư mục **/etc/mtab.**

-v

hiện các chế độ liên quan.

-r

trong trường hợp loại bỏ gắn kết bị lỗi, tùy chọn này sẽ giúp tạo lại gắn kết với chế độ chỉ đọc.

-a

tất cả các tập tin hệ thống được hiển thị trong **/etc/mtab** đã được loại bỏ các gắn kết.

-t kiểu

tùy chọn này cho phép xác định kiểu hệ thống tập tin được tháo bỏ kết gắn. Có thể kết hợp nhiều kiểu hệ thống tập tin cùng lúc bằng cách ngăn chúng bởi dấu "**,**".

-f

bắt buộc phải tháo bỏ các gắn kết.

Ví dụ khi không dùng đến đĩa mềm nữa, có thể dùng lệnh sau:

```
# umount /mnt/fd0
```

Khi một hệ thống tập tin được gắn kết (dùng lệnh **mount**), những thông tin quan trọng về sơ đồ các tập tin trên hệ thống tập tin đó được lưu trong nhân. Nếu loại bỏ thiết bị vật lý chứa hệ thống tập tin mà không tháo bỏ kết gắn (dùng lệnh **umount**) có thể dẫn tới thông tin lưu về hệ thống tập tin bị thất lạc. Mục đích của lệnh **umount** là xóa bỏ mọi thông tin đó ra khỏi bộ nhớ khi không dùng đến nữa.

5.2.2 Các lệnh định dạng đĩa và tạo hệ thống tập tin trong Linux

Định dạng vật lý một thiết bị đĩa là một chuyện, tạo một tập tin hệ thống trên nó lại là một chuyện khác. Nếu như trong DOS, lệnh FORMAT A: thực hiện cả hai công việc trên thì ở trong Linux, đó là hai lệnh riêng biệt. Sau đây là một số lệnh giúp định dạng thiết bị lưu trữ vật lý của mình.

** Ổ đĩa cứng*

Ổ đĩa cứng phải được phân hoạch trước khi có thể định dạng và sử dụng nó. Tương tự như DOS, trong Linux có **fdisk.** Trong Linux có thể tạo các kiểu phân hoạch khác nhau, mỗi phân hoạch được gắn với một chỉ số (index: ID) để thông báo cho hệ điều hành biết kiểu phân hoạch của nó. Dùng các lệnh sau thực hành:

```
# su
```

passwd:

```
# fdisk /dev/hda
```

Command (m for help):

Lệnh trên báo cho **fdisk** biết sẽ làm việc với loại ổ đĩa nào. Nếu dùng đĩa kiểu SCSI thì **hda** sẽ được thay bằng **sda**. Để xem các lệnh của **fdisk**, hãy đánh "**m**".

```
# fdisk /dev/hda
```

Command (m for help):m

```
Hành động lệnh
a           bật/tắt cớ có thể boot được
b           hiệu chỉnh loại thiết bị lưu trữ bsd
c           bật cờ tương thích với thiết bị kiểu DOS
d           xoá một phân vùng
l           danh sách các kiểu phân vùng sẵn có
m   hiển thị trang trợ giúp này
n           thêm một phân vùng mới
o           tạo một bảng phân vùng DOS trống
p           hiển thị bảng phân vùng trên hệ thống
q           thoát và không ghi mọi sự thay đổi
s           tạo một phân vùng cho loại thiết bị lưu trữ SUN
t           thay đổi chỉ số phân vùng hệ thống
u           thay đổi các đơn vị hiển thị
v           kiểm tra bảng phân vùng
w   ghi sự thay đổi trên bảng phân vùng và thoát
x           các tính năng mở rộng (chỉ dành cho các chuyên gia)
Command (m for help):p
Disk /dev/hda: 64 heads, 63 sectors, 847 cylinders
Units = cylinders of 4032 * 512 bytes

Device      Boot  Start  End  Bloocks    Id   System
/dev/hda1   *     1      817  1647040+   83   Linux
/dev/hda2         818    847  60480      82   Linux swap

Command (m for help):q
        #
```

Các cột thông báo trên đây có ý nghĩa như sau:

- Cột **Device** có mục thiết bị dành cho phân vùng trong **/dev**, chẳng hạn **/dev/hda1**.
- Cột **Boot** chỉ xem phân vùng đó có khả năng khởi động hay không, nếu có khả năng khởi động thì có dấu '*'.
- Cột **Start, End** chỉ ra chỉ số trụ (cylinder) đầu và cuối của phân vùng.
- Cột **Bloocks** chỉ ra dung lượng của phân vùng (là số lượng tính theo khối 1KB).
- Cột **Id** và **System** chỉ số ID và ý nghĩa của ID đó, ví dụ ID = 82 có

ý nghĩa là **`Linux swap`**.

Lệnh **l** đưa ra danh sách các kiểu phân vùng sẵn có với thông tin chi tiết hơn.

Sau khi thực hiện lệnh **fdisk**, hãy sử dụng lệnh **mkfs** với tư cách **root** rồi định dạng phân vùng theo một hệ thống tập tin cụ thể (file system).

* *Xây dựng môt hê thống tâp tin trên Linux với lênh* mkfs

Cú pháp lệnh:

```
mkfs [tùy-chọn] <hệ-thống-tập-tin> [khối]
```

Lệnh mkfs thường được sử dụng để xây dựng một hệ thống tập tin trên thiết bị, thường là phân vùng đĩa cứng. **`hệ-thống-tập-tin`** hoặc là tên thiết bị (ví dụ **/dev/hda1**, **/dev/sdb2**) hoặc là điểm kết nối tập tin hệ thống (ví dụ **/**, **/usr**, **/home**); **`khối`** là số khối được sử dụng cho hệ thống tập tin.

-t `kiểu`

tùy chọn này xác định kiểu tập tin hệ thống được xây dựng. Nếu không có tùy chọn này, kiểu tập tin hệ thống mặc định sẽ được sử dụng (hiện tại là kiểu **ext2**).

-c

kiểm tra thiết bị để tìm ra các khối hỏng trước khi xây dựng hệ thống tập tin.

Giờ đây sau khi xác lập xong phân hoạch, và hệ tập tin dùng trên đó, có thể ráp nối và bắt đầu dùng hệ tập tin. Hãy đăng nhập với tư cách quản trị (người dùng **root**) và dùng lệnh **mount**. Sau khi sử dụng xong có thể tiến hành tháo kết nối với lệnh **umount.**

* *Đinh dang mức thấp môt đĩa mềm (lênh* fdformat*)*

Cú pháp lệnh định dạng mức thấp một đĩa mềm:

```
fdformat [-n] thiết-bị
```

Lệnh fdformat thực hiện việc định dạng một đĩa mềm ở mức thấp. Tham số **`thiết-bị`** thường là một trong các loại sau (chỉ định dạng với đĩa mềm, với chỉ số thiết bị là **major=2** - đây là chỉ số xác định kiểu của thiết bị, và **minor** - chỉ số xác định số của thiết bị trong trường hợp có nhiều thiết bị cùng loại):

/dev/fd0d360 (minor = 4)

/dev/fd0h1200 (minor = 8)

/dev/fd0D360 (minor = 12)

/dev/fd0H360 (minor = 12)

/dev/fd0D720 (minor = 16)

/dev/fd0H720 (minor = 16)

/dev/fd0h360 (minor = 20)

/dev/fd0h720 (minor = 24)

/dev/fd0H1440 (minor = 28)

/dev/fd1d360 (minor = 5)

/dev/fd1h1200 (minor = 9)

/dev/fd1D360 (minor = 13)

/dev/fd1H360 (minor = 13)

/dev/fd1D720 (minor = 17)

/dev/fd1D720 (minor = 17)

/dev/fd0H720 (minor = 16)

/dev/fd1h360 (minor = 21)

/dev/fd1h720 (minor = 25)

/dev/fd1H1440 (minor = 29)

Các tùy chọn lệnh:

-n

cho phép bỏ qua các kiểm tra được thực hiện sau khi định dạng đĩa.

Các đĩa mềm, **/dev/fd0** và **/dev/fd1**, sẽ gặp lỗi nếu dùng lệnh **format** khi kiểu định dạng là không chuẩn.

**** Thêm hệ thống tập tin vào đĩa mềm đã được định dạng với lệnh* mformat**

Lệnh **mformat** được sử dụng để thêm một hệ thống tập tin MS-DOS vào một đĩa mềm định dạng cấp thấp với cú pháp là:

```
mformat <các-tùy-chọn> <ổ-đĩa>:
```

mformat thêm một hệ thống tập tin MS-DOS tối thiểu (boot sector, FAT, và thư mục gốc) lên một đĩa đã định dạng bằng một định dạng cấp thấp.

Các tùy chọn sau được hỗ trợ (Tuỳ chọn **-S**, **-2**, **-1** và **-M** có thể không có nếu bản **mtools** được biên dịch không có tuỳ chọn **USE_2M**):

-t `số-trụ`

số lượng các trụ (cylinders).

-h `số-mặt`

số lượng các mặt đĩa (heads).

-s `số-sector`

số lượng các sectors trên mỗi rãnh (track). Nếu tuỳ chọn **2m** được đặt trước, là số lượng sector 512 byte tương đương với các track cùng loại (tức là không có head 0, track 0). Nếu tuỳ chọn **2m** không được đặt, là số lượng các sector vật lý trên mỗi rãnh (sector có thể lớn hơn 512 byte).

-l `tên-nhãn-đĩa`

tuỳ chọn tên nhãn đĩa.

-S `mã-kích-thước`

Kích thước của sector là 2^ (`mã-kích-thước` +7).

-2 sector-00

định dạng **2m**. Tham số của tuỳ chọn này (**sector-00**) miêu tả số lượng các sector trên rãnh 0, mặt 0. Tuỳ chọn này thường dùng cho các sector lớn hơn thông thường.

-1

không sử dụng định dạng **2m**, thậm chí khi kiểu đĩa (geometry) hiện thời là kiểu định dạng **2m**.

-M cỡ-mềm

kích thước sector phần mềm là **cỡ-mềm**. Tham số này mô tả kích thước sector trong các byte được sử dụng bởi hệ thống tập tin MS-DOS. Trong chế độ ngầm định thì đây là kích thước vật lý của sector.

-X

định dạng đĩa như một đĩa XDF (1 loại định dạng đĩa dung lượng lớn được sử dụng bởi OS/2). Các đĩa đã được định dạng cấp thấp sử dụng tiện ích **xdfcopy** nằm trong gói (package) **fdutils.**

-C

tạo tập tin ảnh đĩa để cài đặt hệ thống tập tin MS-DOS trên đó. Rõ ràng, điều này vô dụng trên các thiết bị vật lý chẳng hạn các ổ đĩa mềm và các phân vùng ổ cứng.

-H lượng-bị-che

số lượng các sector ẩn là **lượng-bị-che**. Tham số này rất hữu ích cho việc định dạng các phân vùng ổ cứng, với các đường biên rãnh không được sắp thẳng hàng. (Chẳng hạn, mặt đầu tiên của rãnh đầu tiên không thuộc phân vùng nhưng lại chứa một bảng phân vùng). Trong trường hợp này, số lượng của các sector ẩn chung với số lượng các sector trên trụ. Điều này đang được kiểm chứng.

-n

số serial.

-F

định dạng phân vùng như FAT32 (thực nghiệm).

-I phiên-bản

đặt **phiên-bản** (fsVersion ID) khi định dạng một ổ đĩa FAT32. Để nhận biết được điều này, chạy **minfo** trên một ổ đĩa FAT32 đang tồn tại.

-c dài-clustor

đặt kích thước của một cluster (theo sector) là **dài-clustor**. Nếu kích thước cluster này tạo ra một bảng FAT quá lớn với số lượng các bit của nó, **mtools** sẽ tự động tăng kích thước cluster, cho đến khi bảng FAT nhỏ xuống phù hợp.

-r cỡ-gốc

đặt kích thước của thư mục gốc là **cỡ-gốc** (theo sector). Chỉ thích hợp cho bảng FAT 12 bit và 16 bit.

-B boot-sector

sử dụng boot sector được lưu trong tập tin hay device cho trước (theo tham số **boot-sector**), thay vì sử dụng boot sector của nó. Chỉ có các trường định dạng được cập nhật để phù hợp với các tham số đĩa đích.

-k

giữ các boot sector đang tồn tại càng nhiều càng tốt. Chỉ có các trường định dạng được cập nhật để phù hợp các tham số đĩa đích.

-0 tỷ-lệ-0

đặt tỷ lệ chuyển dữ liệu trên rãnh 0 là **tỷ-lệ-0.**

-A tỷ-lệ

đặt tỷ lệ chuyển dữ liệu trên các rãnh khác rãnh 0 là **tỷ-lệ**.

Để định dạng một đĩa có mật độ khác ngầm định, phải cung cấp (ít nhất) các tham số dòng lệnh khác với ngầm định trên đây.

5.2.3 Lệnh quản lý đĩa

** Xem dung lượng đĩa đã sử dụng với lệnh* du*:*

Linux cho phép người dùng xem thông tin về dung lượng đĩa đã được sử dụng bằng lệnh **du** với cú pháp:

```
du [tùy-chọn] ... [tập-tin] ...
```

Lệnh **du** liệt kê kích thước (tính theo kilobytes) của mỗi tập tin thuộc vào hệ thống tập tin có chứa **tập-tin** được chỉ trong lệnh.

Các tuỳ chọn là:

-a

liệt kê kích thước của tất cả các tập tin có trong hệ thống tập tin lưu trữ **tập-tin.**

-b, --bytes

hiển thị kích thước theo byte.

-c, --total

hiển thị cả tổng dung lượng được sử dụng trong hệ thống tập tin.

-D, --dereference-args

không tính kích thước các tập tin được liên kết đến nếu chúng nằm trên các thư mục khác.

-h, --human-readable

hiển thị kích thước các tập tin kèm theo đơn vị tính (ví dụ: 1K, 234M, 2G ...).

-k, --kilobytes

hiển thị kích thước tính theo kilobytes.

-L, --dereference

tính cả kích thước của các tập tin được liên kết tới.

-l, --count-links

tính kích thước các tập tin nhiều lần nếu được liên kết cứng.

-m, --megabytes

tính kích thước theo megabytes.

-S, --separate-dirs

không hiển thị kích thước của thư mục con.

-s

đưa ra kích thước của hệ thống tập tin có lưu trữ **tập-tin**.

-x, --one-file system

bỏ qua các thư mục trên các hệ thống tập tin khác.

--help

hiển thị trang trợ giúp và thoát.

Cần lưu ý rằng, lệnh **du** không cho phép có nhiều tùy chọn trên cùng một dòng lệnh.

Ví dụ: lệnh sau cho biết kích cỡ của các tập tin trong thư mục **/usr/doc/test**:

```
# du /usr/doc/test
28      ./TODO/1.0_to_1.5
24      ./TODO/lib++
16      ./TODO/unreleased
12      ./TODO/unstable
144     ./TODO
44      ./code
160     ./languages
56      ./licences
532     .
```

Nhìn vào màn hình có thể biết được kích thước của tập tin **./TODO/1.0_to_1.5** là 28 KB, tập tin **./TODO/lib++** là 24 KB, ..., và kích thước của thư mục hiện thời là 532 KB.

**** Kiểm tra dung lương đĩa trống với lênh* df:**

Cú pháp lệnh:

```
df [tùy-chọn]... [tập-tin]...
```

Lệnh này hiển thị dung lượng đĩa còn trống trên hệ thống tập tin chứa **tập-tin**. Nếu không có tham số **tập-tin** thì lệnh này hiển thị dung lượng đĩa còn trống trên tất cả các hệ thống tập tin được kết nối.

Các tùy chọn là:

-a, --all

bao gồm cả các tập tin hệ thống có dung lượng là 0 block.

--block-size=cỡ

thiết lập lại độ lớn của khối là **cỡ** byte.

-k, --kilobytes

hiển thị dung lượng tính theo kilobytes.

-l, --local

giới hạn danh sách các tập tin cục bộ trong hệ thống.

-m, --megabytes

hiển thị dung lượng tính theo megabytes.

-t, --type=kiểu

giới hạn danh sách các tập tin hệ thống thuộc **kiểu.**

-T, --print-type

hiển thị các kiểu của tập tin hệ thống.

--help

đưa ra trang trợ giúp và thoát.

Để chỉ ra được dung lượng đĩa còn trống trong Linux không phải là điều dễ làm. Người dùng có thể sử dụng lệnh **df** để làm được điều này, tuy nhiên kết quả của lệnh này chỉ cho biết dung lượng đĩa đã được sử dụng và dung lượng đĩa còn trống của từng hệ thống tập tin. Nếu muốn biết tổng dung lượng đĩa còn trống là bao nhiêu, sẽ phải cộng dồn dung lượng đĩa còn trống của từng hệ thống tập tin.

Ví dụ, lệnh

```
# df /mnt/floppy
```

sẽ cho kết quả như sau trên màn hình (dòng đầu tiên là tên cột):

Filesystem	1k-blocks	Used	Available	Use%	Mounted on
/dev/hda2	2174808	1378228	686104	67%	/
none	0	0	0	-	/proc
none	0	0	0	-	/dev/pts
automount(pid411)	0	0	0	-	/misc
/dev/fd0	1423	249	1174	18%	/mnt/floppy

có thể xác định được, đĩa mềm đã được sử dụng 18%, như vậy là còn 82% (tức là còn 1174 KB) dung lượng đĩa chưa được sử dụng.

- Cột **Filesystem** chứa tên của thiết bị đĩa, cột **1k-blocks** chứa dung lượng của thiết bị, cột **Used** chứa dung lượng đĩa đã được sử dụng, cột **Available** chứa dung lượng đĩa còn trống, cột **Use%** chứa % dung lượng đĩa đã sử dụng và cột **Mounted on** chứa điểm kết gắn của thiết bị.

Cách nhanh nhất để biết được dung lượng đĩa còn trống bao nhiêu là phải xác định được tên của một thư mục bất kỳ có trong đĩa đó, sử dụng lệnh **df** với tham số **tập-**

tin là tên của thư mục. Sau đó đọc nội dung cột **Available** trên màn hình hiển thị để biết dung lượng đĩa còn trống.

Chẳng hạn, trên đĩa cứng đang sử dụng có thư mục **/etc**, khi đó gõ lệnh:

```
# df /etc
```

kết quả hiển thị lên màn hình như sau:

```
   Filesyst       1k-    Used      Available  Use%       Mounted
em             blocks                                 on
   /dev/hda     1984240     14171     466252     75%     /
1                       92
```

cho biết đĩa còn có 466252 khối rỗi.

5.3 Các cổng nối tiếp và modem

Các cổng nối tiếp, ví dụ COM1 trong DOS hoặc Window, được Linux xem là **/dev/ttyS0** và **/dev/ttyS** dành cho dữ liệu đến và **/dev/cua** dành cho dữ liệu đi.

Để xác lập trong nhân kiểu Modem hiện có, sử dụng chương trình **setserial**.

Trong quá trình khởi động bình thường thì các cổng nối tiếp sẽ sử dụng những địa chỉ cổng vào/ra và các IRQ mặc định. Tuy nhiên để thay đổi những giá trị này ta có thể sử dụng **setserial**. Chương trình **/etc/rc.d/rc.serial** tự động xác lập hầu hết các modem mỗi khi khởi động.

Ví dụ:

```
# setserial b/dev/cua0 irq 15 autoconfig
```

Lệnh trên sẽ đặt lại ngắt dùng cho dữ liệu đi của COM1 là IRQ15. Sau khi đã xác lập modem, có thể sử dụng một số tiện ích để quay số như minicom, seyon ...

5.4 Các cổng song song và máy in

Để sử dụng máy in, phải đảm bảo có phần hỗ trợ máy in trong nhân và đảm bảo IRQ máy in (7) bình thường chưa có thiết bị khác sử dụng.

Nếu máy in nằm trên cổng song song đầu tiên, với tư cách siêu người dùng, có thể nhập lệnh dưới đây để đảm bảo phần mềm máy in và các tuyến giao kết làm việc tốt.

5.4.1 Khởi tao và thiết lập máy in trong lpd

Việc sử dụng máy in có thể gây rắc rối cho người dùng. Tuy nhiên nếu quan tâm thêm một số hiểu biết về phần cứng, người dùng cũng có thể thiết lập cấu hình cho máy in một cách dễ dàng.

Hầu hết các máy in trên hệ thống Linux đều được điều khiển bởi một chương trình chạy ngầm được gọi là **daemon lpd**: chương trình này được khởi động cùng với hệ thống. Trong suốt quá trình khởi động, **daemon lpd** sẽ đọc tập tin **/etc/printcap** (tập tin này lưu trữ thông tin về máy in được thiết lập trên hệ thống) để nhận dạng các phần được áp dụng cho bất kỳ máy in nào gắn vào hệ thống.

Nói chung, không cần thay đổi **daemon lpd**, tuy nhiên đôi lúc cần phải nạp lại nó. Lệnh nạp **daemon** này là **lpd**.

Cú pháp lệnh:

```
lpd [-l] [port#]
```

Các tham số:

-l

Tuỳ chọn cho phép kích hoạt hệ thống ghi nhật ký lưu ý đến mỗi yêu cầu của máy in. Tuỳ chọn này sẽ hữu ích khi đang gỡ rối hệ thống máy in.

port#

xác định số cổng Internet được sử dụng để kiểm tra thông tin cấu hình của hệ thống có bị ghi đè không.

Việc cài đặt **lpd** khá đơn giản, nó cho phép có thể giữ lại các tập tin trong hàng đợi in và từ từ in chúng. Tuy nhiên khi gặp rắc rối, **lpd** sẽ không gửi thông báo nào cho người dùng, và điều đó có thể gây khó khăn cho người dùng, nhưng đây là bước khởi đầu để tìm hiểu về công việc in ấn.

Về cơ bản, để thêm một máy in vào hệ thống **lpd**, phải thêm một đầu vào máy in vào trong tập tin **/etc/printcap**, và tạo một thư mục hàng đợi như: **/var/spool/lpd**.

Một đầu vào trong tập tin **/etc/printcap** có dạng sau (dấu hàng rào **#** tại dòng đầu tiên giải thích cho đầu vào, không phải là dấu nhắc shell):

```
# LOCAL djet500
```

lp|dj|deskjet:

:sd=/var/spool/lpd/dj:

:mx#0:

:lp=/dev/lp0:

:sh:

Các dòng trên định nghĩa một hàng đợi in gọi là **lp**, **dj** hoặc **deskjet**, thư mục hàng đợi là: **/var/spool/lpd/dj** và không giới hạn kích thước lớn nhất đối với mỗi công việc in ấn, sử dụng thiết bị máy in **/dev/lp0**, và không có lề và không thêm tên của người in vào trước mỗi trang in.

Để hiểu thêm chi tiết về các đầu vào cần đọc trang **man** về tập tin **printcap**.

Định nghĩa trên nhìn có vẻ rất đơn giản, nhưng sẽ có một vướng mắc khó chịu (trừ khi gửi một tập tin mà máy DeskJet 500 có thể hiểu được) là DeskJet 500 sẽ in những dòng rất lạ.

Ví dụ, khi gửi một tập tin văn bản thông thường của Linux tới DeskJet 500, kết quả là nó in ra những dòng mới như sau:

This is line one.

This is line two.

This is line three.

Khi gửi một tập tin PostScript tới hàng đợi in thì hàng đợi sẽ tiếp nhận được một danh sách lệnh PostScript. Hàng đợi được in ra với hiệu ứng bậc thang (từ lệnh PostScript) tuy nhiên lại không có xảy ra điều gì đối với các trang in đối với máy in Postscript (như DeskJet 500, chẳng hạn).

Như vậy, cần phải thêm vào một bộ phận với chức năng lọc. Khi đọc trang **man** về tập tin **printcap**, cần lưu ý đến hai thuộc tính **if** và **of. if** (input filter) chính là bộ phận cần được bổ sung trong trường hợp nói trên.

Trong trường hợp viết một đoạn chương trình lọc nhớ bằng shell script thì đoạn chương trình này sẽ điều khiển việc trả về một dòng mới khi in và lúc đó hiệu ứng bậc thang sẽ được loại bỏ. Do vậy, phải thêm vào dòng **if** trong đầu vào **printcap** ở trên như dưới đây:

```
lp|dj|deskjet:\
:sd=/var/spool/lpd/dj:\
:mx#0:\
:lp=/dev/lp0:\
:if=/var/spool/lpd/dj/filter:\
:sh:
```

Sau đây là ví dụ về một đoạn chương trình lọc đơn giản:

```
#!perl
# The above line should really have the whole path to perl
# This script must be executable: chmod 755 filter
 while(<STDIN>){chop $_; print "$_\r\n";};
# You might also want to end with a form feed: print "\f";
```

Khi hoàn thiện các việc như trên, chúng ta đã có một hàng đợi in cho phép in các tập tin văn bản thông thường theo đúng nghĩa của nó.

Chỉ còn một vấn đề là khi in ấn văn bản thuần túy thì không nhanh bằng khi in PostScript và các kiểu đồ họa hay định dạng khác. Vấn đề này rất dễ giải quyết bằng phương thức đơn giản là mở rộng vào bộ lọc trên. Nếu viết một bộ lọc có thể chấp nhận các loại tập tin là đầu vào và đưa ra đầu ra cho mỗi trường hợp, khi đó sẽ có một hàng đợi in thông minh hơn.

Những bộ lọc như vậy gọi là **`magic filter`**. Không nên vô cớ viết các bộ lọc này trừ khi máy in in ra những dấu hiệu lạ. Có rất nhiều các bộ lọc đã được viết sẵn trên mạng. ASP là một trong số những bộ lọc tốt nhất, hoặc hệ thống Linux đang sr dụng đã có sẵn công cụ cài đặt máy in sẽ tự động tạo ra các bộ lọc như vậy.

Có một điều lưu ý đối với các bộ lọc: một vài phiên bản cũ của **lpd** không chạy các bộ lọc **if** đối với các máy in mạng.

5.4.2 Các lênh in ấn cơ bản

Phần này trình bày các cách thức làm việc với máy in, ví như, làm như thế nào để biết được máy in nào kết nối trên hệ thống, làm thế nào để đưa dữ liệu cần in ra máy in, làm thế nào để kiểm tra hay xóa bỏ các dữ liệu trong hàng đợi in v.v. thông qua việc giới thiệu một số lệnh cơ bản làm việc với máy in.

** In môt tập tin với lênh* lpr

Phương pháp đơn giản nhất để in trên hệ thống Linux là gửi các tập tin cần in trực tiếp tới thiết bị in ấn (máy in). Với tư cách người sử dụng **root**, có thể dùng lệnh **cat** thực hiện được phương pháo đó:

```
# cat thesis.txt > /dev/lp
```

Trong lệnh nói trên, **/dev/lp** là một liên kết tĩnh đến một máy in thực sự như: máy in ma trận điểm, máy in laser .v.v.

Do vấn đề bảo mật, chỉ người sử dụng **root** và những người sử dụng cùng nhóm mới có quyền đưa thông tin trực tiếp tới máy in.

Linux cho phép sử dụng các lệnh **lpr**, **lprm**, **lpq** để in ấn văn bản mà không đòi hỏi vấn đề bảo mật quá chặt chẽ.

Cũng do chính vấn đề bảo mật, người sử dụng thông thường phải dùng lệnh **lpr** để in các tập tin.

Khi lệnh **lpr** được chạy, đầu tiên lệnh sao chép một bản của tập tin cần in vào thư mục hàng đợi in, tại đây tập tin này sẽ được lưu giữ đến khi chúng được đưa ra máy in. Khi một tập tin trên hàng đợi được lượt in, **lpd** sẽ sao chép một bản sao của tập tin và chính bản sao mới được in ra trong khi bản chính thức vẫn nằm trong hàng đợi in. Chính cách thức như vậy tạo ra tình huống có thể có rất nhiều công việc in ấn được đợi tại một thời điểm.

Cú pháp lệnh:

```
lpr [tùy-chọn] [tập-tin]
```

Nếu **tập-tin** không có trong lệnh, **lpr** sẽ in ra mọi thứ tại đầu vào chuẩn (thông thường là bàn phím hoặc đầu ra của chương trình khác). Việc này cho phép người sử dụng điều hướng các đầu ra của lệnh tới hàng đợi in.

Ví dụ:

```
# cat thesis.txt | lpr
```

hoặc

```
# lpr -160 thesis.txt
```

Lệnh **lpr** chấp nhận một vài lớp tuỳ chọn cho phép người sử dụng điều khiển công việc in ấn và dưới đây là một số tùy chọn thông dụng:

- Các tùy chọn được sử dụng để in các tập tin không thuộc dạng văn bản thông thường:

  -c

  in các tập tin chứa dữ liệu được tạo từ **cifplot(l).**

  -d

  in các tập tin dữ liệu được tạo từ **tex(l)** (một kiểu văn bản **latex**, dạng **DVI** từ **Stanford**).

  -l

  sử dụng một bộ lọc cho phép các ký tự điều khiển sẽ được in ra và ngăn chặn sự ngắt trang.

  -p

  sử dụng **pr(1)** để định dạng tập tin (giống như **print**).

- Các tùy chọn điều khiển việc in ấn:

  -P

xác định máy in để in. Bình thường, máy in thiết lập mặc định, hoặc máy in có tên là nội dung của biến môi trường **PRINTER** được sử dụng.

-m

sau khi in xong sẽ gửi một thông báo đã hoàn thành.

-r

loại bỏ tập tin trên bộ đệm khi đã in xong. Không sử dụng kết hợp với tuỳ chọn **-s** vì liên quan đến vấn đề bảo mật.

-s

tạo một liên kết thay vì sao chép một tập tin tới hàng đợi in. Bình thường khi một tập tin được in, nó sẽ được sao vào bộ đệm, nếu dùng tuỳ chọn **-s** thì nó sẽ tạo ra một liên kết tới tập tin cần in. Khi in, thông qua mối liên kết này để lấy dữ liệu. Vì thế, tập tin sẽ không bị thay đổi hoặc bị xóa bỏ cho đến khi nó được in xong.

- Các tùy chọn liên quan đến việc sao chép, hiển thị trang in ...:

-P máy-in

xác định máy in sử dụng là **máy-in**.

-#số

xác định số lượng bản in là **số**.

Ví dụ:

```
# lpr -#2 -sP dj thesis.txt
```

Lệnh này sẽ tạo ra một liên kết của tập tin **thesis.txt** tới thư mục hàng đợi in của máy in **dj** và sẽ in ra làm 2 bản.

-[1234]font

chỉ ra một font chữ được đưa lên tại một vị trí trong danh sách các font. Daemon sẽ xây dựng một tập tin **.railmag** để chỉ ra đường dẫn của font đó.

-T tiêu-đề

tên tiêu đề trong **pr(l)** được thay thế cho tên tập tin.

-i [số-cột]

thiết lập khoảng cách giữa các dòng, được biểu thị bằng **số-cột** các ký tự trống.

-w số

thiết lập khoảng cách giữa hai từ liên tiếp là **số** kí tự rỗng.

* *Đinh dang tâp tin trước khi in với lênh* pr

Sử dụng lệnh **lpr** thì kết quả in ra nói chung là tốt, tuy nhiên, kết quả có vẻ đơn điệu. Nếu muốn có tiêu đề là tên tập tin và số trang trên mỗi trang in, hay muốn định dạng trang in theo một số đặc tính nào đó, cần sử dụng lệnh **pr**. Không hỗ trợ trực tiếp

cho việc in ấn, **pr** là một lệnh giúp cho việc định dạng và ngắt trang, thường được sử dụng để hiển thị thông tin trên màn hình.

Cú pháp lệnh:

```
pr [tùy-chọn] ... [tập-tin] ...
```

Lệnh này có tác dụng chuyển định dạng các tập tin văn bản sang định dạng của máy in.

Các tuỳ chọn thông dụng nhất:

-<số-cột>, --columns=<số-cột>

phân nội dung của tập tin thành **số-cột** cột, số hàng trong mỗi cột là như nhau, nội dung tập tin được bắt đầu theo chiều dọc từ cột 1 cho đến hết, trừ khi sử dụng tùy chọn **-a**.

-a, --across

tùy chọn này được sử dụng kết hợp với **-<số-cột>**, nó có tác dụng phân nội dung của tập tin thành **số-cột** cột nhưng nội dung tập tin được bắt đầu theo chiều ngang từ cột 1, cột 2, ... rồi trở về cột 1.

-d, --double-space

khoảng cách giữa các dòng sẽ tăng gấp đôi.

-F, -f, --form-feed

sử dụng lệnh cưỡng bức in đẩy trang hiện tại ra và bắt đầu một trang mới.

-h tiêu-đề, --head=tiêu-đề

sử dụng **tiêu-đề** đặt ở giữa thay cho tên tập tin trong đầu của mỗi trang. In một dòng trống không sử dụng **-h**.

-l số-dòng, --length=số-dòng

thiết lập số dòng trong trang (**số-dòng**), ngầm định trong văn bản là 56 dòng, với **-F** là 63.

-t, --omit-header

bỏ qua những đầu mục và các dấu phân cách trang.

Ví dụ, giả sử có tập tin **test** có nội dung như sau:

```
# cat test
```

```
PID TTY TIME CMD
 1 ?   00:00:04 init
 2 ?   00:00:00 kflushd
 3 ?   00:00:00 kupdate
 4 ?   00:00:00 kpiod
```

5 ? 00:00:03 kswapd

6 ? 00:00:00 mdrecoveryd

330 ? 00:00:00 portmap

345 ? 00:00:00 lockd

346 ? 00:00:00 rpciod

355 ? 00:00:00 rpc.statd

369 ? 00:00:00 apmd

396 ? 00:00:00 automount

449 ? 00:00:00 syslogd

```
# pr -2 test
```

PID TTY TIME CMD 345 ? 00:00:00 lockd

1 ? 00:00:04 init 346 ? 00:00:00 rpciod

2 ? 00:00:00 kflushd 355 ? 00:00:00 rpc.statd

3 ? 00:00:00 kupdate 369 ? 00:00:00 apmd

4 ? 00:00:00 kpiod 396 ? 00:00:00 automount

5 ? 00:00:03 kswapd 449 ? 00:00:00 syslogd

6 ? 00:00:00 mdrecoveryd 458 ? 00:00:00 klogd

330 ? 00:00:00 portmap

2000-11-25 14:21 vd2 Page 1

Lệnh trên đã chia nội dung tập tin **test** ra làm hai cột nội dung.

** Làm việc với hàng đợi in thông qua lệnh* lpq

Trên một máy tính cá nhân, có thể sử dụng trực tiếp máy in kết nối vào hệ thống thông qua các lệnh hoặc tiện ích. Nhưng Linux không cho phép điều khiển trực tiếp máy in của riêng mỗi người dùng. Thay vào đó, Linux điều khiển các yêu cầu in thông qua một hàng đợi, ở đây lưu trữ danh sách các tập tin cần in. Khi dùng lệnh **lpr** để in một tập tin, yêu cầu in được bổ sung vào hàng đợi các tập tin đang chờ in. Yêu cầu in mới luôn được đặt vào cuối của hàng đợi.

Để hiển thị danh sách các ấn bản nằm trong hàng đợi in, sử dụng lệnh **lpq**, nếu không có tham số máy in thì lệnh đưa ra danh sách các ấn bản trong hàng đợi in của máy in ngầm định.

Cú pháp lệnh:

```
lpq [-l] [-P máy-in] [số-hiệu-in...] [người-dùng...]
```

Giải thích về các tham số:

-l

hiển thị thêm thông tin về mỗi mục nhập trong hàng đợi. Thường thì chỉ có một thông tin hiển thị.

-P máy-in

xác định tên máy in muốn sử dụng.

Ví dụ:

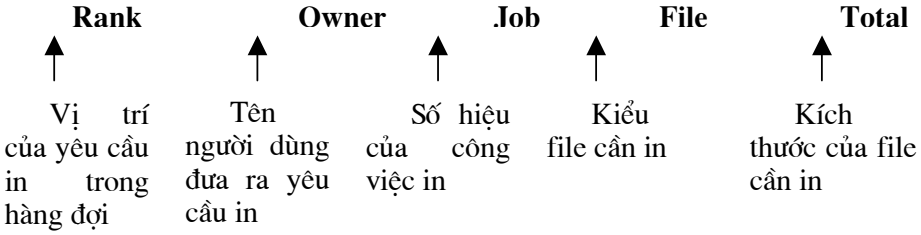
```
# lpq
```

Rank Owner Job File Total Size

1st root 9 (standard input) 97 bytes

2nd taylor 10 (standard input) 4142 bytes

Mỗi một yêu cầu in đều được gán một số các thông tin cần thiết, sau đây là ý nghĩa của chúng:



Để giới hạn chỉ đưa ra các yêu cầu in của một người dùng cụ thể, hãy xác định tên người dùng:

```
# lpq taylor
```

Rank	Owner	Job	File	Total Size
1st	taylor	10	standard input	4142 bytes

* *Xóa bỏ hàng đợi in với lệnh* lprm

Linux cho phép quản lý tốt hệ thống in ấn **lpd** bao gồm cả khả năng huỷ bỏ các công việc in ấn khi chúng còn nằm trong hàng đợi in bằng lệnh **lprm** với cú pháp:

```
lprm [-P máy-in] [-] [số-hiệu-in...] [người-dùng...]
```

Lệnh **lprm** xóa bỏ một hoặc nhiều yêu cầu trên hàng đợi in.

Các tham số được giải thích như sau:

-P máy-in

Hủy việc in các tài liệu trên hàng đợi in của **máy-in** được chỉ ra.

-

nếu chỉ một dấu trừ '**-**' được đưa ra, lệnh **lprm** hủy việc in tất cả các tài liệu đợi in của người dùng gõ lệnh này (Khi người dùng đăng nhập với tư cách **root** thì hủy bỏ việc in mọi tài liệu đang đợi in).

số-hiệu-in

cho phép hủy bỏ một công việc in riêng biệt bằng cách chỉ ra số hiệu của **số-hiệu-in** (lệnh **lpq** cho biết số hiệu đó).

người-dùng

loại bỏ bất kỳ công việc nào trong hàng đợi in thuộc về người dùng đó.

Ví dụ, lệnh

```
# lprm -P HPLaserJetIII
```

sẽ xoá bỏ tất cả tài liệu trong hàng đợi của máy in **HPLaserJetIII**.

Ví dụ, có dòng lệnh:

```
# lpq -l
```

```
Rank   Owner Job   File              Total Size
1st    taylor  10    standard input 4142 bytes
```

```
#
# lprm -10
```

Lệnh này huỷ bỏ công việc in ấn có số hiệu 10 (yêu cầu in của **taylor**).

* *Lênh* lpc

Lệnh **lpc** được sử dụng để điều khiển máy in và các dịch vụ **lpd**. Lệnh **lpc** cho phép vô hiệu hoá hoặc làm cho có hiệu lực đối với máy in hay hàng đợi nào đó, sắp xếp lại các công việc trong hàng đợi in và giám sát trạng thái của các máy in hay hàng đợi của chúng.

Cú pháp:

```
lpc [tùy-chọn][lệnh [đối-số ...] ]
```

Lệnh này khởi động chương trình **lpc**. Ngầm định, có thể sử dụng lệnh này ở chế độ tương tác và có thể bắt đầu các lệnh làm việc với **lpc**. Các lựa chọn khác được sử dụng với lệnh **lpc** trong chương trình **lpc** cũng như trong chế độ tương tác. Tham số của lệnh **lpc** rất phong phú (tham khảo trang **man** của lệnh **lpc** để có được nội dung đầy đủ), ở đây giới thiệu những nội dung thông dụng nhất.

Các lệnh trong **lpc**:

?[command...] hoặc

help[command...]

cho một sự mô tả ngắn gọn về mỗi lệnh được chỉ ra trong danh sách các đối số hoặc các đối số không được đưa ra thì danh sách các lệnh được bỏ qua.

abort {**all | printer**}

lệnh này gần giống lệnh **stop**, điều khác ở đây là nó không cho phép bất kỳ tài liệu nào hiện đang được in dừng lại trước khi máy in ngừng. Bất kỳ tài liệu nào chấm dứt bất thường bởi lệnh **abort** đều xếp hàng lại khi khởi động lại máy in.

clean {**all | printer**}

loại bỏ mọi tài liệu đang trong bộ đệm của máy in, kể cả các tài liệu đang được in dở. Thông thường, tài liệu in hiện hành sẽ tiến hành bình thường vì tài liệu này sẽ được chuyển đến daemon in hoặc đến bộ đệm của máy in. Nhưng ở đây tất cả các tài liệu khác sẽ bị xoá đi. Nếu sử dụng đối số **all** thì những máy in nào có hàng đợi in sẽ bị xoá đi.

disable {**all | printer**}

bỏ qua các công việc in ấn mới.

down {all | printer}

vô hiệu hoá toàn bộ công việc in ấn trên máy in.

enable {all | printer}

cho phép công việc chuyển tới máy in.

exit

thoát khỏi **lpc**.

quit

thoát khỏi **lpc** (giống lệnh **exit**).

restart {all | printer}

khởi động lại **lpd** cho máy in

status {all | printer}

hiển thị trạng thái máy in.

up {all | printer}

làm cho có hiệu lực công việc in ấn và khởi động **lpd** mới.

stop {all | printer}

lệnh này để ngừng máy in. Mặc dù các yêu cầu in có thể lưu tạm nhưng nó vẫn không được in cho đến khi máy in khởi động. Nếu tài liệu đang in trong khi lệnh **stop** thực hiện thì tài liệu in sẽ được in hoàn chỉnh rồi mới ngừng lại.

5.5 Sound card

Hầu hết các sound card chính đều được Linux hỗ trợ. Thường thì phần hỗ trợ được cài sẵn trong nhân. Linux cho phép dùng lệnh **cat** để làm việc với các dạng tập tin **audio**.

Dạng tập tin đặc biệt của nó là: **/dev/audio**, **/dev/mide**, **/dev/dsp** (**Digital signal processor**).

Ví dụ:

```
# cat /dev/audio > file.au
```

sẽ nhập dữ liệu từ sound card theo hạng thức **.au** vào tập tin đã chỉ định. Để phát lại dữ liệu có thể dùng:

```
# cat file.au >/dev/audio
```

CHƯƠNG 6. 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** do **vim** chính là bộ soạn thảo tương thích với **vi**. **vim** được phần lớn người dùng sử dụng để soạn thảo các tập tin 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 trong 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 ***i** là cách viết tắt một danh sách các lệnh được giải thích ở sau):

Chuyển Từ sang chế chế độ độ	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**.

6.1 Khởi động vim

6.1.1 Mở chương trình soạn thảo vim

Cách đơn giản nhất bắt đầu dùng **vim** để soạn thảo một tập tin văn bản, là gõ một trong ba lệnh sau:

vim [**tuỳ-chọn**]	bắt đầu soạn thảo hay hiệu chỉnh một tập tin
vim [**tuỳ-chọn**] <danh sách các tập-tin>	bắt đầu soạn thảo một hoặc nhiều tập tin
vim [tuỳ chọn] **-**	soạn thảo một tập tin từ thiết bị vào chuẩn

Nếu tham số danh sách các tập tin không có thì **vim** sẽ thao tác với một tập tin mới (vùng đệm soạn thảo rỗng). Ngược lại, tập tin đầu tiên trong danh sách trở thành tập tin 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 tập tin kế tiếp, ta đánh lệnh "**:next**" ở chế độ lệnh. Để soạn thảo một tập tin có tên bắt đầu bằng "**-**" thì phải điền vào tên tập tin dấu "**--**".

Ví dụ:

```
        # vim vdvim
~           ▬
~
~
~
~
~
~
"vdvim"[New File] 0,0-1 All
```

Lệnh trên mở một cửa sổ cho người dùng soạn thảo một tập tin mới có tên là **"vdvim"**

Một số các tuỳ chọn cơ bản:

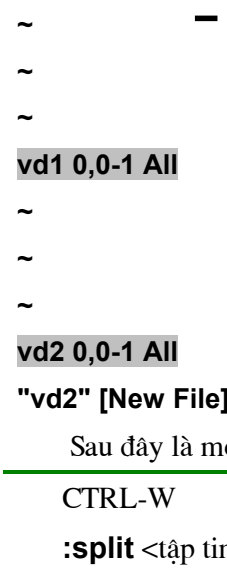
+ [n]	đặt dấu nhắc trỏ tại dòng thứ n (ngầm định là dòng cuối)
+ <lệnh>	thực hiện lệnh sau khi nạp tập tin
+/<mẫu> <tập tin>	đặt dấu nhắc trỏ tại dòng đầu tiên có chứa mẫu trong tập tin
-o[n]	mở n cửa sổ (ngầm định có một cửa sổ cho một tập tin: n=1)
--help	hiển thị danh sách các tham số và thoát

6.1.2. 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 tập tin khác nhau.

Ví dụ lệnh sau sẽ mở hai tập tin **vd1** và **vd2** trên hai cửa sổ soạn thảo:

```
# vim -o2 vd1 vd2
```



Sau đây là một số các lệnh hay dùng:

CTRL-W	chia cửa sổ hiện tại thành hai phần
:split <tập tin>	chia cửa sổ và soạn thảo <tập tin> trên một phần chia của cửa sổ
:sf <tập tin>	chia cửa sổ, tìm tập tin 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 **_**	thiết đặt chiều cao của cửa sổ hiện thời

6.1.3. Ghi và thoát trong vim

Bảng dưới đây giới thiệu các lệnh để ghi nội dung tập tin lên hệ thống tập tin và thoát khỏi **vim** sau khi đã soạn thảo xong nội dung của tập tin (tham số [n, m] nếu có mang ý nghĩa "*từ dòng n tới dòng m")*.

:[n,m] w [!]	ghi tập tin hiện thời.
:[n,m] w <tập tin>	ghi nội dung ra <tập tin>, trừ khi tập tin đó đã thực sự tồn tại
:[n,m] w! <tập tin>	ghi nội dung ra <tập tin>, nếu tập tin đã tồn tại thì ghi đè lên nội dung cũ
:[n,m] w[!] >> [<tập tin>]	chèn thêm vào <tập tin>, nếu không có tập tin, mặc định là tập tin 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 tập tin hiện thời nếu nó được sửa đổi
:q [!]	thoát khỏi **vim**
:wq [!] [<tập tin>]	ghi nội dung <tập tin> (mặc định là tập tin hiện thời) và thoát khỏi **vim**
:x [!] <tập tin>	giống :wq nhưng chỉ ghi khi thực sự có sự thay đổi trong nội dung tập tin (giống ZZ)
:st [!]	dừng **vim** và khởi tạo một shell (giống CTRL-Z)

6.2. Di chuyển trỏ soạn thảo trong Vim

6.2.1. 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:

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 tập tin
n	G	di chuyển đến dòng thứ n trong tập tin (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

6.2.2. 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

6.2.3. 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)

6.3. Các thao tác trong văn bản

vim có rất nhiều các lệnh hỗ trợ thao tác soạn thảo hay hiệu chỉnh một tập tin. 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.

6.3.1. 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

6.3.2. 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 tập tin
	d1G	xoá ngược từ vị trí hiện thời đến đầu tập tin
	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)

6.3.3. 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

6.3.5. 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:

** Sao chép văn bản vào bô nhớ đệm*

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

** Dán văn bản:*

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 tập tin

6.3.6. 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 tập tin
? <xâu>	tìm xâu từ dòng hiện tại ngược lên dòng đầu trong tập tin
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 tập tin đang soạn thảo các từ **the** và thay chúng bởi các từ **The**.

6.3.7. 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 tập tin hiện thời
'{A-Z}	dịch chuyển con trỏ tới vị trí đã được đánh dấu bởi các chữ cái {A-Z} trong một tập tin bất kỳ
:marks	hiển thị các đánh dấu hiện thời

6.3.8. Các phím sử dung trong chế đô 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-@	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

6.3.9. 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

6.3.10. 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

6.4. Các lệnh khác

6.4.1. 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

6.4.2. Các lệnh liên quan đến tập tin

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 tập tin.

CTRL-G	hiển thị tên tập tin hiện thời kèm theo trạng thái tập tin và vị trí dấu nhắc trỏ (trạng thái có thể là: chỉ đọc, được sửa, lỗi khi đọc, tập tin mới) (giống **:f**)

n	CTRL-G	hiển thị thông tin như CTRL -G và có thêm đường dẫn đầy đủ của tập tin (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 tập tin hiện thời thành tên mới
	:ls	liệt kê tất cả các tập tin hiện thời đang được sử dụng trong Vim (giống **:buffer** và **:files**)
	:cd	đưa thêm đường dẫn vào tên tập tin
	:w <tên tập tin>	tạo một bản sao của tập tin hiện thời với tên mới là tên tập tin (giống như save as trong Win)
		Xác định tập tin cần soạn thảo
	:e[n, /mẫu] <tập tin>	soạn thảo tập tin, từ dòng thứ n hoặc từ dòng có chứa mẫu, trừ khi có sự thay đổi thực sự trong tập tin
	:e[n, /mẫu]! <tập tin>	luôn soạn thảo tập tin, từ dòng thứ n hoặc từ dòng có chứa mẫu, bỏ qua mọi sự thay đổi trong tập tin
	:e	nạp lại tập tin hiện thời, trừ khi có sự thay đổi thực sự trong tập tin
	:e!	luôn nạp lại tập tin hiện thời, bỏ qua mọi sự thay đổi thực sự trong tập tin
	:fin [!] <tập tin>	tìm tập tin trên đường dẫn và soạn thảo
	:e #n	soạn thảo tập tin 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

CHƯƠNG 7. LỆNH ĐỐI VỚI TIẾN TRÌNH

7.1. Khái niệm

Khi mở một trang **man**, liệt kê các tập tin 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 tiến trình. Trong Linux, bất cứ chương trình nào đang chạy đều được coi là một tiến trình. Có thể có nhiều tiến trình cùng chạy một lúc. Ví dụ dòng lệnh **ls -l | sort | more** sẽ khởi tạo ba tiến trình: **ls**, **sort** và **more**.

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

Ký hiệu	Ý nghĩa
D	(**uninterruptible sleep**) ở trạng thái này tiến trình bị treo và không thể chạy lại nó bằng một tín hiệu.
R	(**runnable**) trạng thái sẵn sàng thực hiện, tức là tiến trình có thể thực hiện được nhưng chờ đến lượt thực hiện vì một tiến trình khác đang có CPU.
S	(**sleeping**) trạng thái tạm dừng, tức là tiến trình tạm dừng không hoạt động (20 giây hoặc ít hơn)
T	(**traced or stopped**) trạng thái dừng, tiến trình có thể bị treo bởi một tiến trình ngoài
Z	(**zombie process**) tiến trình đã kết thúc thực hiện, nhưng nó vẫn được tham chiếu trong hệ thống
W	không có các trang thường trú
<	tiến trình có mức ưu tiên cao hơn
N	tiến trình có mức ưu tiên thấp hơn
L	có các trang khóa bên trong bộ nhớ

7.2. Các lệnh cơ bản

7.2.1. Lệnh fg *và lệnh* bg

Linux cho phép người dùng sử dụng tổ hợp phím **CTRL+z** để dừng một tiến trình và khởi động lại tiến 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 tiến 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 tiến 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 tiến 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 tiến trình nền kết thúc thì hệ thống sẽ gửi thông báo tới người dùng, nhưng trên hầu hết các hệ thống, khi tiến 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 tiến trình (thường thì một tiến 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 tập tin **thu1** trong chế độ nền:

```
# find -name thu1 &
```

[5] 918

trong chế độ này, số thứ tự của chương trình là [**5**], chỉ số tiến trình tương ứng với lệnh **find** là **918**. Vì gõ Enter khi tiến trình chưa thực hiện xong nên trên màn hình chỉ hiển thị số thứ tự của chương trình và chỉ số tiến 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 tập tin **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.

7.2.2. Tìm ra các tiến 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 tiến 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 tiến 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 tiến trình, TTY - tên thiết bị đầu cuối trên đó tiến trình được thực hiện, TIME - thời gian để chạy tiến trình, CMD - lệnh khởi tạo tiến 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 tiến trình.

-T

chọn để hiển thị các tiến trình trên trạm cuối đang chạy.

-a

chọn để hiển thị tất cả các tiến trình trên một trạm cuối, bao gồm cả các tiến trình của những người dùng khác.

-r

chỉ hiển thị tiến trình đang được chạy.

- ***Chọn theo danh sách***

-C

chọn hiển thị các tiến trình theo tên lệnh.

-G

hiển thị các tiến trình theo chỉ số nhóm người dùng.

-U

hiển thị các tiến trình theo tên hoặc chỉ số của người dùng thực sự (người dùng khởi động tiến trình).

-p

hiển thị các tiến trình theo chỉ số của tiến trình.

-s

hiển thị các tiến trình thuộc về một phiên làm việc.

-t

hiển thị các tiến trình thuộc một trạm cuối.

-u

hiển thị các tiến trình theo tên và chỉ số của người dùng hiệu quả.

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

-f

hiển thị thông tin về tiến trình với các trường sau UID - chỉ số người dùng, PID - chỉ số tiến trình, PPID - chỉ số tiến trình khởi tạo ra tiến trình, C - , STIME - thời gian khởi tạo tiến trình, TTY - tên thiết bị đầu cuối trên đó tiến trình được chạy, TIME - thời gian để thực hiện tiến trình, CMD - lệnh khởi tạo tiến trình

-l

hiển thị đầy đủ các thông tin về tiến 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 tiến 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 tiến trình
%mem	% bộ nhớ được sử dụng để chạy tiến trình
%G	tên nhóm người dùng
%P	chỉ số của tiến trình cha khởi động ra tiến trình con
%U	định danh người dùng
%c	lệnh tạo ra tiến trình
%p	chỉ số của tiến trình
%x	thời gian để chạy tiến trình
%y	thiết bị đầu cuối trên đó tiến trình được thực hiện

Ví dụ, muốn xem các thông tin như tên người dùng, tên nhóm, chỉ số tiến trình, chỉ số tiến trình khởi tạo ra tiến trình, tên thiết bị đầu cuối, thời gian chạy tiến trình, lệnh khởi tạo tiến 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
```

7.2.3. Hủy tiến trình với lệnh **kill**

Trong một số trường hợp, sử dụng lệnh **kill** để hủy bỏ một tiến trình. Điều quan trọng nhất khi sử dụng lệnh **kill** là phải xác định được chỉ số của tiến 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 tiến 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 tiến 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 tiến trình ngay lập tức
15	SIGTERM	tín hiệu này yêu cầu dừng tiến trình ngay lập tức, nhưng cho phép chương trình xóa các tập tin tạm.

-p

lệnh **kill** sẽ chỉ đưa ra chỉ số của tiến trình mà không gửi một tín hiệu nào.

-l

hiển thị danh sách các tín hiệu mà lệnh **kill** có thể gửi đến các tiến trình (các tín hiệu này có trong tập tin **/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

7.2.4. 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 [tuỳ-chọn]... NUMBER[SUFFIX]
```

- 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

7.2.5. Xem cây tiến trình với lệnh pstree

Đã biết lệnh để xem các tiến 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 tiến trình, đó là lệnh **pstree**.

Cú pháp lệnh:

```
pstree [tùy-chọn] [pid | người-dùng]
```

Lệnh **pstree** sẽ hiển thị các tiến trình đang chạy dưới dạng cây tiến trình. Gốc của cây tiến trình thường là init. Nếu đưa ra tên của một người dùng thì cây của các tiến trình do người dùng đó sở hữu sẽ được đưa ra.

pstree thường gộp các nhánh tiến 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 tiến 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 tiến trình hiện thời và "tổ tiên" của nó với màu sáng trắng

-H

giống như tùy chọn -h, nhưng tiến trình con của tiến trình hiện thời không có màu sáng trắng

-l

hiển thị dòng dài.

-n

sắp xếp các tiến trình cùng một tổ tiên theo chỉ số tiến 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

7.2.6. Lênh thiết đăt lai đô ưu tiên của tiến trình nice và lênh renice

Ngoài các lệnh xem và hủy bỏ tiến trình, trong Linux còn có hai lệnh liên quan đến độ ưu tiên của tiến 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 tiến 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 tiến trình đang chạy.

-g

thay đổi quyền ưu tiên theo nhóm người dùng

-p

thay đổi quyền ưu tiên theo chỉ số của tiến 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 tiến trình có chỉ số là **987** và **32**, và tất cả các tiến trình do người dùng **daemon** và **root** sở hữu.

CHƯƠNG 8. MIDNIGHT COMMANDER

8.1. Giới thiệu về Midnight Commander (MC)

Nếu đã là một người sử dụng máy tính lâu năm, chắc chắn các đã từng dùng hệ điều hành MS-DOS và chắc chắn đều biết đến và sử dụng thành thạo một tiện ích rất mạnh trong việc quản lý, điều khiển các thao tác về tập tin, thư mục, đĩa, một môi trường trực quan trong chế độ văn bản (text). Có lẽ không cần nói thêm mọi người cũng biết đó là tiện ích Norton Commander (NC). Trong các hệ điều hành Windows sau này, mặc dù đã có sự hỗ trợ của các 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 tập tin và thư mục hơn. Trong Linux có một tiện ích mang tên Midnight Commander (viết tắt là MC) cũng có những chức năng và giao diện gần giống với NC của DOS. Và nếu đã từng sử dụng NC thì việc sử dụng MC trong Linux chẳng có gì là khó khăn cả.

8.2. Khởi động MC

Cấu trúc lệnh khởi động MC như sau:

```
# mc [Tùy-chọn]
```

Có một số tuỳ chọn khi dùng tiện ích này. Tham số này bao gồm 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 tập tin 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.

8.3. 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 tập tin 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 tập tin 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

tập tin và thư mục con của thư mục hiện hành. Midnight Commander có cơ chế hiển thị các kiểu tập tin khác nhau bằng các ký hiệu và màu sắc khác nhau, ví dụ như các tập tin biểu tượng liên kết sẽ có ký hiệu '@' ở đầu, các tập tin thiết bị sẽ có màu đỏ tím, các tập tin 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.

8.4. Dùng chuột trong MC

Midnight Commander sẽ hỗ trợ chuột trong trường hợp không gọi với tham số --d. Khi kích chuột vào một tập tin trên Panel, tập tin đó sẽ được chọn, có nghĩa là thanh sáng chọn sẽ nằm tại vị trí tập tin đó và panel chứa tập tin đó sẽ trở thành panel hiện hành. Còn nếu kích chuột phải vào một tập tin, tập tin đó sẽ được đánh dấu hoặc xoá dấu tuỳ thuộc vào trạng thái kích trước đó.

Nếu kích đôi chuột tại một tập tin, tập tin đó sẽ được thi hành nếu đó là tập tin 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>

8.5. 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, Midnight Commander ký hiệu 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. có thể tham khảo thêm trong phần sau của cuốn sách.

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 tập tin hay thư mục mà hoặc việc chuyển đổi thư mục hoặc thi hành tập tin 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 tập tin ở vị trí thanh sáng chọn xuống dòng lệnh
M-Tab	hoàn thành tên tập tin, lệnh, các biến, tên người dùng hoặc tên máy giúp
C-x t, C-x C-t	sao các tập tin được đánh dấu (mặc định là tập tin 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 tập tin hoặc thư mục

M-g, M-h, M-j	lần lượt chọn tập tin đầu tiên, tập tin giữa và tập tin cuối trên panel hiển thị
C-s, M-s	tìm kiếm tập tin 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. Khi gõ các kí tự, thanh sáng chọn sẽ di chuyển đến tập tin 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 tập tin hoặc thư mục
C-\	thay đổi thư mục hiện thời
+	sử dụng dấu cộng để lựa chọn đánh dấu một nhóm tập tin. Có thể sử dụng các ký tự đại diện như '*', '?'... để biểu diễn các tập tin sẽ chọn
\	sử dụng dấu trừ để xoá đánh dấu một nhóm tập tin. Có thể sử dụng các ký tự dại diện như '*', '?' để biểu diễn các tập tin sẽ xoá
*	sử dụng dấu * để đánh dấu hoặc xoá đánh dấu tất cả các tập tin 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

8.6. 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 tập tin. Có bốn kiểu hiển thị: * Full - hiển thị thông tin về tên , kích thước, và thời gian sử đổi của tập tin; * Brief - chỉ hiển thị tên của tập tin; * Long - hiển thị thông tin đầy đủ về tập tin (tương tự lệnh ls -l); * User - hiển thị các thông tin do tự chọn về tập tin;
Quick view	C-x q	xem nhanh nội dung của một tập tin
Info	C-	xem các thông tin về một thư mục hoặc tập tin

x i

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 tập tin theo tên
Network link ...		thực hiện liên kết đến một máy tính
FTP link ...		thực hiện việc lấy các tập tin 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 tập tin đã được đánh dấu hoặc tập tin 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 tập tin hiện thời
View file ...		mở và xem nội dung của một tập tin bất kì
Filtered view	M-!	thực hiện một lệnh lọc với tham số là tên tập tin và hiển thị nội dung của tập tin đó
Edit	F4	soạn thảo tập tin 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 tập tin
Link	C-x l	tạo một liên kết cứng đến tập tin hiện thời
Symlink	C-x s	tạo một liên kết tượng trưng đến tập tin 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 tập tin
Advanced chown		thay đổi quyền sở hữu cũng như quyền truy nhập của tập tin hay thư mục
Rename/Move	F6	thực hiện việc đổi tên hay di chuyển đối với một tập tin
Mkdir	F7	tạo một thư mục
Delete	F8	xoá một hoặc nhiều tập tin
Quick cd	M-c	chuyển nhanh đến một thư mục
select Group	M-+	thực hiện việc chọn một nhóm các tập tin
Unselect group	M-\	ngược với lệnh trên
reverse selecTion	M-*	chọn các tập tin 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 tập tin
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 tập tin 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 tiến trình nền
Extension file edit		cho phép hiệu chỉnh tập tin ~/.mc/ext để xác định chương trình sẽ thực hiện khi xem, soạn thảo hay làm bất cứ điều gì trên các tập tin 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 tập tin ảo
Save setup	ghi mọi sự thiết lập được thay đổi

8.7. 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 tập tin
F4	soạn thảo nội dung một tập tin
F5	thực hiện sao chép tập tin
F6	thực hiện di chuyển hoặc đổi tên tập tin
F7	tạo thư mục mới
F8	xoá thư mục hoặc tập tin
F9	đưa trỏ soạn thảo lên thanh thực đơn nằm ngang
F10	thoát khỏi MC

8.8. 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 tập tin văn bản, hãy di chuyển thanh sáng chọn đến vị trí tập tin đó rồi nhấn F4, nội dung của tập tin 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 tập tin

Open/load	C-o	mở hoặc nạp một tập tin
New	C-n	tạo một tập tin mới
Save	F2	ghi nội dung tập tin đã được soạn thảo
Save as ...	F12	tạo một tập tin khác tên nhưng có nội dung trùng với nội dung tập tin hiện thời
Insert file ...	F15	chèn nội dung một tập tin vào tập tin hiện thời
Copy to file ...	C-f	sao đoạn văn bản được đánh dấu đến một tập tin 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 tập tin

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 tập tin
Move	F6	thực hiện di chuyển tập tin
Delete	F8	xoá tập tin
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 tập tin
F3	thực hiện việc đánh dấu đạon văn bản
F4	tìm và thay thế xâu văn bản
F5	thực hiện việc sao chép
F6	di chuyển tập tin
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

CHƯƠNG 9. MTOOLS - TIỆN ÍCH TRUY CẬP Ổ ĐĨA DOS TRONG LINUX

9.1 Phần giới thiệu

Mtools là một nhóm các công cụ chung cho phép các hệ thống UNIX nói chung (và Linux nói riêng) thao tác với các tập tin MS-DOS như: đọc, ghi, di chuyển các tập tin trong một hệ thống tập tin MS-DOS (chẳng hạn ổ đĩa mềm). Trong Mtools, mỗi chương trình cố gắng mô phỏng các lệnh tương đương trong MS-DOS.

Mtools rất hiệu quả trong việc truy cập vào hệ thống tập tin MS-DOS.

Với **Mtools**, một người có thể dễ dàng truy cập vào ổ đĩa mềm với chỉ một lệnh mà không cần **mount** và **unmount** lại.

9.2 Các thuộc tính chung của các lệnh mtools

9.2.1 Các tuỳ chon và tên các tập tin

Đối với MSTOOLS, để sử dụng một tập tin trong Linux, cần xác định rõ vị trí của tập tin đó và tên của nó, một cách tông quát có thể viết như sau:

[<ổ-đĩa:>] [/<tên-thư-mục>/... /] <tên-tập-tin>

tên ổ đĩa kèm dấu hai chấm (**:**), tên thư mục (đầy đủ) chứa tập tin và tên tập tin.

Chỉ phần tên tập tin là bắt buộc, tên ổ đĩa và tên thư mục (đường dẫn) là có thể có hoặc không. Ngầm định các tập tin không có tên ổ đĩa đi kèm là các tập tin Linux. Các thư mục trong tên tập tin được phân cách bởi dấu '/' hoặc dấu '\'. Nếu sử dụng dấu '\' hoặc các ký tự đại diện trong tên tập tin MS-DOS, nên bao chúng bởi dấu nháy kép ("") để đảm bảo tính đúng đắn của lệnh.

Mọi tuỳ chọn đều được bắt đầu bởi dấu trừ (-), chứ không phải ký tự '/' như trong MS-DOS.

Hầu hết các lệnh **mtools** đều cho phép tác động lên nhiều tập tin cùng lúc hoặc sử dụng nhiều tuỳ chọn lệnh khác nhau, không tuân theo các quy tắc trong MS-DOS, nhưng điều này lại giúp ích cho người dùng hơn.

Tên ổ đĩa

Cách đặt tên ổ đĩa phụ thuộc vào từng cấu trúc máy, tuy nhiên hầu hết thường qui định ổ đĩa A là ổ đĩa mềm đầu tiên, ổ đĩa B là ổ đĩa mềm thứ hai (nếu có), ổ đĩa J (nếu có) là một ổ đĩa Jaz và Z là ổ đĩa Zip. Trên các hệ thống này, các tên thiết bị được bắt nguồn từ SCSI id, ổ đĩa Jaz được giả định đến SCSI đích 4, và Zip ở SCSI đích 5 (sắp đặt ngầm định của nhà sản xuất). Trên LINUX, cả hai ổ đĩa này được đặt trên ổ đĩa thứ hai trên SCSI bus (/dev/sdb). Sự thiết lập mặc định này có thể được thay đổi bằng một tập tin cấu hình (configuration file).

Thư muc làm viêc hiên thời

Lệnh **mcd** được sử dụng để khởi tạo một thiết bị và thư mục làm việc hiện thời (liên quan đến hệ thống tập tin MS-DOS), ngầm định được đặt là A:/, không như MS-DOS, ở đây chỉ có một thư mục làm việc cho mọi ổ đĩa chứ không phải cho mỗi ổ đĩa.

Tên tập tin dài kiểu VFAT

Phiên bản này của **mtools** hỗ trợ các tên tập tin dài VFAT. Nếu tên một tập tin Linux dài quá qui định đặt tên trong MS-DOS, nó sẽ được lưu thành một tên VFAT dài, và một tên ngắn đi kèm sẽ được tạo ra. Tên ngắn này là cái nhìn thấy khi kiểm tra

ổ đĩa bằng một phiên bản của DOS trước version 7.0. Bảng dưới đây chỉ ra vài ví dụ của tên tập tin ngắn:

Tên dài	Tên MS-DOS	Lý do thay đổi
thisisatest	THISIS~1	Tên tập tin quá dài
alain.knaff	ALAIN~1.KNA	Tên mở rộng quá dài
prn.txt	PRN~1.TXT	PRN là tên thiết bị
.abc	ABC~1	Tên chính của tập tin không có mà chỉ có tên mở rộng
hot+cold	HOT_CO~1	Trong tên tập tin có chứa ký tự đặc biệt

Nói chung, có một số các qui tắc sau để khởi tạo một tên tập tin ngắn:

- ❖ Các ký tự đặc biệt được thay thế bởi dấu gạch dưới. Các ký tự đó là ; + = [ ] ' , \ " * \\ < > / ? : |
- ❖ Các dấu chấm phân cách giữa tên chính và tên mở rộng sẽ được bỏ qua
- ❖ Một số '~'n được tạo ra ở cuối tên tập tin (ví dụ: HOT_CO~1)
- ❖ Tên tập tin được rút ngắn để nằm trong giới hạn 8 ký tự tên chính và 3 ký tự tên mở rộng.

Tên tập tin kiểu Linux (cả dài và ngắn) được gọi là tên chính (**primary name**), và tên được rút ngắn theo dạng MS-DOS là tên phụ (**secondary name**).

Ví dụ:

```
# mcopy /etc/motd a:Reallylongname
```

Mtools sẽ tự động tạo ra một tên ngắn là REALLY~1 thay cho tên dài là Reallylongname. Reallylongname là primary name, và REALLY~1 là secondary name.

```
# mcopy /etc/motd a:motd
```

Trong lệnh trên, tên tập tin là phù hợp với giới hạn tên tập tin của DOS, do đó, **mtools** không cần tạo ra tên khác, motd là tên chính, và không cần tên phụ.

Xung đột tên tập tin

Khi ghi một tập tin vào đĩa, rất có thể tên tập tin này (tên ngắn hoặc tên dài) trùng với tên một tập tin hoặc một thư mục đã tồn tại. Điều này có thể xảy ra với tất cả các lệnh tạo ra một thư mục mới, chẳng hạn **mcopy, mmd, mren, mmove, mwrite** và **mread**. Khi một xung đột tên tập tin xảy ra, **mtools** hỏi nên làm thế nào, có vài lựa chọn sau:

overwrite

Ghi đè một tập tin đang tồn tại. Không thể ghi đè một tên tập tin lên một thư mục.

rename

Đổi tên tập tin mới tạo ra. **Mtools** nhắc đưa vào tên mới

autorename

Mtools tự động đổi tên tập tin mà không hỏi

skip

Bỏ tập tin này, và chuyển đến tập tin tiếp theo (nếu có)

Để chọn một trong những lựa chọn này, đánh chữ đầu tiên của các lựa chọn trên tại dấu nhắc. Nếu sử dụng chữ thường, thao tác chỉ áp dụng cho tập tin đó, nếu dùng chữ hoa, thao tác sẽ áp dụng cho mọi tập tin, sẽ không bị hỏi lại nữa.

Đồng thời có thể chọn các thao tác tác dụng lên tên tập tin (cho mọi tập tin) trên dòng lệnh khi gọi **mtools**:

-o Ngầm định ghi đè tên tập tin chính.

-O Ngầm định ghi đè tên tập tin phụ.

-r Ngầm định đổi tên tập tin chính.

-R Ngầm định đổi tên tập tin phụ.

-a Ngầm định tự động đổi tên tập tin chính.

-A Ngầm định tự động đổi tên tập tin phụ.

-s Ngầm định bỏ qua tên tập tin chính.

-S Ngầm định bỏ qua tên tập tin phụ.

-m Hỏi người sử dụng làm gì với tên tập tin chính.

-M Hỏi người sử dụng làm gì với tên tập tin phụ.

Lưu ý rằng các tuỳ chọn liên quan đến tên tập tin của lệnh có sự tương ứng giữa chữ hoa/chữ thường cho tên chính/tên phụ, ngược lại với các lựa chọn tương tác chữ thường/chữ hoa giữa một thời điểm/thường xuyên.

Ngầm định, người sử dụng sẽ được nhắc nếu xảy ra xung đột tên chính, và tên phụ sẽ được tự động đổi tên.

Nếu xung đột tên xảy ra trong một thư mục Linux, **mtools** chỉ hỏi khi ghi đè tập tin hoặc bỏ qua tập tin đó.

Phân biệt chữ hoa, chữ thường trong hệ thống tập tin VFAT

Hệ thống tập tin VFAT cho phép tên tập tin có thể bao gồm cả chữ hoa hay chữ thường. Tuy nhiên trong cùng một thư mục, không thể tồn tại hai tập tin cùng có tên với nội dung như nhau chỉ khác chữ hoa chữ thường.

Ví dụ nếu lưu một tập tin có tên LongFileName trên một hệ thống tập tin VFAT, **mdir** hiển thị tập tin này là LongFileName. Và khi đó, nếu cố thêm tập tin LONGFILENAME vào cùng thư mục, nó sẽ bị từ chối, vì kiểm tra xung đột bỏ qua phân biệt chữ hoa/chữ thường.

```
# mcd a:/
# mmd LongFileName
# mdir
CRACKVK    EXE    110592 11-27-2000  0:49 CrackVK.exe
LONGFI~1   <DIR>          08-14-2000 14:32 LongFileName
```

```
        2 files        110 592  bytes
                     1 346 560  bytes free
# mmd LONGFILENAME
Long file name "LONGFILENAME" already exists.
a)utorename A)utorename-all r)ename R)ename-all
s)kip S)kip-all q)uit (aArRsSq): _
```

Hệ thống tập tin VFAT cho phép lưu kiểu chữ của tên tập tin trong byte thuộc tính, nếu mọi ký tự của tên tập tin có cùng kiểu chữ, và mọi ký tự của tên mở rộng cũng cùng kiểu chữ. **Mtools** sử dụng thông tin này khi hiển thị tập tin, và đồng thời tạo ra tên tập tin Linux khi **mcopying** đến một thư mục Linux. Có thể kết quả sẽ không như mong muốn nếu áp dụng cho tập tin được viết bởi các phiên bản 7.0 trở về trước của DOS vì: các tên tập tin MS-DOS kiểu cũ được chuyển thành chữ hoa, trong khi các phiên bản cũ của **mtools** lại sử dụng các tên tệp Linux chữ thường.

Đinh dang dung lương lớn

Mtools hỗ trợ một số loại định dạng cho phép lưu trữ nhiều dữ liệu trên đĩa hơn thông thường. Các định dạng này có thể không được hỗ trợ trên tất cả các hệ điều hành vì các khả năng khác nhau của chúng.

Để định dạng các ổ đĩa, cần phải sử dụng một công cụ có sẵn trên hệ điều hành đang sử dụng. Đối với Linux, các công cụ dành cho ổ mềm có thể thấy trong gói tin **fdutils** nằm trong các địa chỉ sau:

ftp://www.tux.org/pub/knaff/fdutils/.

ftp://sunsite.unc.edu/pub/Linux/utils/disk-management/fdutils-*

Nhiều sector hơn

Phương pháp cũ nhất để lưu trữ nhiều dữ liệu hơn trên một đĩa là sử dụng nhiều sector và cylinder hơn. Mặc dù định dạng chuẩn sử dụng 80 cylinder và 18sector (trên một đĩa mật độ cao 3 1/2 inch), nhưng có thể tăng lên thành 83 cylinder (với hầu hết các đĩa) và 21 sector. Phương pháp này cho phép lưu trữ thêm 1743K trên một đĩa mật độ cao 3 1/2 inch. Tuy nhiên, các đĩa có 21 sector đọc chậm gấp đôi so với đĩa chuẩn 18 sector bởi vì các sector được nén lại sát nhau hơn mà chúng ta lại cần phải xen vào giữa chúng để đọc. Vấn đề này không gặp phải với định dạng ổ đĩa 20 sector.

Các định dạng này được hỗ trợ bởi rất nhiều các tiện ích phần mềm của DOS chẳng hạn **fdformat** và **vgacopy**. Mtools hỗ trợ các định dạng này trên Linux, trên Sun và DELL Unix PC.

Sectors lớn hơn

Bằng việc sử dụng các sector lớn hơn, có thể vượt ra khỏi dung lượng có được của các sector chuẩn 512-byte. Đó là vì các đầu đọc (header) sector. Các đầu đọc sector có cùng cỡ, bất kể có bao nhiêu byte bên trong sector. Do đó, sẽ tiết kiệm thêm khoảng trống bằng cách sử dụng một số ít hơn các sector có dung lượng lớn hơn. Ví dụ, 1 sector có dung lượng 4K chỉ cần một đầu đọc là có thể đọc dữ liệu một cách thông suốt, trong khi 8 sector với dung lượng 512 byte cần tới những 8 đầu đọc để có thể đọc thông suốt cùng một lượng dữ liệu. Phương pháp này cho phép lưu trữ lên tới 1992K trên một đĩa 3 1/2 inch. **Mtools** chỉ hỗ trợ các định dạng này trên Linux.

Đinh dang 2m

Định dạng 2m được phát minh bởi Ciriaco Garcia de Celis. Định dạng này cũng sử dụng các sector lớn hơn thông thường để chứa được nhiều dữ liệu trên đĩa. Tuy nhiên, định dạng này vẫn sử dụng định dạng chuẩn (18 sector, 1 sector = 512 byte) trên cylinder đầu tiên để DOS dễ điều khiển các đĩa hơn. Thực sự phương pháp này cho phép có một bootsector dạng chuẩn, dùng để chứa các mô tả cách đọc những phần còn lại như thế nào.

Dù vậy, một trở ngại của định dạng này là cylinder đầu tiên có thể chứa ít dữ liệu hơn các cylinder khác. Thật không may, DOS chỉ có thể đọc được các đĩa mà mỗi track chứa cùng một lượng dữ liệu. Vì vậy, định dạng 2m phải dấu track đầu tiên chứa ít dữ liệu dơn bằng cách sử dụng một shadow FAT.(Thông thường DOS lưu bảng FAT thành 2 bản để dự phòng. XDF chỉ lưu một bảng, và nó báo với DOS là nó lưu 2 bản. Vì vậy, phần tương tự bản FAT thứ hai cũng được ghi lại). Nó đồng nghiã rằng không bao giờ nên sử dụng một ổ đĩa định dạng 2m để lưu những thứ không phải là DOS fs.

Mtools chỉ hỗ trợ định dạng này trên Linux.

Đinh dang XDF

XDF là một định dạng dung lượng lớn được OS/2 sử dụng. Định dạng này có thể chứa 1840 K trên một đĩa mềm, tức là vẫn nhỏ hơn các định dạng 2m tốt nhất. Nhưng ưu điểm chính của nó tốc độ đọc nhanh: 600 miligiây trên một track, nhanh hơn định dạng 21 sector, và hầu như tương đương với định dạng chuẩn 18 sector. Để truy cập vào các đĩa định dạng này, phải chắc chắn rằng mtools đã được biên dịch với hỗ trợ XDF, và đặt biến *usedf* cho ổ đĩa trong tập tin cấu hình (configuration).Tham khảo thêm về *Compiling mtools*, và *misc* về cách thực hiện. Truy cập Fast XDF chỉ là biến của các nhân Linux từ 1.1.34 trở đi.

Mtools chỉ hỗ trợ định dạng này trên Linux.

Cẩn thận: Chú ý các nhà phân phối: Nếu mtools đ* được biên dịch trên nhân Linux sau phiên bản 1.3.34, nó sẽ không thể chạy trên một nhân Linux trước đó. Dù vậy, nếu mtools đã được biên dịch trên một nhân trước đó, nó vẫn có thể chạy trên một nhân mới hơn, trừ khi truy cập XDF chậm hơn. Có đề nghị là các nhà phân phối sản phẩm chỉ bao gồm mtools binaries được biên dịch trên nhân trước phiên bản 1.3.34 cho đến khi phiên bản 2.0 được phát hành. Khi bản 2.0 được phát hành, mtools binaries được biên dịch trên một nhân mới có thể (và nên) được phát hành. Mtools binaries được biên dịch trên nhân trước bản 1.3.34 sẽ không chạy được trên phiên bản từ 2.1 trở đi.

Mã thoát ra

Mọi lệnh Mtools trả lại giá trị 0 khi thực hiện thành công, giá trị 1 nếu hỏng, và 2 khi hỏng một phần. Mọi lệnh Mtools thực hiện một vài kiểm tra thông minh trước khi tiếp tục thực hiện, để xác định ổ đĩa thực sự là một đĩa MS-DOS (chứ không phải là ổ đĩa ext2 hay minix). Các kiểm tra này có thể loại bỏ cục bộ các đĩa hỏng mà có thể vẫn đọc được bằng cách khác. Để tránh các kiểm tra này, đặt biến môi trường MTOOLS_SKIP_CHECK hoặc biến tập tin thiết lập tương ứng (corresponding configuration file variable).

Vướng mắc

Một ảnh hưởng phụ là việc không ước lượng thiết bị hợp lệ (khi các dung lượng đĩa đa dạng được hỗ trợ) là thông báo lỗi thường gặp của driver thiết bị. Những thông báo này có thể bỏ qua.

Mã kiểm tra FAT chặn lại trên các đĩa 1.72 Mb được định dạng với các phiên bản Mtools trước 2.0.7, điều này có thể bỏ qua. Đặt biến môi trường MTOOLS_FAT_COMPATIBILITY (hoặc biến tập tin thiết lập tương ứng, `global variables') để bỏ qua kiểm tra FAT.

Các lênh hay sử dung

* *Lênh floppyd_installtest*

Floppyd_installtest được sử dụng để kiểm tra một chương trình floppyd daemon đang chạy. Lệnh này rất hữu ích, nếu có một frontend script tới mtools, nó quyết định có sử dụng **floppyd** hay không.

Cú pháp:

```
floppyd_installtest [-f] Connect-String
```

Tuỳ chọn

-f

floppyd_installtest thực hiện một xác nhận X-Cookie đầy đủ và báo lỗi nếu không làm việc.

Connect-String có định dạng được mô tả ở phần floppyd-section:

HOSTNAME:DISPLAYNUMBER[/BASEPORT]

* *Lênh mattrib*

Mattrib được dùng để thay đổi các thuộc tính của tập tin MS-DOS. Lệnh này có cú pháp như sau:

```
mattrib [-a |+a] [-h | +h] [-r | +r] [-s | +s] [-/
] [ -p ][-X] MSDOSFILE [ MSDOSFILES ... ]
```

Mattrib thêm các thuộc tính vào một tập tin MS-DOS (với ký hiệu phép toán ``+") loại bỏ thuộc tính (với ký hiệu phép toán ``-").

Mattrib hỗ trợ các bit thuộc tính sau:

a thuộc tính lưu trữ (Archive bit). Được sử dụng bởi các chương trình sao lưu để chỉ ra các tập tin mới

r thuộc tính chỉ đọc (Read-only bit). Dùng để xác định một tập tin chỉ đọc. Các tập tin có thuộc tính này không thể chỉnh sửa hay xoá bằng lệnh `DEL'

s thuộc tính hệ thống. Được MS-DOS dùng để xác định một tập tin hệ thống

h thuộc tính ẩn. Được dùng để giấu các tập tin khi dùng lệnh`DIR'

Mattrib cũng hỗ trợ các ký hiệu lệnh sau:

/ đệ quy. Liệt kê đệ quy danh sách các thuộc tính của các tập tin trong thư mục con

X rút gọn. In ra các thuộc tính không có các ký tự trống chèn giữa

Nếu tuỳ chọn "/" không được đặt, hay MSDOSFILE chứa các ký tự liên kết, và chỉ có một đường dẫn tập tin Msdos trên dòng lệnh, thì chỉ các thuộc tính được in ra, chứ không phải là tên tập tin. Tuỳ chọn này rất thuận tiện cho các scripts

p chế độ thực hiện lại (Replay). Đưa ra một dãy các lệnh **mformat** sẽ lưu lại trạng thái hiện thời, bắt đầu từ một trạng thái lưu lại bởi tập tin hệ thống DOS. Các lệnh chỉ đưa ra cho các thiết lập thuộc tính khác ngầm định. (Các thuộc tính lưu trữ đặt cho tập tin, không đặt cho thư mục). Tuỳ chọn này nhằm mục đích sử dụng thêm vào tar

**** Lênh mbadblocks***

Lệnh **mbadblocks** được sử dụng để quét một ổ mềm MS-DOS và đánh dấu các sector hỏng không dùng được là bad.

Lệnh này có cú pháp:

```
mbadblocks  DRIVE
```

Lệnh mbadblocks quét một ổ mềm MS-DOS để xác định các sector hỏng. Mọi bad sector được đánh dấu như thực tế trên bản FAT. Điều này nhằm tránh sử dụng các sector hỏng sau khi thực hiện **mformat**. Nó không dùng để cứu các sector hỏng.

Các lỗi:

Mbadblocks cũng cố gắng (nhưng không hoàn toàn) cứu các sector hỏng đang sử dụng bằng cách đọc lại nhiều lần, sau đó mới đánh dấu là hỏng.

**** Lênh mcat***

Lệnh **mcat** được dùng để sao chép toàn bộ nội dung ảnh đĩa từ một ổ mềm hoặc lên một ổ mềm.

Cú pháp lệnh:

```
mcat [ -w ] DRIVE
```

Lệnh mcat thực hiện cùng nhiệm vụ như lệnh **cat** của Linux. Nó được bao gồm trong các gói của mtools, vì lệnh **cat** không thể truy cập đến các ổ mềm mà tiến trình ổ mềm mtools yêu cầu. Bây giờ ta có thể tạo các ổ đĩa mềm khởi động từ xa bằng **mcat**.

Thao tác ngầm định là đọc. Đầu ra được ghi vào stdout.

Nếu tuỳ chọn `-w' được xác định, mcat đọc một ảnh đĩa từ stdin và ghi vào thiết bị được cho trước. ***Sử dụng cẩn thận*** Do tính tự nhiên cấp thấp của lệnh này, nó sẽ huỷ một cách hợp lệ bất cứ dữ liệu nào được ghi trên đĩa trước đó mà không hề cảnh báo.

Lênh mcd

Lệnh **mcd** được sử dụng để thay đổi thư mục làm việc mtools trên ổ đĩa MS-DOS . Nó sử dụng cú pháp sau:

```
mcd [thư-mục-MSDOS]
```

Không có đối số, **mcd** thông báo thiết bị hiện thời và thư mục đang làm việc. Mặt khác, **mcd** thay đổi thiết bị hiện thời và thư mục làm việc hiện thời liên quan đến một hệ thống tập tin MS-DOS .

Biến môi trường **MCWD** có thể sử dụng để xác định tập tin mà thông tin thiết bị và thư mục làm việc hiện thời được lưu trữ.

Giá trị ngầm định là `$HOME/.mcwd'. Thông tin trong tập tin này được bỏ qua nếu tập tin này đã tạo ra được giờ trở lên.

Lệnh **mcd** trả lại giá trị 0 khi hoàn thành hoặc 1 khi bị hỏng.

Không như lệnh 'CD' của MS-DOS , **mcd** có thể dùng để thay đổi sang một thiết bị khác. Nó có thể sáng suốt để xoá bỏ tập tin **.mcwd** cũ khi logout.

Lênh mcopy

Lệnh **mcopy** được sử dụng để copy các tập tin MS-DOS từ Linux/vào Linux.

Lệnh này có ba dạng:

```
mcopy [-b/ptnvmoQOsSrRA] tập-tin-gốc  tập-tin-đích

mcopy [-b/ptnvmoQOsSrRA] tập-tin-gốc ... tập-tin-đích

mcopy [-tnvm] tập-tin-gốc-MSDOS
```

Lệnh **mcopy** sao nội dung các tập tin được chỉ định đến các tập tin được đặt tên, hoặc copy nhiều tập tin đến một thư mục đã đặt tên. Nguồn và đích có thể đồng thời là tập tin MS-DOS hoặc Linux .

Việc sử dụng xác định tên ổ đĩa trên các tập tin MS-DOS , ví dụ 'a:', xác định hướng di chuyển dữ liệu. Việc không xác định tên ổ đĩa sẽ ám chỉ đến một tập tin Unix có đường dẫn là thư mục hiện thời.

Nếu tên ổ đĩa nguồn không có tập tin đi kèm, chẳng hạn

 # mcopy a:

thì mọi tập tin trong ổ đĩa sẽ được copy.

Nếu chỉ một tên tập tin, chẳng hạn

 # mcopy a:foo.exe

đường dẫn nguồn MS-DOS sẽ được cung cấp, một đích mặc định của thư mục hiện hành (``.'') được thừa nhận.

Một tên tập tin "`-" có nghĩa là đường vào chuẩn hoặc đường ra chuẩn, phụ thuộc vào vị trí của nó trên dòng lệnh.

Lệnh **mcopy** chấp nhận các tuỳ chọn lệnh sau:

-b

Chế độ mẻ (Batch mode): Tối ưu cho việc copy đệ quy lớn, nhưng không đảm bảo nếu có sự cố trong quá trình copy.

-/

Copy đệ quy. Đồng thời copy các thư mục và nội dung của chúng

-p

Bảo quản các thuộc tính của các tập tin được copy.

-Q

Khi **mcopy** nhiều tập tin, thoát ngay khi một copy bị hỏng (Ví dụ, để tiết kiệm bộ nhớ trên đĩa đích)

-t

Chuyển tập tin text, **mcopy** chuyển đổi phím xuống dòng/ tín hiệu xuống dòng nhận được thành tín hiệu xuống dòng.

-n

Không xác nhận khi ghi đè các tập tin Unix. **mcopy** không cảnh báo người sử dụng khi ghi đè một tập tin Unix đang tồn tại. Để chuyển thành không xác nhận các tập tin DOS , sử dụng **-o**.

-m

Lưu giữ thời gian chỉnh sửa tập tin cũ. Nếu tập tin đích đã tồn tại, và tuỳ chọn **-n** không hiện hành, **mcopy** hỏi ghi đè hoặc sửa tên tập tin mới (*Xem phần name clash*) .

Vướng mắc

Không như MS-DOS, tuỳ chọn dấu '+' (bổ sung) của MS-DOS không được hỗ trợ. Dù vậy, có thể sử dụn **mtype** để đưa ra hiệu quả tương tự:

```
mtype a:tập-tin1 a:tập-tin2  > tập-tin-Linux
```

```
mtype a:tập-tin1 a:tập-tin2 | mcopy - a:tập-tin-MSDOS
```

Lênh mdel

Lệnh **mdel** được sử dụng để xoá một tập tin MS-DOS, có cú pháp là:

```
mdel [-v] tập-tin-MSDOS...
```

Lệnh **mdel** hỏi kiểm tra trước khi xoá bỏ một tập tin chỉ đọc.

Lênh mdeltree

Lệnh **mdeltree** được dùng dể xoá một thư mục MS-DOS có cú pháp là:

```
mdeltree [-v] thư-mục-MSDOS ...
```

Lệnh **mdeltree** xoá bỏ một thư mục và tất cả các tập tin và các thư mục con bên trong từ một hệ thống tập tin MS-DOS. Một lỗi sẽ xuất hiện nếu thư mục cần xoá bỏ không tồn tại.

Lênh mdir

Lệnh **mdir** được dùng để hiển thị một thư mục MS-DOS. Cú pháp của lệnh này:

```
mdir [-/] [-f] [-w] [-a] [-X] tập-tin-MSDOS ...
```

Lệnh **mdir** hiển thị nội dung của một thư mục MS-DOS, hoặc các mục của một số tập tin MS-DOS.

Các tuỳ chọn lệnh **mdir**:

-/

Đệ quy đầu ra, giống như tuỳ chọn **-s** của DOS

-w

Đầu ra mở rộng. Với tuỳ chọn này, **mdir** in ra các tên tập tin ra màn hình theo chiều ngang và không hiển thị kích thước tập tin hay ngày tạo ra tập tin.

-a

Liệt kê các tập tin ẩn.

-f

Nhanh. Không cố tìm bộ nhớ trống. Trên các đĩa lớn, việc tìm ra số lượng bộ nhớ trống mất một khoảng thời gian đáng kể, như việc toàn bộ bảng FAT phải được đọc vào và quét. Tuỳ chọn **-f** sẽ bỏ qua bước này. Tuỳ chọn này không cần thiết trên các hệ thống tập tin FAT32, đã lưu trữ kích thước õ ràng.

-X

Liệt kê vắn tắt. Chỉ liệt kê một danh sách riêng biệt dòng mới của các tên đường dẫn mà không có phần trang trí hay thong tin bổ sung.

Một lỗi sẽ xuất hiện nếu một thành phần của đường dẫn không phải là một thư mục.

Lệnh mdu

Lệnh mdu được dùng để liệt kê bộ nhớ mà một thư mục chiếm, bao gồm các thư mục con và các tập tin. Nó tương tự lệnh **du** của Unix. Đơn vị được sử dụng là clusters. Dùng lệnh **mifo** để tìm ra kích thước cluster.

```
mdu [tùy-chọn] [ tập-tin-MSDOS ... ]
```

Với tuỳ chọn:

-a

Tất cả các tập tin. Đồng thời liệt kê bộ nhớ mà các tập tin riêng lẻ sử dụng.

-s

Chỉ hiển thị bộ nhớ tổng, không chi tiết từng thư mục con.

Lệnh mformat

Lệnh **mformat** được sử dụng để thêm một hệ thống tập tin MS-DOS vào một đĩa định dạng cấp thấp. Cú pháp lệnh này là:

```
mformat [tùy-chọn] ổ-đĩa:
```

Lệnh **mformat** thêm một hệ thống tập tin MS-DOS tối thiểu (boot sector, FAT, và thư mục gốc) lên một đĩa đã định dạng bằng một định dạng cấp thấp Unix.

Các tuỳ chọn sau được hỗ trợ: (Tuỳ chọn -S, -2, -1 và -M có thể không có nếu bản **mstools** được biên dịch không có tuỳ chọn USE_2M).

-t `số-trụ`

Đĩa cần định dạng có số lượng trụ (số lượng rãnh) là **`số-trụ`**.

-h `số-mặt`

Đĩa cần định dạng có số lượng mặt là **`số-mặt`**.

-s `số-sector`

Đĩa cần định dạng có số lượng sector trên một mặt là **`số-sector`**.

Nếu tuỳ chọn **-2** (chế độ đặt mềm) được đặt trước, là số lượng sector độ dài 512 byte tương đương với rãnh cùng loại (tức là không có mặt 0, rãnh 0). Nếu tuỳ chọn **-2** không được đặt, là số lượng các sector vật lý trên mỗi rãnh (có thể lớn hơn 512 byte).

-I nhãn

Đĩa được định dạng có tên là **nhãn**.

-S mã-kích-thước

Kích thước của sector tính theo byte là 2^(**mã-kích-thước** +7)

-2 sector-rãnh-0

Tham số này theo chế độ gọi ***2m*** cho phép kích thước sector ở rãnh 0 mặt 0 lớn hơn thông thường: số lượng sector trên track 0, head 0 của đĩa định dạng là **sector-rãnh-0**.

-1

Không sử dụng định dạng ***2m***, thậm chí khi kiểu đĩa (geometry) hiện thời là kiểu định dạng ***2m***.

-M cỡ-sector-phần-mềm

Kích thước sector định theo phần mềm. Tham số này mô tả kích thước sector tính theo byte được hệ thống tập tin MS-DOS sử dụng là **cỡ-sector-phần-mềm**. Ngầm định đó là kích thước vật lý của sector.

-a

Nếu tuỳ chọn này được sử dụng, một số hiệu kiểu Atari sẽ được tạo ra. Ataris lưu trữ số serial của nó trong nhãn OEM.

-X

Định dạng đĩa như một đĩa XDF. Các đĩa đã được định dạng mức thấp sử dụng tiện ích **xdfcopy** nằm trọng gói ***fdutils***.

-C

Tạo tập tin ảnh đĩa để cài đặt hệ thống tập tin MS-DOS trên đó. Rõ ràng, điều này vô dụng trên các thiết bị vật lý chẳng hạn các ổ đĩa mềm và các phân vùng ổ cứng.

-H sector-bị-che

Số lượng các sector bị che (còn gọi là số lượng các sector đi trước đĩa lôgic) là **sector-bị-che**. Tham số này rất hữu ích cho việc định dạng các phân vùng ổ cứng, với các đường biên track không được sắp thẳng hàng, chẳng hạn, mặt đầu tiên của rãnh đầu tiên không thuộc phân vùng nhưng lại chứa một bảng phân vùng. Trong trường hợp này, số lượng của các sector ẩn chung với số lượng các sector trên cylinder. Điều này đang được kiểm chứng.

-n

Số serial

[-0 RATE_ON_TRACK_0] [-A RATE_ON_OTHER_TRACKS]

[-1] [-k] ổ-đĩa:

-F

Định dạng phân vùng như FAT32 (đang thực nghiệm).

-I phiên-bản-httt

Đặt chỉ số phiên bản hệ thống tập tin là **phiên-bản-httt** khi định dạng một ổ đĩa FAT32. Để biết được điều này, chạy **minfo** trên một ổ đĩa FAT32 đang tồn tại.

-c cỡ-cluster

Đặt kích thước của một cluster (số lượng sector trong một cluster) của đĩa cần đinh dạng. Nếu kích thước cluster này tạo ra một bảng FAT quá lớn với số lượng các bit của nó, **mtools** sẽ tự động tăng kích thước cluster, cho đến khi bảng FAT nhỏ xuống phù hợp.

-r số-sector-ở-gốc

Đặt kích thước (số lượng tính theo sector) của thư mục gốc là **số-sector-ở-gốc**. Chỉ thích hợp cho bảng FAT 12 bit và 16 bit.

-B boot-sector

Sử dụng bootsector được lưu trong tập tin hay device cho trước (được xác định bởi **boot-sector**), thay vì sử dụng boot sector của đĩa cần định dạng. Chỉ có các trường định dạng được cập nhật để phù hợp với các tham số đĩa đích.

-k

Giữ các boot sector đang tồn tại càng nhiều càng tốt. Chỉ có các trường định dạng được cập nhật để phù hợp các tham số đĩa đích.

-0 cấp-độ-rãnh-0

Cấp độ truyền dữ liệu trên track 0 là **tốc-độ-rãnh-0.**

-A tốc-độ-rãnh-khác

Tỉ lệ chuyển dữ liệu trên các track khác là **tốc-độ-rãnh-khác.**

Để định dạng một đĩa có mật độ khác ngầm định, phải cung cấp (ít nhất) các tham số dòng lệnh khác với ngầm định trên đây.

Lệnh **mformat** trả lại giá trị 0 khi thực hiện thành công và 1 khi lỗi. Lệnh này không ghi lại các thông tin của các khối bad lên bảng FAT, sử dụng lệnh **mkmanifest** để làm việc này.

Lệnh mkmanifest

Lệnh **mkmanifest** được sử dụng để tạo một shell script (danh sách đóng gói) phục hồi các tên tập tin Unix . Cú pháp lệnh này như sau:

```
mkmanifest [tập-tin]
```

Lệnh **mkmanifest** tạo một shell script hỗ trợ trong việc phục hồi các tên tập tin UNIX bị mất bởi giới hạn tên tập tin của MS-DOS.

Các tên tập tin MS-DOS bị giới hạn trong 8 ký tự phần tên và 3 ký tự phần mở rộng, chỉ chấp nhận kiểu chữ hoa, không được trùng tên thiết bị,và không được chứa ký tự không hợp lệ.

Chương trình **mkmanifest** tương thích với các phương pháp được sử dụng trong **pcomm, arc** và **mstools** để thay đổi các tên tập tin Unix chuẩn phù hợp với giới hạn tên của MS-DOS. Lệnh này chỉ hữu dụng khi hệ thống đích (sẽ đọc đĩa) không thể kiểm soát các tên tập tin dài VFAT.

Ví dụ,

Khi sao chép các tập tin Unix có tên đặc biệt (không theo quy tắc đạt tên tập tin của MS-DOS) như sau đây lên một đĩa MS-DOS (sử dụng lệnh **mcopy**).

very_long_name

2.many.dots

illegal:

good.c

prn.dev

Capital

mcopy sẽ chuyển các tên tập tin thành:

very_lon

2xmany.dot

illegalx

good.c

xprn.dev

capital

Lệnh:

```
# mkmanifest very_long_name 2.many.dots illegal:
good.c prn.dev Capital >manifest
```

sẽ tạo ra như sau:

mv very_lon very_long_name

mv 2xmany.dot 2.many.dots

mv illegalx illegal:

mv xprn.dev prn.dev

mv capital Capital

Lưu ý rằng tập tin "good.c" không cần chuyển đổi nên không xuất hiện trong kết quả chuyển.

Giả sử đã copy các tập tin này từ đĩa mềm lên một hệ thống Unix khác, và bây giờ muốn các tên tập tin quay trở lại tên ban đầu. Nếu tập tin "manifest" (tập tin đầu ra có được khi thực hiện lệnh **mkmanifest** ở trên) được đi kèm với các tập tin này, nó sẽ được dùng để chuyển đổi lại các tên tập tin như cũ.

Vướng mắc

Các tên ngắn được tạo bởi lệnh **mkmanifest** theo cách chuyển đổi cũ (từ mtools-2.0.7) không giống trong Windows 95 và mtools-3.0.

Lệnh minfo

Lệnh **mifor** hiển thị các tham số của hệ thống tập tin MS-DOS, chẳng hạn số các sector, các đầu đọc và các cylinder. Nó cũng đồng thời in ra một dòng lệnh **mformat** dùng để tạo một hệ thống tập tin DOS tương tự trên một môi trường khác. Tuy nhiên lệnh này không dùng được trên môi trường 2m hoặc Xdf, và trên hệ thống tập tin Dos 1.0.

Cú pháp lệnh

minfo [tùy-chọn] ổ-đĩa:

Lệnh **mifor** có tuỳ chọn sau:

-v

In ra một hexdump của bootsector, thêm vào các thông tin khác.

Lệnh mlabel

Lệnh **mlabel** dùng để thêm một nhãn đĩa vào ổ đĩa, cú pháp như sau:

mlabel [-vcs] ổ-đĩa:[nhãn-mới]

Lệnh **mlabel** hiển thị nhãn đĩa hiện thời, nếu có. Nếu phần **nhãn-mới** không được cho, và nếu tuỳ chọn **-c** hay **-s** không được đặt, nó sẽ yêu cầu người sử dụng nhập một nhãn đĩa mới. Để xoá một nhãn đĩa hiện thời, ấn phím return tại dấu nhắc.

Vì sự thận trọng hợp lý, cần phải tạo một nhãn đĩa MS-DOS hợp lệ. Nếu đặt một nhãn đĩa không hợp lệ, , **mlabel** sẽ thay đổi nhãn đĩa (và hiển thị nhãn đĩa mới nếu mode chi tiết (verbose) được đặt). Lệnh **mlabel** trả lại giá trị 0 khi thực hiện thành công và 1 khi hỏng.

Lệnh **mlabel** có các tuỳ chọn sau:

-c

Xoá một nhãn đĩa hiện thời mà không hỏi người sử dụng

-s

Hiện ra nhãn đĩa hiện thời mà không hỏi người sử dụng

Lệnh mmd

Lệnh **mmd** được sử dụng để tạo một thư mục con MS-DOS. Cú pháp của nó là:

mmd [-voOsSrRA] thư-mục-MSDOS...

Lệnh **mmd** tạo ra một thư mục trên một hệ thống tập tin MS-DOS. Nếu tên thư mục **thư-mục-MSDOS** cần tạo đã tồn tại thì báo lỗi.

Lệnh mmount

Lệnh **mmount** được dùng để ***mount*** một đĩa MS-DOS. Lệnh này chỉ có trên Linux, và cũng chỉ hữu ích khi nhân OS cho phép cấu hình hình dạng của đĩa. Cú pháp của nó như sau:

mmount ổ-đĩa-MSDOS [thư-mục-mount]

Lệnh **mmount** đọc boot sector của một đĩa MS-DOS, định cấu hình định dạng đĩa, và cuối cùng gắn vào đích **thư-mục-mount**.

Nếu không xác định đối số mount , tên của thiết bị sẽ được sử dụng. Nếu đĩa có thuộc tính chống ghi (write protected), lệnh này sẽ tự động mount chế độ chỉ đọc (read only).

Lênh mmove

Lệnh `mmove' được dùng để di chuyển hoặc đổi tên một tập tin hoặc thư mục MS-DOS đang tồn tại.

`mmove' [`-voOsSrRA'] Tập-tin-nguồn Tập-tin-đích

`mmove' [`-voOsSrRA'] Tập-tin-nguồn [Tập-tin-nguồn...] Tập-tin-đích

`Mmove' di chuyển hoặc đổi tên một tập tin hay thư mục MS-DOS đang tồn tại.

Không như MOVE của phiên bản MS-DOS, `mmove' có khả năng di chuyển các thư mục con. Các tập tin hoặc thư mục con chỉ có thể di chuyển bên trong một hệ thống tập tin. Dữ liệu không thể được di chuyển từ Dos sang Unix hoặc ngược lại. Nếu bỏ sót tên ổ đĩa của tập tin hay thư mục đích, lệnh này sẽ đặt bằng tên ổ đĩa của tập tin nguồn . Nếu không đánh tên ổ đĩa cho mọi tham số, ổ đĩa a: sẽ ngầm định được đặt.

Lênh mpartition

Lệnh `mpartition' được sử dụng để tạo một hệ thống tập tin MS-DOS như một phân vùng. Lệnh này với mục đích sử dụng trên một hệ thống không phải Linux, chẳng hạn các hệ thống không sử dụng được fdisk và các truy nhập dễ dàng đến các thiết bị Scsi. Lệnh này chỉ làm việc trên các ổ đĩa có các biến phân vùng đã được đặt.

Cú pháp lệnh này như sau:

`mpartition' `-p' DRIVE

`mpartition' `-r' DRIVE

`mpartition' `-I' [`-B' BOOTSECTOR] DRIVE

`mpartition' `-a' DRIVE

`mpartition' `-d' DRIVE

`mpartition' `-c' [`-s' SECTORS] [`-h' HEADS]

[`-t' CYLINDERS] [`-v' [`-T' TYPE] [`-b'

BEGIN] [`-l' length] [`-f']

Lệnh mpartition hỗ trợ các thao tác sau:

`p'

In ra một dòng lệnh để tạo lại phân vùng cho ổ đĩa DRIVE.

Lệnh này sẽ không in ra gì nếu phân vùng cho ổ đĩa không được xác định, hoặc xác định được một mâu thuẫn nào đó. Nếu tuỳ chọn chi tiết (`-v') được đặt, lệnh này sẽ in ra bảng phân vùng.

`r'

Xoá bỏ phân vùng được mô tả bởi DRIVE.

`I'

Khởi tạo bảng phân vùng, và xoá bỏ mọi phân vùng.

`c'

Tạo phân vùng được mô tả bởi DRIVE.

`a'

"Kích hoạt" phân vùng, chẳng hạn tạo khả năng boot. Chỉ có một phân vùng có thể boot vào một thời điểm.

`d'

"Không kích hoạt" phân vùng, chẳng hạn không cho phép boot.

Nếu không có hoạt động nào được cho trước, thiết đặt hiện thời sẽ được in ra.

Đối với việc tạo phân vùng, sẽ có các tuỳ chọn sau:

`s SECTORS'

Xác định số lượng các sector trên track của phân vùng (cũng đồng thời là số lượng của sector trên track cho toàn bộ ổ đĩa).

`h HEADS'

Xác định số đầu đọc (heads) của phân vùng (cũng đồng thời là số đầu đọc của toàn bộ ổ đĩa). Ngầm định, thông tin định dạng này(số lượng các sectors và heads) được tính từ các mục nhập của các bảng phân vùng bên cạnh, hoặc ước đoán từ kích thước phân vùng.

`t CYLINDERS'

Xác định số cylinders của phân vùng (không phải số lượng cylinder của toàn bộ ổ đĩa.

`b BEGIN'

Xác định khoảng chừa trống bắt đầu của phân vùng, xác định bằng các sector. Nếu phần BEGIN không được đặt, *mpartition* sẽ đặt phân vùng bắt đầu từ phần đầu tiên của ổ đĩa (phân vùng số 1), hoặc ngay sau phần kết thúc của phân vùng trước.

`l LENGTH'

Xác định kích thước (chiều dài) của phân vùng, xác định bằng các sector. Nếu phần kết thúc không được đặt, *mpartition* tính ra kích thước từ số lượng các sectors, heads và cylinders. Nếu các thông tin này cũng không được cho trước, nó sẽ tạo phân vùng lớn nhất có thể, tuỳ vào kích thước đĩa hoặc phần bắt đầu của phân vùng tiếp theo.

Các thao tác chỉnh sửa bảng phân vùng sẽ có các tuỳ chọn sau:

`f'

Thông thường, trước khi ghi lại các thông tin sửa đổi vào phân vùng,

mpartition thực hiện các kiểm tra chắc chắn, chẳng hạn kiểm tra xem các phân vùng có gối chồng lên nhau không và định vị có đúng không. Nếu một trong các kiểm tra này hỏng, bảng phân vùng sẽ không được thay đổi. Tuỳ chọn `-f'' cho phép bỏ qua các kiểm tra an toàn này.

Tất cả các thao tác với phân vùng sẽ có các tuỳ chọn sau:

`v'

Đi cùng với tuỳ chọn `-p' in ra thông tin bảng phân vùng hiện tại (không có các thao tác chỉnh sửa), hoặc thông tin sau khi phân vùng thực hiện các chỉnh sửa.

`vv'

Nếu tuỳ chọn chi tiết 'v'được đưa ra hai lần, *mpartition* sẽ in ra hexdump của bảng phân vùng từ khi đọc nó đến khi ghi nó vào thiết bị.

Quá trình khởi tạo bảng phân vùng sẽ có tuỳ chọn sau:

`B BOOTSECTOR'

Đọc master boot record mẫu từ tập tin BOOTSECTOR.

Lênh mrd

===

Lệnh `mrd' được sử dụng để xoá bỏ một thư mục con MS-DOS. Cú pháp lệnh này là:

`mrd' [`-v'] MSDOSDIRECTORY [MSDOSDIRECTORIES...]

`Mrd' Xoá một thư mục con từ một hệ thống tập tin MS-DOS. Lệnh này báo lỗi khi tên thư mục con cần xoá không tồn tại hoặc không rỗng.

Lênh mren

====

Lệnh `mren' được sử dụng để đổi tên hay di chuyển một tập tin hoặc một thư mục con MS-DOS đang tồn tại. Cú pháp của nó là:

`mren' [`-voOsSrRA'] SOURCETậP TIN TARGETTậP TIN

`Mren' đổi tên một tập tin đang tồn tại trên một hệ thống tập tin MS-DOS.

Trong mode chi tiết (verbose mode), `Mren' hiển thị tên tập tin mới nếu tên cung cấp không hợp lệ.

Nếu cú pháp đầu tiên được sử dụng (chỉ một Tập-tin-nguồn), và tên Tập-tin-đích không chứa bất kỳ dấu xoạc chéo '/' hoặc dấu hai chấm ':', thì tập tin hoặc thư mục đó sẽ được đổi tên trong cùng thư mục hiện tại, không giống như lệnh `*mmove*' trong trường hợp này sẽ chuyển đến thư mục `mcd' hiện thời. Không giống lệnh `REN' của phiên bản MS-DOS, `mren' có thể sử dụng để đổi tên các thư mục.

Lệnh mshowfat

Lệnh `*mshowfat*' được sử dụng để hiển thị các mục nhập bảng FAT cho một tập tin. Cú pháp lệnh:

```
`$ mshowfat Tập-tin'
```

Lênh mtoolstest

Lệnh `*mtoolstest*' dùng để kiểm tra các tập tin cấu hình mtools. Để gọi lệnh này chỉ cần gõ `mtoolstest' mà không cần đối số nào.

`Mtoolstest' đọc các tập tin cấu hình mtools, và in cấu hình luỹ tích (cumulative configuration) ra `stdout'. Bản thân kết quả có thể sử dụng như một tập tin cấu hình (mặc dù có thể muốn xoá bỏ các mệnh đề thừa). có thể sử dụng chương trình này để chuyển đổicác tập tin cấu hình kiểu cũ thành các tập tin cấu hình mới.

Lênh mtype

Lệnh `*mtype*' được dùng để hiển thị nội dung của một tập tin MS-DOS.

Cú pháp của nó là:

`mtype' [`-ts'] MSDOSTậP TIN [MSDOSTậP TINS...]

`*Mtype*' hiển thị tập tin MS-DOS được chỉ định lên trên màn hình.

Ngoài các tuỳ chọn chuẩn, `*Mtype*' còn cho phép các tuỳ chọn dòng lệnh sau:

`t'

Xem nội dung tập tin text. `*Mtype*' có thể dịch các phím xuống dòng/tín hiệu chuyển dòng.

`s'

`*Mtype*' bỏ đi các bit cao của dữ liệu.

Lệnh `*mtype*' có thể dùng để thiết lập các thiết bị và các thư mục làm việc hiện thời (Tương tự MS-DOS), mặt khác, ngầm định là `A:/'.

`*Mtype*' trả lại giá trị 0 khi hoàn thành, 1 khi hỏng hoàn toàn, hoặc 2 khi hỏng một phần.

KHông như `TYPE' trong phiên bản của MS_DOS, `*mtype*' cho phép sử dụng nhiều đối số.

Lênh mzip

Lệnh `*mzip*' dùng để đưa racác lệnh ổ đĩa ZIP cụ thể trên Solaris hay HPUX. Cú pháp lệnh này là::

`mzip' [`-epqrwx']

`*Mzip*' cho phép sử dụng cácc tuỳ chọ dòng lệnh sau:

`e'

Đưa ổ đĩa ra ngoài.

`f'

Bắt buộc đưa ổ đĩa ra thậm chí khi đĩa đang được mount (phải được đặt thêm trước vào tuỳ chọn `-e').

`r'

Xác lập đĩa chống ghi.

`w'

Bỏ thuộc tính chống ghi của đĩa.

`p'

Đặt Password chống ghi.

`x'

Đặt Password bảo vệ

`u'

Gỡ bỏ tạm thời các đặc tính bảo vệ đĩa cho đến khi đưa đĩa ra ngoài. Khi đặt tuỳ chọn này, đĩa sẽ ghi được, và chuyển lại trạng thái bảo vệ cũ sau khi đưa đĩa ra ngoài.

`q'

Truy vấn trạng thái

Để gỡ bỏ password, đặt ổ đĩa vào một trong các mode không bảo mật `-r'

hoặc `-w': mzip sẽ hỏi password, và sau đó gỡ bỏ chế độ bảo mật của ổ đĩa. Nếu quên password, có thể khắc phục bằng cách format cấp thấp ổ đĩa đó (sử dụng BIOS setup của SCSI adaptor).

ổ đĩa ZipTools được chuyển cùng ổ đĩa cũng được đặt chế độ bảo mật bằng password.

Trên một hệ Dos hoặc Mac, password này được tự động xoá khi ZipTools đã được cài đặt. Từ những bài gửi đưa lên Usenet, Tôi đã biết được password cho ổ đĩa tools là `APlaceForYourStuff. Mzip biết password này, và thử nó đầu tiên rtước khi hỏi password của bạn. Vì vậy `mzip -w z:' gỡ bảo mật cho ổ đĩa tools . ổ đĩa tools được formattheo một cách đặc biệt nên có thể sử dụng cả trong PC và trong Mac. Trên một PC, hệ thống tập tin Mac như một tập tin ẩn có tên `partishn.mac'. có thể xoá nó để có thêm 50 Meg bộ nhớ bị chiếm bởihệ thống tập tin Mac.

Lênh xcopy

Script `*xcopy*' được dùng để copy đệ quy một thư mục đến một thư mục khác. cú pháp của nó như sau:

```
xcopy Thư-mục-nguồn Thư-mục-đích
```

Nếu thư-mục-đích không tồn tại, nó sẽ được tạo ra. Nếu đã tồn tại, các tập tin của thư mục con sẽ được copy thẳng đến đó, và không có Thư-mục-nguồn nào được tạo ra, không như với `cp -rf'.

Vướng mắc

Lệnh này là một giải pháp rất thô sơ. Một thao tác đúng cách sẽ khiến nhiều phần của mtools phải hoạt động lại, nhưng thật không may bây giờ tôi không có thời gian để làm điều này. Nhược điểm chính của phương pháp này là không có hiệu quả trên một vài cấu trúc (một số lệnh gọi lần lượt đến mtools, cái làm hỏng các caching của mtools).

PHỤ LỤC A. QUÁ TRÌNH CÀI ĐẶT LINUX

A.1. Giới thiệu sơ bộ về Linux

Nếu chúng ta đang tìm một Hệ điều hành có tốc độ cao, đáng tin cậy, không đắt tiền, có thể cho nhiều người dùng cùng sử dụng một lúc, có khả năng làm server cho mạng Internet đồng thời hỗ trợ các giao diện bắt mắt thì đó chính là Linux.

Khắp nơi trên thế giới, mỗi ngày có hàng ngàn người dùng mới muốn khám phá sức mạnh của hệ điều hành có bộ mã nguồn mở, và có xuất xứ từ Unix này. Vậy có những bí ẩn thú vị nào bên trong Linux ?

Trước hết khó mà làm cho Linux bị ngưng trệ và tê liệt. Đã có nơi thử nghiệm nhiều hệ thống chạy Linux liên tục hàng năm trời mà không phải khởi động lại. Linux có thể chạy trên các máy tính thế hệ cũ vốn không thể chạy Windows 95, 98, thậm chí cả những máy 486 vứt trong nhà kho.

Trên đây là những điểm khiến Linux được nhiều người ủng hộ. Thế nhưng việc cài đặt hệ điều hành này có thể làm lo lắng. Linux được phân phối tự do nên có nhiều công ty tìm cách đưa ra thị trường bản phân phối riêng của họ. Ngoài chương trình Linux cốt lõi, khác nhau cơ bản ở phần mềm đi kèm và cách thức cài đặt. Các nhà phân phối Linux phổ biến hiện nay là Red Hat Software, Caldera, Slackware, S.U.S.E và Debian.

Trong phụ lục này chúng ta giới thiệu quá trình cài đặt bản Linux của Red Hat Software 6.2.

A.2. Chuẩn bị cho việc cài đặt

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. phải 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. 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 này. 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 sẽ 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 thế, hệ thống có thể bị treo.

Nếu có thể hãy ghi số điện thoại của một người dùng Linux có kinh nghiệm và gọi cho họ khi cần thiết, hoặc các có thể gọi cho chúng tôi theo số điện thoại ***7761075***, chúng tôi sẽ rất vui khi được giải đáp những vướng mắc của các .

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 3 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 sẽ cao hơn).

A.3. 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 thì chúng ta sẽ nhận kèm đĩa khởi động. Còn nếu mua bản copy từ công ty thứ 3, chúng 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.

Bây giờ chúng ta 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.

A.4. 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,... Chúng ta cần phải 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 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! Như vậy chẳng dại gì mà chúng ta lại không dùng một phần mềm miễn phí cực mạnh như PQMagic.

A.5. 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.

A.5.1. Lưa chon chế đô cài đăt

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 chúng 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.

A.5.2. Lưa chon ngôn ngữ hiển thi.

Hệ thống đưa ra rất nhiều ngôn ngữ cho chúng ta lựa chọn, ví dụ như Czech, English, French, German.... Thường là chọn English.

A.5.3. Lưa chon cấu hình bàn phím

Linux đòi hỏi lựa chọn cấu hình bàn phím từ các mô hình sau:

- Model :

  Brazilian ABNT 2

  Dell

  Generic.

  Microsoft.

  .....

Chúng ta sẽ lựa chọn bàn phím tương thích theo các dạng trên, nếu không rõ bàn phím của chúng ta 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.

A.5.4. Chon cấu hình chuôt.

Hệ thống đưa ra 10 loại chuột để chúng ta lựa chọn loại tương thích với chuột của mình, nếu không biết rõ chuột chúng ta 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.

A.5.5. Hệ thống đưa ra lời giới thiệu về bản Red Hat đang cài đặt.

A.5.6. Lưa chọn kiểu cài đăt.

Install (cài mới) gồm có các chế độ:

GNome Workstation.

KDE Workstation.

Server

Custom

Upgrade (Nâng cấp)

- **Tuỳ chọn cài đặt WorkStation.**

Nếu chúng ta 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 phù hợp nhất đối 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 chúng ta 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 cho chúng ta phải cấu hình nhiều thành phần phần cứng. Và khi cài đặt theo kiểu này thì chúng ta 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 chúng ta 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, chúng tôi sẽ 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.

A.5.7. Xác đinh các Partition

Đầu tiên chúng ta phải 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 chúng ta sẽ 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 chúng ta có kinh nghiệm về RAID.

Drive Summaries: hiển thị thông tin về cấu hình đĩa.

A.5.8. Chon Partition để Format.

Gồm có các thông tin:

Partition muốn Format.

Lựa chọn "Check for bad blocks while formating"

Chúng ta sẽ 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.

A.5.9. Chon 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. có thể 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.

A.5.10. Chon 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 mục "System clock uses UTC".

A.5.11. Thiết đăt cấu hình Account (người sử dung)

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.

A.5.12. Thiết đăt cấu hình quyền han (Authentication Configuration)

Có 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.

A.5.13. Lưa chon các gói phần mềm cài đăt (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 tập tin 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.

A.5.14. 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

Xserver: 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. Tốt nhất là nên sắm một chiếc card màn hình loại chuẩn mà cũ một chút cũng được.

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. (Nếu muốn biết rõ hơn về phần này, hãy đọc phụ lục 2)

A.5.15. 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.

Cuộc chu du vào thế giới Linux của bắt đầu từ đây!

A.6. Các hạn chế về phần cứng đối với Linux

A.6.1. Các bô 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 sẽ 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.

Hiện tại Apple đã hỗ trợ MkLinux trên các Power Macs dựa trên OSF của Mach vi nhân.

Linux cho máy 64 bit DEC Alpha/AXP.

Hiện tại người ta cũng đang nghiên cứu về Linux cho MIPS, bắt đầu đối với R4 trên các máy Deskstation Type.

Hiện đang 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.

Cũng có bản "Hardhat" cho các máy SGI/Indy.

A.6.2. 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.

A.6.3. Các yêu cầu về bô 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 64MB bộ nhớ vật lý chẳng hạn như Netscape.

Linux có thể sử dụng được bao nhiêu dung lượng bộ nhớ ?

Thực ra cũng chưa có tài liệu nào nói chính xác về vấn đề này nhưng hiện tại thì Linux có thể hỗ trợ được 128MB Ram. Ta có thể làm thêm các thí nghiệm khác để kiểm chứng khả năng quản lý bộ nhớ của Linux.

A.6.4. 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).

Liệu ta có thể chạy một chương trình Windows trong Linux? Một chương trình tên WINE đang được nghiên cứu 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:\LINUX, 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.