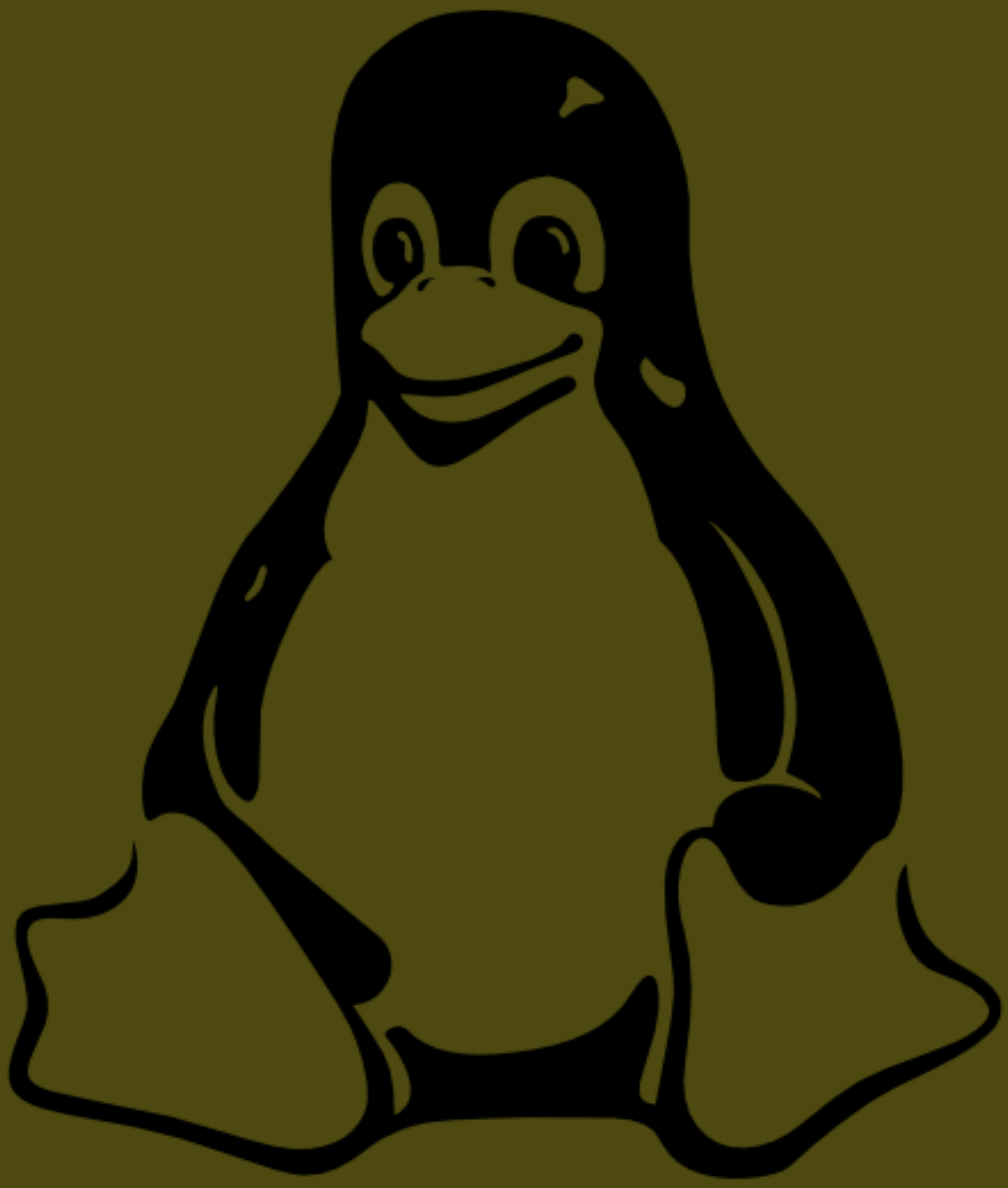
Tự học sử dụng linux

Phan Vĩnh Thịnh



Tự học sử dụng Linux

Tác giả: Kostromin V. A.
Dịch và cộng tác: Phan Vĩnh Thịnh

Phiên bản: 0.9.4
Ngày 13 tháng 9 năm 2006

Dành cho người dùng mới và rất mới...

Mục lục

Danh sách hình vẽ

Danh sách bảng

Lời mở đầu

Đây là bản dịch cuốn "Linux cho người dùng" (sêri sách tự học) của Kostromin Victor Alexeevich cộng thêm một vài kinh nghiệm sử dụng Linux của người dịch. Bản gốc được viết trên tiếng Nga. Theo yêu cầu của Kostromin A. V., xin được đưa ra các liên kết tới bản gốc sau đây:
http://rus-linux.net/book1.php?name=book1/oglavl
http://linux-ve.chat.ru/

Cảm ơn

Trước tiên cần cảm ơn **Kostromin V. A.** đã viết một cuốn sách về Linux cho người dùng mới tuyệt vời, hai bác **Nguyễn Đại Quý** và **Nguyễn Đặng Hoàng Tuân** đã giúp trong việc sử dụng LaTeX. Bác **Nguyễn Đại Quý** đã đọc và sửa cho phiên bản 0.9. Xin hãy gửi thư nhắc người dịch tại teppi82@gmail.com nếu như người dịch có quên ai đó.

Bản quyền

Chương 1

HĐH Linux: lịch sử và các bản phân phối

"Just for fun" – Linus Torvalds.

Người dịch: *Lịch sử luôn là điểm khởi đầu khi nghiên cứu một ngành khoa học nào đó. Không có ngoại lệ đối với Toán học, Vật lý, môn chuyên ngành của tôi – Hoá học và tất nhiên cả HĐH Linux. Trong chương đầu tiên của cuốn sách "Tự học sử dụng Linux" này chúng ta sẽ trả lời ngắn gọn cho câu hỏi "Linux là gì?". Đồng thời nói đôi dòng về những điểm đặc biệt của Linux, yêu cầu của Linux đối với phần cứng, khái niệm bản phân phối Linux, và cách có được những bản phân phối này. Hơn thế nữa bạn đọc sẽ hiểu ít nhiều về OpenSource, GNU và FSF.*

1.1 Thế nào là HĐH nói chung và Linux nói riêng

1.1.1 Các hệ điều hành dạng UNIX

Hệ điều hành (HĐH) đó là một bộ các chương trình hỗ trợ việc điều khiển phần cứng của máy tính, tổ chức làm việc với các tập tin (trong đó có chạy và điều khiển việc thực hiện của các chương trình), và đồng thời thực thi sự giao tiếp với người dùng, tức là dịch các câu lệnh của người dùng và hiển thị kết quả làm việc của những lệnh này.

Không có hệ điều hành thì máy tính không thực hiện được chức năng của mình. Trong trường hợp đó máy tính chỉ là một tập hợp các thiết bị điện tử không làm việc, không hiểu là để làm gì.

Đến thời điểm hiện nay thì các hệ điều hành nổi tiếng nhất cho máy tính là Microsoft Windows (C) và UNIX. Windows bắt nguồn từ hệ điều hành MS-DOS trước đây làm việc trên các máy tính của hãng IBM. Hệ điều hành UNIX do nhóm các nhà phát triển Bell Labs viết ra vào năm 1969 dưới sự điều khiển của Dennis Ritchie, Ken Thompson và Brian Kernighan. Nhưng bây giờ khi nói đến hệ điều hành UNIX thường có ý không nói cụ thể một hệ điều hành cụ thể nào mà là một nhóm các hệ điều hành dòng UNIX (UNIX-liked OS). Chính bản thân từ UNIX (viết hoa tất cả các chữ cái) trở thành nhãn hiệu thương mại của tổng công ty AT&T.[1]

[1]Người dịch: Người mỹ "không ngại ngần" đăng ký nhãn hiệu thương mại bất kỳ thứ gì, kể cả Yoga mà bắt nguồn từ Ấn Độ.

Vào cuối những năm 70 của thế kỷ trước (thế kỷ XX) các nhà phát triển của trường đại học California ở Berkeley đã thêm vào mã nguồn của UNIX rất nhiều sự cải tiến trong đó có hỗ trợ giao thức[2] TCP/IP (giao thức mạng chính hiện nay). Sản phẩm này nổi tiếng dưới tên BSD ("**B**erkeley **S**ystems **D**istribution"). Điều đặc biệt ở chỗ bản quyền của sản phẩm cho phép người khác phát triển và cải tiến và chuyển kết quả thu được đến người thứ ba (cùng với mã nguồn hoặc không) với điều kiện là phải chỉ ra phần nào của mã được phát triển ở Berkeley.

Hệ điều hành dòng UNIX, trong đó có BSD, lúc đầu được phát triển để làm việc với các máy tính nhiều người dùng – các mainframe. Nhưng dần dần cấu hình trang thiết bị của máy tính cá nhân cũng mạnh lên và hiện nay có khả năng cao hơn so với những mainframe của những năm 70 thế kỷ trước. Và và đầu những năm 90 một sinh viên của trường đại học Helsinki (Phần Lan), Linus Torvalds, đã bắt đầu phát triển một HĐH kiểu UNIX cho các máy tính cá nhân tương thích với IBM (IBM-compatible PC).

1.1.2 Một chút về lịch sử

HĐH Linux vừa kỷ niệm sinh nhật lần thứ 15 của mình. Đây là bức thư mà Linus gửi vào nhóm tin tức **comp.os.minix** ngày 25 tháng 8 năm 1991 (được coi là ngày sinh nhật của HĐH này):

```
From: torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)
Newsgroups: comp.os.minix
Subject: What would you like to see most in minix?
Summary: small poll for my new operating system
Message-ID: <1991Aug25.205708.9541@klaava.Helsinki.FI>
Date: 25 Aug 91 20:57:08 GMT
Organization: University of Helsinki
Hello everybody out there using minix -
I'm doing a (free) operating system (just a hobby, won't be big and
professional like gnu) for 386(486) AT clones. This has been brewing
since april, and is starting to get ready. I'd like any feedback on things
people like/dislike in minix, as my OS resembles it somewhat (same
physical layout of the file-system (due to practical reasons) among
other things).
I've currently ported bash(1.08) and gcc(1.40), and things seem to
work. This implies that I'll get something practical within a few
months, and I'd like to know what features most people would want.
Any suggestions are welcome, but I won't promise I'll implement
them :-)
Linus (torvalds@kruuna.helsinki.fi)
PS. Yes - it's free of any minix code, and it has a multi-threaded fs.
It is NOT portable (uses 386 task switching etc), and it probably
never will support anything other than AT-harddisks, as that's all
I have :-(.
```

Trong thư này Linus cho biết anh đang phát triển một hệ điều hành tự do cho các máy tính đời 386 (486) và yêu cầu những ai quan tâm cho biết những thành phần nào của hệ thống cho người dùng cần phải có đầu tiên. Những người dùng trong nhóm tin tức này đã làm việc dưới hệ điều hành Minux do giáo sư Andy

[2] protocol

Tanenbaum viết ra để làm công cụ học tập cho các sinh viên lập trình. Minux làm việc trên các máy tính với bộ xử lý 286 và được Linus dùng làm mô hình cho HĐH mới.

Tập tin phiên bản đầu tiên của Linux (phiên bản 0.01) được công bố trên Internet ngày 17 tháng 09 năm 1991. Linus Torvalds viết: "As I already mentioned, 0.01 didn't actually come with any binaries: it was just source code for people interested in what linux looked like. Note the lack of announcement for 0.01: I wasn't too proud of it, so I think I only sent a note to everybody who had shown interest." ("Như tôi đã nói trước đây, 0.01 không đi kèm theo binary nào: nó chỉ là mã nguồn cho những ai muốn biết linux trông ra sao. Chú ý rằng không có thông báo cho bản 0.01: tôi không tự hào lắm về nó, vì thế chỉ gửi thông báo đến tất cả những ai muốn thể hiện sự quan tâm.")[3]

Sau đó ngày 05 tháng 10 năm 1991 phiên bản 0.02 ra đời. Đây là phiên bản đã có thể làm việc trên máy. Nếu bạn đọc quan tâm đến lịch sử của HĐH này thì hãy đọc trang web sau: http://www.li.org/linuxhistory.php. Ở đó bạn sẽ nhận được thông tin chi tiết về lịch sử xuất hiện và phát triển Linux.

Linus Torvalds không đăng ký bằng sáng chế cũng như không giới hạn việc phân phối HĐH mới này. Ngay từ đầu Linux đã được phân phối theo điều kiện của bản quyền General Public License (GPL)[4] thường dùng cho các phần mềm ứng dụng Open Source và dự án GNU. Theo tiếng lóng của Linux thì bản quyền này đôi khi được gọi là Copyleft. Về bản quyền này, Open Source và dự án GNU cần phải nói đến một cách đặc biệt.

Vào năm 1984 nhà bác học người mỹ Richard Stallman sáng lập ra Tổ chức phần mềm tự do (FSF, **F**ree **S**oftware **F**oundation) có trang chủ nằm tại địa chỉ http://www.fsf.org. Mục đích của tổ chức này là loại trừ tất cả những điều cấm đoán và hạn chế phân phối, sao chép, sửa đổi, nghiên cứu chương trình ứng dụng. Bởi vì tính đến thời điểm bắt đầu xây dựng tổ chức thì các công ty thương mại giữ rất cẩn thận các chương trình ứng dụng của mình, bảo vệ nó bằng các bằng sáng chế, các dấu hiệu bảo vệ quyền tác giả, giữ bí mật nghiêm ngặt mã nguồn của chương trình viết trên các ngôn ngữ bậc cao (như C++). Stallman cho rằng việc này rất có hại đối với phát triển chương trình ứng dụng, dẫn đến việc giảm chất lượng chương trình và sự có mặt của rất nhiều lỗi không xác định được trong những chương trình này. Tồi tệ nhất là làm chậm quá trình trao đổi ý tưởng trong ngành lập trình, làm chậm quá trình tạo ứng dụng mới vì mỗi nhà lập trình sẽ phải viết lại từ đầu một ứng dụng thay vì dùng đoạn mã nguồn đã có trong ứng dụng khác.

Trong khuôn khổ Tổ chức phần mềm tự do đã bắt đầu làm việc dự án GNU (http://www.gnu.org) – dự án tạo chương trình ứng dùng miễn phí. GNU là từ viết tắt của GNU's Not Unix, tức là những gì thuộc về dự án GNU không phải là một phần của Unix (vào thời gian đó thậm chí từ UNIX đã trở thành thương

[3] Người dịch: Rất thú vị là sau khi Linus Torvalds phát triển HĐH của mình thì giữa anh và giáo sư Andy Tanenbaum đã nổ ra một cuộc tranh cãi. Nếu bạn đọc quan tâm thì có thể tìm đọc những thư mà hai người này gửi cho nhau trong nhóm tin tức nói trên, hoặc tìm đọc cuốn "Linux: Just for fun", một cuốn sách nói về đời tư của Linus Torvalds đến thời điểm anh làm cho Transmeta và việc phát triển HĐH Linux.

[4] Người dịch: Thật ra lúc đầu nhân Linux được phân phối theo bản quyền mà FSF không không nhận là tự do vì nghiêm cấm phân phối thương mại. Bản quyền này có thể tìm thấy trong những phiên bản đầu tiên của nhân trên ftp.kernel.org, ví dụ ftp://ftp.kernel.org/pub/linux/kernel/Historic/old-versions/RELNOTES-0.01. Bản quyền được đổi sang GPL từ phiên bản 0.12, hãy xem RELNOTES-0.12 theo địa chỉ ở trên.

hiệu, do đó không còn tự do). Trong "Manifesto GNU" (http://www.gnu.org/gnu/manifesto.html) vào năm 1985 Stallman viết rằng động lực để ông sáng lập ra FSF và dự án GNU đó là sự khó chịu trong quyền sở hữu của một số người đối với chương trình ứng dụng.

Những gì do dự án GNU phát triển đề là *tự do*, nhưng không có nghĩa là chúng được phân phối không có bản quyền và không được luật pháp bảo vệ. Những chương trình Open Source (Mã nguồn mở) được phân phối theo điều kiện của bản quyền General Public License (GPL). Bạn có thể đọc bản quyền này theo địa chỉ http://www.gnu.org/copyleft/gpl.html. Bản dịch tiếng Việt không chính thức nằm tại http://vi.openoffice.org/gplv.html. Nếu như nói một cách thật ngắn gọn thì bản chất của GPL như sau. Chương trình ứng dụng phân phối theo GPL được quyền phát triển, sửa đổi, chuyển hoặc bán cho người khác không hạn chế với một điều kiện là kết quả thu được cũng phải phân phối theo bản quyền copyleft. Điều kiện cuối là quan trọng và then chốt của bản quyền này. Nó bảo đảm rằng kết quả lao động của các nhà phát triển phần mềm tự do sẽ luôn luôn mở và không trở thành một phần của sản phẩm nào đó dùng bản quyền bình thường (ý nói sản phẩm đóng). Điều kiện này cũng phân biệt phần mềm *tự do* với phần mềm phân phối *miễn phí*. Nói như các nhà sáng lập ra FSF, thì bản quyền GPL "làm cho chương trình ứng dụng tự do và đảm bảo là chương trình này sẽ tự do"[5].

Gần như tất cả các chương trình ứng dụng phân phối theo điều kiện GPL có thể coi là miễn phí đối với người dùng (trong phần lớn các trường hợp để nhận được nó bạn chỉ phải trả tiền đĩa CD, DVD hoặc kết nối Internet). Điều đó không có nghĩa là các nhà lập trình không còn nhận được phần thưởng (tiền) cho công việc của mình. Ý tưởng chính của Stallman là ở chỗ không phải bán chương trình ứng dụng, mà bán chính *sức lao động* của nhà lập trình. Ở đây cần phải đưa ra ví dụ để bạn đọc hiểu rõ hơn: nguồn thu nhập có thể là các sản phẩm đi kèm hoặc dịch vụ cài đặt và cấu hình cho những máy tính mới hoặc phát triển cho những điều kiện làm việc mới, dạy cách sử dụng, v.v... Một phần thưởng tốt nữa đó là khi chương trình trở lên nổi tiếng thì tác giả của chương trình sẽ có điều kiện tìm một công việc có lương cao. Các nhà phát triển `xvnkb` (http://xvnkb.sf.net), `unikey` (http://unikey.org) và `pdfLaTeX` (http://www.tug.org), là những người hiểu rõ nhất điều này. Hãy viết thư cho họ để học hỏi kinh nghiệm!

Trong khuôn khổ của hoạt động Open Source nói chung và dự án GNU nói riêng, đã phát triển một lượng đáng kể các chương trình ứng dụng, nổi tiếng nhất trong số chúng đó là trình soạn thảo **Emacs** và trình biên dịch GCC (GNU C Compliler) – trình biên dịch ngôn ngữ C tốt nhất hiện nay. Việc mở mã nguồn đồng thời nâng cao rất nhiều chất lượng của chương trình ứng dụng: tất cả những gì tốt nhất, những ý tưởng và cách giải quyết mới được phân phối rộng rãi ngay lập tức, còn các lỗi sẽ được nhận ra và sửa nhanh chóng. Ở đây chúng ta gặp lại cơ chế đào thải (hay tốt hơn là chọn lọc) tự nhiên như trong thuyết sinh học của Darwin. Cơ chế này bị kìm nén trong thế giới chương trình ứng dụng thương mại.

[5]Người dịch: Bạn đọc cũng nên biết là sắp tới sẽ có phiên bản thứ 3 của GPL (GPLv3). Cùng với sự ra đời của phiên bản thứ 3 này đã nảy ra rất nhiều tranh cãi xung quanh tính tự do của bản quyền. Tham gia vào tranh cãi có cả người viết ra nhân Linux đầu tiên – Linus Torvalds.

Tuy nhiên bây giờ xin quay lại với lịch sử của Linux. Cần nói rằng Linus Torvalds chỉ phát triển phần *nhân* (kernel) của hệ điều hành. Nhân này "đậu" đúng vào miền "đất lành", vì trong dự án GNU đã phát triển số lượng lớn các tiện ích khác nhau. Nhưng để chuyển GNU thành một HĐH hoàn chỉnh thì chỉ còn thiếu nhân. Dự án GNU cũng đã bắt đầu phát triển nhân cho riêng mình (được gọi là Hurd), nhưng vì lý do nào đó đã bị chậm lại. Vì thế sự xuất hiện của nhân Linux là rất đúng lúc. Nó đồng nghĩa với việc ra đời của một hệ điều hành mới tự do phân phối cùng với mã nguồn mở. Stallman tất nhiên đã đúng khi đòi hỏi hệ điều hành Linux phải được gọi là GNU/Linux. Nhưng đã thành lệ người dùng thường sử dụng tên gọi của nhân làm tên gọi của hệ điều hành, và chúng ta cũng làm như vậy trong cuốn sách này.

1.1.3 Đặc điểm chính của HĐH Linux

Do mã nguồn Linux phân phối tự do và miễn phí, nên ngay từ đầu đã có rất nhiều nhà lập trình tham gia vào quá trình phát triển hệ thống. Nhờ đó đến thời điểm hiện nay Linux là hệ điều hành hiện đại, bền vững và phát triển nhanh nhất, hỗ trợ các công nghệ mới gần như ngay lập tức. Linux có tất cả các khả năng, đặc trưng cho các hệ điều hành đầy đủ tính năng dòng UNIX. Xin đưa ra đây danh sách ngắn gọn những khả năng này.

1. **Nhiều tiến trình thật sự**
   Tất cả các tiến trình là độc lập, không một tiến trình nào được cản trở công việc của tiến trình khác. Để làm được điều này nhân thực hiện chế độ phân chia thời gian của bộ xử lý trung tâm, lần lượt chia cho mỗi tiến trình một khoảng thời gian thực hiện. Cách này hoàn toàn khác với chế độ "nhiều tiến trình đẩy nha" được thực hiện trong Windows 95, khi một tiến trình phải nhường bộ xử lý cho các tiến trình khác (và có thể làm chậm trễ rất lâu việc thực hiện).

2. **Truy cập nhiều người dùng**
   Linux không chỉ là HĐH nhiều tiến trình, Linux hỗ trợ khả năng nhiều người dùng làm việc cùng lúc. Khi này Linux có thể cung cấp tất cả các tài nguyên hệ thống cho người dùng làm việc qua các terminal ở xa khác nhau.

3. **Swap bộ nhớ lên đĩa**
   Swap bộ nhớ cho phép làm việc với Linux khi dung lượng bộ nhớ có hạn. Nội dung của một số phần (trang) bộ nhớ được ghi lên vùng đĩa cứng xác định từ trước. Vùng đĩa cứng này được coi là bộ nhớ phụ thêm vào. Việc này có làm giảm tốc độ làm việc, nhưng cho phép chạy các chương trình cần bộ nhớ dung lượng lớn mà thực tế không có trên máy tính.

4. **Tổ chức bộ nhớ theo trang**
   Hệ thống bộ nhớ Linux được tổ chức ở dạng các trang với dung lượng 4K. Nếu bộ nhớ đầy, thì HĐH sẽ tìm những trang bộ nhớ đã lâu không được sử dụng để chuyển chúng từ bộ nhớ lên đĩa cứng. Nếu có trang nào đó trong số những trang này lại trở thành cần thiết, thì Linux sẽ phục hồi chúng từ đĩa cứng (vào bộ nhớ). Một số hệ thống Unix cũ và một số hệ thống hiện đại

(bao gồm cả Microsoft Windows) chuyển lên đĩa tất cả nội dung của bộ nhớ thuộc về những ứng dụng không làm việc tại thời điểm hiện thời (tức là TẤT CẢ các trang bộ nhớ thuộc về ứng dụng sẽ được lưu lên đĩa khi không đủ bộ nhớ) và như vậy kém hiệu quả hơn.

5. **Nạp môđun thực hiện "theo yêu cầu"**
   Nhân Linux hỗ trợ việc cung cấp các trang bộ nhớ theo yêu cầu, khi này chỉ phần mã cần thiết của chương trình mới nằm trong bộ nhớ, còn những phần mã không sử dụng tại thời điểm hiện tại thì nằm lại trên đĩa.

6. **Cùng sử dụng chương trình**
   Nếu cần chạy một lúc nhiều bản sao của cùng một ứng dụng nào đó[6], thì Linux chỉ nạp vào bộ nhớ một bản sao của mã chương trình và tất cả các tiến trình giống nhau cùng sử dụng một mã này.

7. **Thư viện chung**
   Thư viện – bộ các quá trình (thao tác) được chương trình dùng để làm việc với dữ liệu. Có một số thư viện tiêu chuẩn được dùng cùng lúc cho vài tiến trình. Trên các hệ thống cũ những thư viện đó nằm trong mỗi tập tin chương trình, và thực hiện cùng lúc những chương trình này dẫn đến hao hụt bộ nhớ không đáng có. Trên các hệ thống mới (bao gồm Linux) có hỗ trợ làm việc với các thư viện động (dynamic) và tĩnh (static) được chia ra, và như vậy cho phép giảm kích thước bộ nhớ bị ứng dụng chiếm.

8. **Bộ đệm động của đĩa**
   Bộ đệm của đĩa đó là một phần bộ nhớ của hệ thống dùng làm nơi lưu những dữ liệu thường dùng của đĩa, nhờ đó nâng cao rất nhiều tốc độ truy cập tới những chương trình và tiến trình thường dùng. Người dùng MS-DOS sẽ nhớ đến chương trình SmartDrive, chương trình này dự trữ một phần bộ nhớ có kích thước xác định để làm bộ đệm cho đĩa. Linux sử dụng hệ thống đệm linh động hơn: bộ nhớ được dự trữ cho đệm được tăng lên khi bộ nhớ không được sử dụng, và sẽ giảm xuống khi hệ thống hay tiến trình cần nhiều bộ nhớ hơn.

9. **100% tương ứng với tiêu chuẩn POSIX 1003.1. Hỗ trợ một phần các khả năng của System V và BSD**
   POSIX 1003.1 (Portable Operating System Interface – giao diện của hệ điều hành lưu động) đưa ra giao diện tiêu chuẩn cho các hệ thống Unix, đó là một bộ các thủ tục ngôn ngữ C. Ngày nay giao diện này được tất cả các hệ điều hành mới hỗ trợ. Microsoft Windows NT cũng hỗ trợ POSIX 1003.1. Linux 100% tương ứng với tiêu chuẩn POSIX 1003.1. Thêm vào đó Linux còn hỗ trợ các khả năng của System V và BSD để tăng tính tương thích.

10. **System V IPC**
    Linux sử dụng công nghệ IPC (**I**nter**P**rocess **C**ommunication) để trao đổi thông tin giữa các tiến trình, để sử dụng tín hiệu và bộ nhớ chung.

[6] hoặc một người dùng chạy vài tiến trình giống nhau, hoặc nhiều người dùng chạy cùng một chương trình

11. **Khả năng chạy chương trình của HĐH khác**
Trong lịch sử Linux không phải là hệ điều hành đầu tiên. Người ta đã viết ra hàng loạt các chương trình ứng dụng, trong đó có cả những chương trình có ích và không đến nỗi tồi, cho các HĐH đã phát triển trước Linux, bao gồm DOS, Windows, FreeBSD và OS/2. Để chạy những chương trình như vậy dưới Linux đã phát triển các *trình giả lập* (emulator) cho DOS, Windows 3.1, Windows 95 và Wine. Ngoài ra, còn có một loạt các chương trình tạo máy ảo[7] mã nguồn mở cũng như sản phẩm thương mại: qemu, bochs, pearpc, vmware,... HĐH Linux còn có khả năng chạy chương trình dành cho bộ xử lý Intel của các hệ thống Unix khác, nếu hệ thống đáp ứng tiêu chuẩn iBCS2 (intel Binary Compatibility).

12. **Hỗ trợ các định dạng hệ thống tập tin khác nhau**
Linux hỗ trợ một số lượng lớn các định dạng hệ thống tập tin, bao gồm các hệ thống tập tin DOS và OS/2, và cả các hệ thống tập tin mới, như reiserfs, HFS,.... Trong khi đó hệ thống tập tin chính của Linux, được gọi là Second Extended File System (ext2fs) và Third Extended File System (ext3fs) cho phép sử dụng không gian đĩa một cách có hiệu quả.

13. **Khả năng hỗ trợ mạng**
Linux có thể gắn vào bất kỳ mạng nội bộ nào. Hỗ trợ tất cả các dịch vụ Unix, bao gồm **N**etworked **F**ile **S**ystem (NFS), kết nối từ xa (telnet, rlogin, ssh), làm việc trong các mạng TCP/IP, truy cập dial-up qua các giao thức SLIP và PPP,v.v... Đồng thời có hỗ trợ dùng Linux là máy chủ hoặc máy khách cho mạng khác, trong đó có chia sẻ (dùng chung, sharing) các tập tin và in từ xa trong các mạng Macintosh, NetWare và Windows.

14. **Làm việc trên các phần cứng khác nhau**
Mặc dù đầu tiên HĐH Linux được phát triển cho máy tính cá nhân (PC) trên nền tảng Intel 386/486, bây giờ nó có thể làm việc trên tất cả các bộ vi xử lý Intel bắt đầu từ 386 và kết thúc là các hệ thống nhiều bộ xử lý Pentium IV, bao gồm cả các bộ xử lý 64bit. Đồng thời Linux còn làm việc trên rất nhiều bộ xử lý tương thích với Intel của các nhà sản xuất khác, như AMD. Trong Internet còn có những thông báo nói rằng trên các bộ xử lý Athlon và Duron của AMD Linux còn làm việc tốt hơn so với trên Intel. Ngoài ra còn có phiên bản Linux cho các bộ xử lý khác bao gồm ARM, DEC Alpha, SUN Sparc, M68000 (Atari và Amiga), MIPS, PowerPC và những bộ xử lý khác[8]. Xin được nói luôn là trong cuốn sách này chúng ta chỉ xem xét trường hợp Linux cho các máy tính tương thích với IBM.

1.2 Bản phân phối Linux

Trong bất kỳ hệ điều hành nào cũng có thể chia ra 4 phần chính: nhân, cấu trúc (hệ thống) tập tin, trình dịch lệnh người dùng và các tiện ích. *Nhân* đó là

[7] cho phép sử dụng nhiều hệ điều hành trên một máy

[8] Người dịch: bản phân phối Linux hỗ trợ nhiều bộ xử lý nhất cần phải kể đến Debian (http://www.debian.org)

thành phần chính, nòng cốt của HĐH, nó điều khiển các thiết bị phần cứng và điều khiển việc thực hiện chương trình. *Cấu trúc tập tin* (hệ thống tập tin) – là hệ thống lưu tập tin trên các thiết bị lưu. *Trình dịch lệnh* hay *hệ vỏ* (shell) – là chương trình tổ chức giao tiếp giữa máy tính và người dùng. Và cuối cùng các *tiện ích* – đó đơn giản là các chương trình riêng lẻ, nói chung không khác so với những chương trình bình thường khác mà người dùng có thể chạy, nhưng có chức năng chính là thực hiện các công việc dịch vụ (service).

Như đã nói ở trên, nếu chính xác thì từ "Linux" chỉ có nghĩa là nhân. Vì thế khi nói về hệ điều hành nói chính xác hơn sẽ là "hệ điều hành dựa trên nhân Linux". Nhân của HĐH Linux hiện thời đang được phát triển dưới sự lãnh đạo của Linus Torvalds và phân phối một cách tự do (với bản quyền GPL) giống như một số lượng khổng lồ các chương trình ứng dụng và tiện ích khác. Một trong những kết quả của việc phân phối tự do chương trình ứng dụng cho Linux đó là có nhiều công ty cũng như nhóm các nhà phát triển độc lập đã phát hành ra các bản Linux khác nhau được gọi là "bản phân phối Linux".

Bản phân phối – đó là một bộ các chương trình ứng dụng bao gồm tất cả bốn phần chính của HĐH, tức là nhân, hệ thống tập tin, hệ vỏ shell và các tiện ích, đồng thời còn có thêm các chương trình cho công việc hàng ngày của người dùng. Thông thường tất cả những chương trình nằm trong bản phân phối Linux đều dùng bản quyền GPL. Rất có thể xuất hiện trong bạn đọc ý nghĩ rằng bất kỳ ai cũng có khả năng cho ra bản phân phối Linux, hay nói đúng hơn là bất kỳ người nào không lười sưu tập các chương trình tự do. Và suy nghĩ đó có phần nào đúng. Tuy nhiên các nhà phát triển của một bản phân phối Linux cần tạo ra ít nhất một chương trình cài đặt để đưa HĐH lên máy tính trống không chưa có HĐH nào. Ngoài ra, cần tìm cách giải quyết sự phụ thuộc và mâu thuẫn giữa các gói (và giữa các phiên bản của gói) chương trình. Và như chúng ta sẽ thấy ở sau đó không phải là bài toán đơn giản.

Tuy vậy, trên thế giới đang có hàng trăm (hàng nghìn?) bản phân phối Linux và mỗi ngày lại xuất hiện các bản mới. Có thể tìm thấy danh sách tương đối đầy đủ cùng với đặc điểm ngắn gọn của mỗi bản phân phối trên `http://www.linuxhq.com` (còn có một số bản phân phối khác tiếng Anh). Ngoài ra, trên trang đó cũng như nhiều trang web khác còn có liên kết đến những danh sách bản phân phối khác, vì thế nếu muốn có thể tìm thấy tất cả những gì có trên thế giới (hầu hết những trang này dùng tiếng Anh và các bản phân phối Việt Nam ít được nói đến).

Một vài tác giả đã thử phân loại các bản phân phối dựa trên những tiêu chí khác nhau:

- cấu trúc hệ thống tập tin
- chương trình cài đặt
- phương tiện dùng để cài đặt các gói chương trình
- thành phần của các tiện ích và chương trình ứng dụng có trong bản phân phối.

Mặc dù hầu hết các tác giả cho rằng sự khác nhau giữa các bản phân phối là không cơ bản. Nhưng hiện nay có thể chia ít nhất 3 nhóm bản phân phối, mà đại diện của mỗi nhóm là Red Hat, Slackware và Debian.

Vậy thì cần lựa chọn bản phân phối theo tiêu chuẩn nào? Theo ý kiến của tác giả thì đối với người dùng Việt Nam có hai tiêu chuẩn: thứ nhất phải có giao diện người dùng tiếng Việt và thứ hai phải có một nhóm các nhà phát triển hỗ trợ bản phân phối này. Và tốt nhất nếu nhóm các nhà phát triển này nhận được nguồn lợi từ sản phẩm Linux của mình, tức là làm việc như một công ty thương mại. Thậm chí chỉ trong khoảng thời gian không lâu mà tôi sử dụng Linux (khoảng 5 năm) đã có nhiều bản phân phối Việt Nam cũng như nước ngoài đã chào tạm biệt thế giới Linux vì nhóm hỗ trợ của chúng không thu được nhiều lợi nhuận và sau đó một thời gian ngừng hỗ trợ tác phẩm của mình.

Đối với người dùng Linux Việt Nam hiện thời có hai lựa chọn: thứ nhất, sử dụng các sản phẩm Linux tiếng Việt do một số người tự nguyện duy trì; thứ hai, sử dụng các sản phẩm Linux lớn có hỗ trợ tiếng Việt. Chúng ta sẽ nói đến hai sự lựa chọn này một cách kỹ càng hơn. Lựa chọn thứ nhất, theo ý kiến của tôi không được ưu tiên ở đây. Lý do chính là chưa đạt được tiêu chuẩn thứ hai nêu trên. Hiện thời có một bản phân phối như vậy: vnlinux-CD (cùng với một vài biến thể) do anh Larry Nguyễn, một Việt kiều ở Mỹ duy trì. Lựa chọn thứ hai đó là sử dụng các bản phân phối hỗ trợ tiếng Việt tốt như Debian, Ubuntu và Mandriva, openSuSE cũng đang dần dần hỗ trợ tiếng Việt (http://vi.opensuse.org). Lựa chọn thứ hai này được ưu tiên vì:

- chúng có hỗ trợ tiếng Việt và càng ngày càng hoàn thiện
- chúng có trình cài đặt tốt, hỗ trợ nhiều phần cứng khác nhau
- có thể cài đặt thêm các phần mềm khác một cách dễ dàng phần lớn chương trình đã được biên dịch sẵn cho mọi yêu cầu của người dùng
- những bản phân phối này được một nhóm các nhà phát triển duy trì, cập nhật thường xuyên thông thường là theo một lịch định sẵn. Ngoài ra còn đảm bảo là bạn sẽ nhận được phiên bản mới của nó trong tương lai. Không sợ trường hợp "đem con bỏ chợ".

Để kết thúc câu chuyện về lựa chọn bản phân phối Linux cần nói thêm là gần đây bác Nguyễn Đại Quý (http://vnoss.org) đã cho ra đời một bản phân phối mới hướng về người dùng Việt Nam – FCxVnOSS. Bản phân phối này dựa trên nền tảng của FC cộng thêm giao diện tiếng Việt và một số ứng dụng "mang tính Việt Nam" như chương trình gõ tiếng Việt, từ điển tiếng Việt,...

Cần nói vài lời về đánh số phiên bản. Cần phân biệt số phiên bản của **bản phân phối** và số phiên bản của **nhân**. Khi nói đến phiên bản của Linux thường có ý là phiên bản nhân (vì một hệ điều hành là Linux chỉ khi nó sử dụng nhân Linux). Vì Linus Torvalds tiếp tục điều hành việc phát triển nhân, nên phiên bản của nhân tăng lên theo thứ tự, chứ không phân nhánh và nhân lên giống như trường hợp bản phân phối.

Phiên bản nhân Linux thường được ký hiệu bằng ba số[9], phân cách nhau bởi dấu chấm. Ví dụ, bản phân phối openSuSE Linux 10.1 được dựa trên nhân phiên bản 2.6.16.13, tức là Linux phiên bản 2.6.16.13. Phiên bản nhân với số thứ hai

[9]Người dịch: Điều này chỉ đúng với các phiên bản nhân trước 2.6. Từ 2.6 trở đi Linus và các nhà phát triển khác thử nghiệm dùng bốn số.

lẻ (ví dụ, 2.5.0) thường không được sử dụng để tạo các bản phân phối, vì đó là phiên bản thử nghiệm (chỉ dành cho phát triển). Chúng được dành cho những người tình nguyện có mong muốn thử nghiệm để tìm ra các lỗi. Tất nhiên phiên bản như vậy có thể làm việc, nhưng không bền vững. Phiên bản với số thứ hai chẵn (ví dụ 2.6.16.13) được coi là làm việc ổn định. Tất nhiên là bạn có thể cài đặt bất kỳ phiên bản nào, nhưng đối với người dùng mới thì nên chọn phiên bản nhân với số thứ hai trong phiên bản là chẵn. Nếu cài đặt một bản phân phối đầy đủ thì đương nhiên lựa chọn nhân đã được các nhà phát triển làm giùm bạn đọc, nhưng cần biết cách đánh số phiên bản nếu khi nào đó bạn muốn cập nhật nhân Linux của mình.

1.3 Yêu cầu đối với máy tính

Tôi đã đọc ở đâu đó nói rằng có những phiên bản Linux đặc biệt, làm việc thậm chí trên bộ xử lý 8086 với 512Kbyte bộ nhớ, còn phiên bản đặc biệt có thể chạy từ một hoặc hai đĩa mềm không cần đĩa cứng thì tôi đã gặp.

Vì thế nếu bạn có một cái máy tính cũ, trên đó không thể chạy nổi Windows, thì có thể sử dụng nó để học Linux và rất có thể sẽ ngạc nhiên về khả năng của HĐH này. Nhưng trong cuốn sách này chúng ta sẽ không xem xét những trường hợp đặc biệt như vậy. Vì HĐH Linux sử dụng chế độ bảo vệ của bộ vi xử lý, nên để cài đặt HĐH này cần ít nhất là một máy tính có bộ xử lý 386. Theo các nguồn thông tin khác nhau thì tất cả các biến thể đều dùng tốt: SX, DX v.v... Xin đừng lo lắng, những máy tính sản xuất gần đây đáp ứng được toàn bộ những yêu cầu đã đưa ra và sẽ đưa ra dưới đây. Yêu cầu đối với phần cứng của hệ thống muốn cài đặt Linux còn được xác định bởi lựa chọn phần mềm của người dùng (và tức là sẽ phụ thuộc vào phiên bản của các phần mềm và ít nhiều vào bản phân phối). Bảng 1.1 dưới đây sẽ đưa ra một vài con số chỉ với mục đích giúp bạn đọc làm quen, những con số này là không chính xác[10] nhưng không khác biệt nhiều giữa các bản Linux khác nhau.

Bảng 1.1: Yêu cầu đối với phần cứng

Mong muốn của người dùng	**Yêu cầu**, MB	
	Bộ nhớ	**Đĩa cứng**
Yêu cầu nhỏ nhất: chỉ làm việc trong giao diện văn bản với dòng lệnh của shell, có đủ một số ứng dụng người dùng như vim, emacs,...	8	200
Dùng được giao diện đồ hoạ X Window cùng với một số trình quản lý cửa sổ nhỏ nhẹ như icewm, fluxbox, windowmaker.	32	400
Dùng môi trường làm việc đồ hoạ KDE.	128	1000
Chạy các ứng dụng cần nhiều bộ nhớ (như GIMP, các ứng dụng nằm trong KOffice, OpenOffice.org).	256	1500

Như vậy Linux có một ưu điểm lớn đó là khả năng làm việc thậm chí trên

[10]Người dịch: bản gốc đưa ra một bản đã quá cũ, tôi xin đưa ra một bản mới dựa trên cơ sở của openSuSE 10.1.

những máy rất cũ, mà trước đây chỉ có thể dùng MS DOS (tất nhiên là trong trường hợp này chúng ta chỉ thu được chế độ dòng lệnh, nhưng điều này không ngăn cản việc dùng các máy tính cũ làm việc có lợi ví dụ làm router[11]). Để bắt đầu học Linux thì chỉ cần có một máy tính với bộ xử lý 486, 16MB bộ nhớ và ổ cứng khoảng 300MB. Tất nhiên đối với dung lượng bộ nhớ cũng như cấu hình máy nói chung thì máy càng mạnh, càng nhiều bộ nhớ, ổ cứng càng rộng thì càng tốt. Không có gì là thừa thãi.

Người dịch: Để kết thúc xin đưa ra đây trường hợp cấu hình máy của tôi: bộ xử lý 686, bộ nhớ 256Mb (+256Mb swap), 10GB ổ cứng dành cho cài đặt Linux, phần còn lại dành cho dữ liệu. Tôi hiện sử dụng openSuSE Linux 10.1 với môi trường làm việc KDE. Có thể làm việc đồng thời nhiều ứng dụng yêu cầu: OpenOffice.org, KBabel, Kile (dùng để viết những dòng này), StarDict (chương trình từ điển), Konqueror, GIMP. Nếu máy tính của bạn có thể làm việc với các hệ điều hành Windows 2000 và Windows XP thì việc chạy Linux trên nó sẽ không có gì khó khăn.

1.4 Lấy Linux ở đâu?[12]

Và trong phần cuối của chương thứ nhất chúng ta sẽ trả lời ngắn gọn cho câu hỏi "Lấy Linux ở đâu?".

Như đã nói ở trên, Linux cùng với một số lượng khổng lồ các chương trình ứng dụng được phân phối gần như miễn phí. Có nghĩa là người dùng không có ý định thay đổi chương trình hoặc mua bán những chương trình này, thì có toàn quyền sao chép toàn bộ bản phân phối Linux hoặc một phần bất kỳ của nó ở chỗ người quen, hoặc tải xuống từ Internet hoặc mua đĩa CD (DVD) Linux ở chỗ những người bán hàng đâu đó trên hè phố mà không sợ bị truy cứu vì vi phạm bản quyền (các chương trình có bản quyền thương mại thường dùng từ "sự đồng ý" giữa người dùng và nhà phân phối) của các nhà (công ty) phát triển.

Trong số ba phương án kể trên thì đối với người dùng Linux Việt Nam phương án mua CD, DVD là tốt nhất. Cần nói thêm là những đĩa ghi vỉa hè thường có lỗi và có thể gây mất dữ liệu hoặc làm hỏng phần cứng. Tốt hơn hết là nên mua đĩa của một công ty máy tính hoặc qua một cửa hàng trên mạng. Khi đó còn có khả năng lựa chọn và có bảo đảm là sẽ đổi được đĩa xấu. Tất nhiên là chênh lệnh giá thành đĩa phải không quá cao, giá thành bán Linux bao nhiêu đó là quyền của người bán (ngoài ra giá thành cao còn có ở các bản phân phối chuyên nghiệp như Xandros, Novell Desktop,...).

Hiện giờ trên mạng Việt Nam đã xuất hiện một vài cửa hàng bán đĩa Linux. Địa chỉ cụ thể xin không đưa ra ở đây vì quảng cáo trong thế giới hiện đại đã không còn là miễn phí. Và việc giấu địa chỉ không làm ảnh hưởng đến những ai có mong muốn mua Linux để nghiên cứu.

Cần nói riêng về hệ thống phát đĩa với bản phân phối Ubuntu một cách miễn phí. Chỉ cần vào địa chỉ `http://shipit.ubuntu.com` đăng ký số lượng đĩa bạn cần, tất nhiên có kèm theo địa chỉ bưu điện, sau một thời gian khoảng 1 tháng –

[11] máy giúp chuyển hướng các gói mạng
[12] Phần này do người dịch viết

1 tháng rưỡi bạn sẽ nhận được chúng. Theo tôi nghĩ đây là cách tốt nhất nếu bạn không có điều kiện để mua đĩa.

Chương 2

Cài đặt HĐH Linux trên cùng máy tính với Windows

"Software is like sex, it's good when it's free" – Linus Torvalds.

Thông thường trên các đĩa của bản phân phối Linux đã có hướng dẫn ngắn gọn cách cài đặt Linux. Ngoài ra, trên Internet bạn có thể tìm thấy rất nhiều cuốn sách nó về vấn đề này. Và tất cả các bản phân phối lớn (Debian, Slackware, Fedora, Mandrake, ...) đều đã có cuốn hướng dẫn cài đặt rất chi tiết, cho mọi tình huống sử dụng. Hãy chờ đợi và hy vọng trong tương lai không xa sẽ có bản dịch Tiếng Việt của những cuốn sách này. Chính vì vậy trong cuốn sách này, tác giả sẽ không đưa ra các bước cụ thể của việc cài đặt, mà xin bạn đọc hãy tìm các cuốn hướng dẫn tương ứng. Thay vào đó là những gì bạn cần biết và chuẩn bị trước khi cài đặt, đồng thời, tác giả sẽ đi cụ thể và chi tiết vào những gì đặc biệt khi cài đặt Linux trên máy tính đã có một trong các hệ điều hành (HĐH) Windows cũng như việc khởi động nhiều hệ điều hành.

Vấn đề ở chỗ, phần lớn người dùng Việt Nam trước khi bắt đầu học Linux đã làm quen và rất có thể đã có kinh nghiệm sử dụng các HĐH dòng Windows như Windows 98, Windows 2000 và Windows XP. Và trên thực tế thì tạm thời Linux khó có thể là HĐH đầu tiên mà người dùng làm quen. Như thế, một cách tự nhiên, nếu người dùng đã làm việc với HĐH Windows và quyết định thử nghiệm với Linux, thì họ không muốn mất đi môi trường làm việc quen thuộc của mình, cùng với những gì đã tạo ra và đã cấu hình dưới dưới Windows. Rất may là không nhất thiết phải đánh mất tất cả những thứ đó. Bởi vì trên một máy tính có thể cùng "chung sống hòa bình" hai HĐH và thậm chí nhiều hơn nữa (nếu có đủ chỗ trên đĩa!). Chính vì thế, ở phía dưới sẽ nói cách cài đặt HĐH Linux trên máy tính đã cài đặt một trong các hệ điều hành của hãng Microsoft.

2.1 Chuẩn bị cài đặt

Có thể cài đặt Linux bằng một trong các cách sau:

- Từ ổ đĩa CD-ROM
- Từ bản sao chép Linux trên ổ đĩa cứng
- Từ máy chủ tập tin của mạng nội bộ qua NFS;

- Từ máy tính khác trong mạng nội bộ qua SMB;
- Từ máy tính ở xa (ví dụ từ Internet) qua giao thức FTP;
- Từ một máy chủ WWW qua giao thức HTTP.

Theo ý kiến cá nhân của tác giả thì thuận tiện và có tính thực tế nhất là cài đặt Linux từ CD-ROM, hơn nữa việc mua các đĩa CD bây giờ không gây khó khăn gì.

Trước khi bắt đầu cài đặt, hãy thu thập (hãy viết lên một tờ giấy) tất cả những thông tin cấu hình cần thiết của máy tính. Nếu như máy tính của bạn tạm thời vẫn còn làm việc dưới HĐH Windows 95/98/2000/XP, thì bạn sẽ tìm thấy rất nhiều thông tin nếu nhấn chuột phải vào biểu tượng My Computer, chọn lệnh Properties. Ở đây bạn có thể tìm thấy gần hết tất cả thông tin cần thiết. Nếu như bạn không thấy thông tin nào đó, thì cần tìm kiếm theo các cách khác, kể cả việc mở vỏ máy và đọc những dòng chữ trên thiết bị.

Để giúp bạn đọc, xin được đưa ra đây danh sách những thông tin cần thu thập. Xin đừng lười biếng và hãy cố gắng ghi càng nhiều dữ liệu về một thiết bị càng tốt (tất cả những thông tin có thể tìm thấy), những dữ liệu này sẽ cần đến khi cài đặt và cấu hình, khi mà việc tìm kiếm chúng sẽ khó khăn hơn.

- BIOS:
  - nhà sản xuất;
  - số hiệu phiên bản.
- Controller ổ đĩa cứng: loại (IDE hay SCSI) và dung lượng của ổ đĩa (nếu như bạn đọc dùng đĩa IDE, thì cần kiểm tra xem BIOS có hỗ trợ việc truy cập ở chế độ LBA hay không):
  - hda (Master trên controller số 1 hay Primary Master);
  - hdb (Slave trên controller số 1 hay Primary Slave);
  - hdc (Master trên controller số 2 hay Secondary Master);
  - hdd (Slave trên controller số 2 hay Secondary Slave).
  - nhà sản xuất và số mẫu mã của adapter SCSI (nếu có).
- Dung lượng của bộ nhớ (tính bằng Kilobyte
- CD-ROM:
  - Giao diện (IDE, SCSI, hay giao diện khác);
  - đối với các ổ đĩa CD-ROM không phải IDE, cũng như SCSI - nhà sản xuất và số mẫu mã.
- Chuột:
  - loại chuột (serial, PS/2, hay bus mouse);
  - giao thức (Microsoft, Logitech, MouseMan, v.v. . .);

  - số nút;
  - đối với chuột cắm vào cổng nối tiếp thì cần số thứ tự của cổng đó.

- Cạc màn hình
  - nhà sản xuất;
  - số mẫu mã (hay chipset sử dụng)
  - dung lượng bộ nhớ;

- Màn hình
  - nhà sản xuất
  - số mẫu mã;
  - các giá trị giới hạn (min, max) của tần số làm mới theo chiều dọc và theo chiều ngang (những giá trị này bạn đọc chỉ có thể tìm thấy trong tài liệu đi kèm với màn hình, Windows không hiển thị những giá trị này, và chúng rất quan trọng trong khi cấu hình giao diện đồ họa).

- Nếu như bạn đọc muốn kết nối mạng (mà UNIX nói chung là HĐH dành cho mạng), thì hãy ghi lại những dữ liệu sau:
  - nhà sản xuất và số mẫu mã cạc mạng;
  - địa chỉ IP của mình;
  - tên của máy tính trong mạng;
  - mặt nạ mạng con (subnet mask);
  - địa chỉ IP của gateway;
  - địa chỉ IP của các máy chủ tên miền (DNS server);
  - địa chỉ IP của máy chủ WINS(Windows Internet Name Service);
  - tên miền của công ty bạn đọc.

- Loại và nhà sản xuất cạc âm thanh và game controller (nếu như có)

2.2 Phòng xa và những lời khuyên

Trước khi cài đặt HĐH Linux sau Windows, rất nên thực hiện vài thao tác "phòng xa" (*"phòng cháy hơn chữa cháy"*). Vì rất có thể bạn đọc sẽ phải phân vùng lại ổ đĩa, thay đổi bản ghi khởi động (Boot Record) và làm việc với các tập tin khởi động cũng như các tập tin cấu hình. Các thao tác này không phải lúc nào cũng đem lại đem lại một kết quả theo ý muốn, và trong trường hợp xấu có thể máy tính của bạn đọc sẽ không khởi động nữa. Có biết cách thoát ra khỏi tình huống này và phục hồi dữ liệu cần thiết không đó còn là một câu hỏi. Nhưng rơi vào tình huống như vậy hết sức dễ dàng nhất là với người dùng lần đầu tiên cài đặt Linux. Chính vì vậy, đầu tiên, cần tạo một đĩa mềm khởi động hay một đĩa mềm giúp phục hồi hệ thống (nếu bạn đọc còn chưa tạo). Thứ hai, cần ghi lại những dữ liệu có giá trị (backup). Và thứ ba, chuẩn bị các tập tin (đĩa mềm, CD) cài đặt

cho hệ thống cũ. Một lời khuyên quan trọng khác: nếu có gì đó xảy ra không theo ý muốn thì không nên hoang mang. Xin chia sẻ một kinh nghiệm buồn: khi lần đầu tiên tác giả cài Linux trên máy tính đã có Windows NT, và kết quả là máy tính không thể khởi động được. Không hiểu hết vấn đề tác giả nghĩ là không còn cách gì khác ngoài định dạng lại ổ đĩa và cài đặt lại từ đầu. Bây giờ thì tác giả đã hiểu là có thể phục hồi lại nếu như không quyết định quá vội vàng. Vì vậy có thể nói rằng Werner Almesberger đúng, khi trong hướng dẫn sử dụng LILO có đưa ra những lời khuyên sau cho người dùng khi rơi vào trường hợp khó khăn như vậy:

- Không hoảng hốt. Nếu như có gì đó không làm việc, hãy thử mọi cách để tìm ra nguyên nhân, kiểm tra lại nhiều lần thao tác của mình. Chỉ sau khi đó mới thực hiện các bước sửa lỗi.

- Hãy đọc tài liệu. Đặc biệt trong các trường hợp, khi hệ thống làm những gì bạn đọc không mong đợi.

Xin thêm một lời khuyên phổ biến sau: * Hãy xem các tập tin log, tức là các tập tin ghi lại sự kiện của hệ thống (cần tìm chúng trong thư mục `/var/log`).

Như đã nói ở trên, quá trình cài đặt HĐH Linux nói riêng không phải là đề tài của cuốn sách. Người dùng cần tìm các cuốn hướng dẫn tương ứng. Tuy nhiên, tác giả cũng muốn đưa ra vài lời khuyên để giúp người dùng đưa ra quyết định trong khi cài đặt.

Thứ nhất, đừng vội vàng và hãy chú ý đọc những thông báo sẽ hiển thị trên màn hình, và hãy suy nghĩ kỹ khi chọn câu trả lời. Để minh chứng cho lời khuyên này xin được kể lại trường hợp khi tác giả cài Red Hat 7.1, và tự động nhấn lên nút Next, vì cho rằng phương án theo mặc định là đủ. Kết quả là tác giả không thể truy cập được đến máy này qua các giao thức mạng (telnet, ftp, NFS, Samba), mặc dù đã cấu hình giao diện mạng cho máy. Nguyên nhân là trong phương án theo mặc định thì tường lửa được cài đặt, và tường lửa đóng hết các truy cập từ mạng. Để mở truy cập này, thì trong quá trình cài đặt cần chỉ rõ các dịch vụ được mở. Nhưng chúng ta quá vội vàng! Thứ hai, tác giả khuyên không nên đồng ý với việc tự động khởi động vào giao diện đồ họa. Vì cuối cùng người dùng không khó khăn gì khi gõ câu lệnh `startx`, còn việc cấu hình giao diện đồ họa (nếu có gì đó làm việc không đúng) với người dùng mới rất khó thành công.

Sau khi làm xong các công việc phòng xa, cần quyết định sẽ tổ chức khởi động nhiều HĐH như thế nào, chuẩn bị các ổ đĩa (phân vùng) để cài đặt, tức là cần chia ổ đĩa thành số phân vùng cần thiết. Nhưng trước khi chuyển sang các bước cụ thể để chuẩn bị ổ đĩa, xin được nói qua một chút về cấu trúc của đĩa và quá trình khởi động HĐH. Nếu ai đó không đủ kiên nhẫn để đọc phần lý thuyết này, thì có thể bỏ qua chúng và chuyển thẳng đến vấn đề chọn chương trình khởi động.

2.3 Phân vùng trên đĩa và quá trình khởi động

2.3.1 Thế nào là cấu trúc "hình học của đĩa"

Như bạn đọc biết, đĩa cứng gồm vài đĩa có phủ lớp từ tính, nằm trên cùng một trục và quay với vận tốc lớn. Đọc/Ghi dữ liệu được thực hiện bởi các đầu đọc nằm giữa các đĩa này, di chuyển từ tâm đĩa ra rìa ngoài của đĩa. Vòng tròn đầu đọc vẽ ra trên các đĩa khi quay quanh chúng gọi là rãnh (track), còn tập hợp các rãnh nằm chồng lên nhau gọi là cylinder. Mỗi rãnh lại chia thành các sector, và có thể ghi vào mỗi sector 512 byte thông tin. Vì thế đặc điểm của một ổ đĩa thường là tập hợp ba số: số cylinder/số rãnh trong cylinder/số sector trên rãnh hay còn viết tắt là $C/H/S$ (ba chữ cái đầu tiên của các thuật ngữ Tiếng Anh tương ứng: Cyliner/Head/Sector). Ba số này gọi là cấu trúc *"hình học của đĩa"*. Đĩa với cấu trúc hình học C/H/S có dung lượng C*H*S*512 byte.

Đĩa cứng là các thiết bị khối, tứ là đọc và ghi thông tin theo các khối, và kích thước nhỏ nhất của khối bằng một sector (512 byte). Để có thể ghi thông tin lên đĩa, cần đặt đầu đĩa đúng vị trí, tức là chỉ cho controller biết cần ghi thông tin này vào sector nào. Sector được đánh địa chỉ theo số thứ tự cylinder, số thứ tự đầu đọc (hay rãnh) và số thứ tự sector trên rãnh.

2.3.2 Phân vùng và bảng phân vùng của đĩa

Trong các hệ thống Intel ổ đĩa thường được chia thành các phân vùng. Rất có thể nguyên nhân của việc phân vùng là nguyên nhân lịch sử: các phiên bản MS-DOS đầu tiên không thể sử dụng được các đĩa lớn, mà dung lượng đĩa lại phát triển nhanh hơn khả năng của DOS. Khi đó đã nghĩ ra việc chia ổ đĩa thành các phân vùng. Để làm được điều này, trong sector số 0 của đĩa (sector số 0 của rãnh đầu tiên trong cylinder số 0) ghi nhớ *bảng chia ổ đĩa thành các phân vùng* (partition table). Mỗi phân vùng được dùng như một đĩa vật lý riêng rẽ. Một trường hợp nói riêng đó là trong các phân vùng khác nhau có thể cài đặt các hệ điều hành khác nhau.

Bảng phân vùng chứa 4 bản ghi 16 byte cho 4 phân vùng chính. Mỗi bản ghi có cấu trúc như sau:

```
struct partition {
  char active;    /* 0x80: phân vùng kích hoạt, 0: không kích hoạt */
  char begin[3];  /* CHS sector đầu tiên, 24 bit
  char type;      /* loại phân vùng (ví dụ, 83 -- LINUX_NATIVE) */
  char end[3];    /* CHS sector cuối cùng, 24 bit */
  int start;      /* số của sector đầu tiên (32-bit, tính từ 0) */
  int length;     /* số sector có trong phân vùng (32 bit) */
};
```

Bảng phân vùng đĩa thường được tạo bởi chương trình `fdisk`. Trên HĐH Linux ngoài chương trình `fdisk` "truyền thống" (tuy vậy rất khác so với chương trình fdisk trong MS-DOS và Windows), còn có hai chương trình để làm việc với phân vùng đĩa: `cfdisk` và `sfdisk`. Chương trình `cfdisk`, giống như `fdisk` chỉ dành để làm việc với bảng phân vùng đĩa: nó không quan tâm chú ý đến thông tin có trên đĩa. Chỉ khác biệt với `fdisk` ở giao diện thuận tiện: chỉ dẫn sử dụng

lệnh và hệ thống trình đơn (thực đơn). Chương trình `sfdisk` có vài khả năng cao hơn, ví dụ, cho phép thao tác trên các phân vùng đã có của đĩa.

DOS sử dụng trường *begin* và *end* của bảng phân vùng và Interrupt 13 của BIOS (Int 13h) để truy cập tới đĩa, vì thế không thể sử dụng đĩa có dung lượng lớn hơn 8,4 Gbyte, ngay cả với các BIOS mới (về vấn đề này sẽ nói đến ở sau), còn phân vùng thì không thể lớn hơn 2,1 Gbyte (nhưng đây là do hạn chế của hệ thống tập tin FAT16).

Linux thì chỉ sử dụng trường *start* và *length* của bảng phân vùng đĩa và hỗ trợ các phân vùng chứa đến 232 sector, tức là dung lượng có thể đạt 2 Tbyte

Vì trong bảng chia ổ đĩa chỉ có 4 dòng cho các phân vùng, số phân vùng chính trên đĩa ngay từ đầu dã hạn chế: không thể lớn hơn 4. Khi mà 4 phân vùng trở thành ít, thì người ta sáng chế ra phân vùng lôgíc. Một trong số các phân vùng chính trở thành *mở rộng* (loại phân vùng - 5 hay F hay 85 trong hệ cơ số mười sáu). Và trong phân vùng mở rộng người ta tạo ra các *phân vùng lôgíc*. Phân vùng mở rộng không được sử dụng trực tiếp mà chỉ dùng để ghi các phân vùng lôgíc. Sector đầu tiên của phân vùng mở rộng ghi nhớ bảng phân vùng với bốn đầu vào: một dùng cho phân vùng lôgíc, một cho phân vùng mở rộng khác, còn hai cái còn lại không được sử dụng. Mỗi phân vùng mở rộng có một bảng chia của mình, trong bảng này, cũng giống như trong phân vùng mở rộng chính, chỉ sử dụng có hai dòng để đưa ra một phân vùng lôgíc và một phân vùng mở rộng. Như vậy, thu được một chuỗi các mắt xích từ bảng phân vùng, mắt xích đầu tiên mô tả ba phân vùng chính, và mỗi mắt xích tiếp theo – một phân vùng lôgíc và vị trí của bảng tiếp theo.

Chương trình `sfdisk` trên Linux cho thấy toàn bộ chuỗi này:

```
[root]# sfdisk -l -x /dev/hda

Disk /dev/hda: 784 cylinders, 255 heads, 63 sectors/track
Units = cylinders of 8225280 bytes, blocks of 1024 bytes, counting from 0

Device     Boot    Start     End     #cyls     #blocks      Id    System
/dev/hda1   *        0+       189       190-    1526143+      6    FAT16
/dev/hda2           190       783       594      4771305      5    Extended
/dev/hda3            0         -         0         0          0    Empty
/dev/hda4            0         -         0         0          0    Empty

/dev/hda5          190+      380       191-    1534176       6    FAT16
   -               381       783       403     3237097+      5    Extended
   -               190       189         0         0          0    Empty
   -               190       189         0         0          0    Empty

/dev/hda6          381+      783       403-    3237066       7    HPFS/NTFS
   -               381       380         0         0          0    Empty
   -               381       380         0         0          0    Empty
   -               381       380         0         0          0    Empty
```

Số phân vùng lôgíc theo nguyên tắc không hạn chế, vì mỗi phân vùng lôgíc có thể chứa bảng phân vùng và các phân vùng lôgíc của mình. Tuy nhiên trên thực thế vẫn có những hạn chế. Ví dụ, Linux không thể làm việc với hơn 15 phân vùng trên các đĩa SCSI và hơn 63 phân vùng trên đĩa IDE.

Phân vùng mở rộng trên một đĩa vật lý, hay trong một phân vùng mở rộng chứa nó (có thể gọi là “mẹ”) chỉ có thể làm một: không một chương trình phân

chia ổ đĩa nào trong số đã có (`fdisk` và tương tự) có thể tạo thêm một phân vùng mở rộng thứ hai.

Ổ đĩa trên Linux nói riêng (ổ đĩa vật lý) được truy cập qua tên của thiết bị: `/dev/hda`, `/dev/hdb`, `/dev/sda`, v.v... Các phân vùng chính có thêm số 1-4 trong tên thiết bị: `/dev/hda1`, `/dev/hda2`, `/dev/hda3`, còn phân vùng lôgíc thì có các tên: `/dev/hda5`, `/dev/hda6`, `/dev/hda7` ...(bắt đầu từ số 5). Từ những gì đề cập đến ở trên có thể suy ra tại sao lại có thể bỏ qua các tên như `/dev/hda3` hay `/dev/hda4` (đơn giản là phân vùng chính thứ ba và thứ tư không được tạo ra) và ngay sau `/dev/hda2` bạn đọc thấy `/dev/hda5` (phân vùng lôgíc trong phân vùng mở rộng `/dev/hda2`), và sau đó thì việc đánh số lại theo thứ tự thông thường.

Trong Windows các phân vùng lôgíc nhận được tên (chữ cái), bắt đầu từ chữ cái cuối dùng dành cho phân vùng chính. Ví dụ nếu một đĩa cứng có hai phân vùng chính (C: và D:) và một phân vùng mở rộng, trong phân vùng mở rộng tạo ra hai phân vùng lôgíc, thì những phân vùng lôgíc này sẽ được đặt tên E: và F:. Xin nói thêm, trong Windows NT và 2000/XP có thể thay đổi tên của các phân vùng đĩa.

2.3.3 Quá trình khởi động các HĐH của công ty Microsoft

Dù hệ điều hành có là gì, thì để có thể bắt đầu điều khiển máy tính, cần nạp HĐH vào bộ nhớ. Vì thế hãy xem xét qua quá trình khởi động của các HĐH khác nhau. Chúng ta chỉ quan tâm đến việc khởi động từ ổ đĩa cứng, nên sẽ không xem xét đến việc khởi động từ đĩa mềm, CD-ROM và qua mạng. Hãy bắt đầu từ MS-DOS và MS Windows cũ (xin đừng quên rằng, việc phát triển và hoàn thiện máy tính cá nhân song song với sự phát triển của HĐH của Microsoft và những quyết định sử dụng trong các HĐH này có ảnh hưởng mạnh đến quyết định của các nhà phát triển thiết bị).

Như bạn đọc biết, khi bật máy tính đầu tiên sẽ chạy chương trình POST (Power On Self Test). Chương trình xác định dung lượng bộ nhớ, thử nghiệm bộ nhớ, và xác định các thành phần khác (bàn phím, ổ cứng...), khởi động các thẻ adaptor. Trên màn hình thường xuất hiện các thông báo về dung lượng bộ nhớ, về việc thử nghiệm bộ nhớ, danh sách các thiết bị nhận ra (ổ đĩa cứng và mềm, bộ xử lý, cổng COM và v.v...).

Sau khi hoàn thành việc thử nghiệm POST gọi Int 19h. Công việc của Int 19h là tìm thiết bị khởi động. Việc tìm kiếm thực hiện theo thứ tự xác định trong Setup BIOS và theo cách thăm dò sector số 0 của các thiết bị tương ứng. Nếu đĩa có thể khởi động, thì trong sector số 0 của đĩa có *bản ghi khởi động chính* – Master Boot Record (MBR). Hai byte cuối cùng của MBR – *"số màu nhiệm"*, là dấu hiệu cho biết sector có MBR, và theo đó đĩa có thể khởi động. Ngoài "số màu nhiệm" MBR còn chứa bảng phân vùng đĩa đã nói ở trên, và một chương trình nhỏ – trình *khởi động chính*, kích thước chỉ có 446 (0x1BE) byte.

Bảng 2.1 cho thấy cấu trúc của sector khởi động chính sau khi cài đặt Windows.
MS-DOS, Windows95 và NT ghi nhớ DOS MBR trong khi cài đặt. Ngoài ra cũng có thể tạo MBR của MS với **câu lệnh DOS** sau: `fdisk /mbr`.

Xin trở lại với quá trình khởi động. Int 19h của BIOS nạp trình khởi động

Bảng 2.1: Cấu trúc của sector khởi động chính

Dịch chuyển	Nội dung
0x000	Mã của trình khởi động chính
0x1BE	Bảng phân vùng ổ đĩa
0x1FE	"Số màu nhiệm" (0xAA55)

chính vào bộ nhớ máy tính và chuyển quyền điều khiển cho chương trình này. Nhưng chương trình "bé nhỏ" này không đủ khả năng khởi động HĐH; tất cả những gì mà nó có thể làm – đó là nạp vào bộ nhớ chương trình mạnh hơn – trình khởi động thứ hai. Để làm được điều này, nó tìm trong bảng phân vùng kích hoạt và đọc vào bộ nhớ trình khởi động thứ hai, bắt đầu từ sector lôgíc đầu tiên của phân vùng kích hoạt. Hãy chú ý đến cụm từ "bắt đầu từ". Vì trình khởi động thứ hai trên các hệ thống khác nhau có độ dài khác nhau. Trong phân vùng được định dạng dưới hệ thống tập tin FAT, trình khởi động thứ hai chiếm một sector (512 byte). Trong phân vùng định dạng dưới hệ thống tập tin NTFS, trình khởi động thứ hai chiếm vài sector.

Trình khởi động thứ hai nạp lớp chương trình đầu tiên, cần thiết cho việc khởi động hệ điều hành. Trong trường hợp MS DOS chương trình khởi động động nạp IO.SYS theo địa chỉ 700h, sau đó MSDOS.SYS và chuyển quyền điều khiển cho SYSINIT của môđun IO.SYS.

Nếu vì lý do nào đó không tìm thấy trên đĩa phân vùng kích hoạt, thì quá trình khởi động sẽ tiếp tục với việc xử lý Int 18h. Trường hợp này trên thực tế **rất hiếm** khi sử dụng, nhưng khả năng này có thể có ích trong trường hợp nào đó. Trong khi khởi động từ xa, khi hệ điều hành khởi động từ máy chủ, thì Int này được POST chuyển hướng lên ROM của cạc mạng.

Đối với các HĐH khác của Microsoft thì quá trình khởi động diễn ra tương tự.

- Windows95 khởi động giống như DOS nhưng thay thế IO.SYS và MSDOS.SYS bởi các tập tin của mình. Các tập tin DOS được giữ lại dưới các tên tương ứng IO.DOS và MSDOS.DOS. Khi bạn đọc chọn khởi động DOS, Windows95 sẽ đổi tên các tập tin của mình với phần mở rộng w40 và phục hồi tên ban đầu của các tập tin hệ thống của DOS. Quá trình khởi động tiếp tục với việc nạp IO.SYS. Như thế, sector khởi động của DOS và Windows95 là như nhau.

- Windows NT4 sử dụng MBR DOS, nhưng thay thế bản ghi khởi động của phân vùng kích hoạt bằng cách thay thế NTLDR vào chỗ IO.SYS. Đây là một chương trình mạng và có thể làm được nhiều thứ. Ví dụ, có thể tìm tập tin boot.ini và nếu như tham số timeout lớn hơn 0, thì đưa ra trình đơn (thực đơn) khởi động.

Mỗi dòng của phần `[operating systems]` trong tập tin boot.ini xác định một phương án (một HĐH) khởi động và được viết theo mẫu sau:

```
địa_chỉ_trình_khởi_động_thứ_hai=''tên_gọi_của_phương_án''
```

Địa chỉ của trình khởi động thứ hai có thể là một phân vùng cụ thể nào đó của đĩa cũng như tập tin khởi động. Dưới đây là một ví dụ tập tin boot.ini:

```
[operating systems]
multi(0)disk(0)rdisk(0)partition(3)\WINNT="Windows NT Workstation 4"
C:\="Microsoft Windows"
C:\BOOTSECT.LNX="Linux"
```

Nếu người dùng chọn NT, thì sẽ khởi động theo địa chỉ phân vùng được chỉ trên dòng đầu tiên. Trên dòng tương ứng với phương án Microsoft Windows, chỉ đưa ra "C:\", vì tên của tập tin khởi động được lấy theo mặc định: bootsect.dos. Tập tin được nạp vào bộ nhớ và quá trình khởi động được tiếp tục giống như khi bản ghi khởi động được nạp bởi mã chương trình từ MBR.

Đối với việc khởi động các hệ thống khác, có thể sử dụng cách đó. Chỉ cần thêm vào boot.ini các dòng chứa liên kết đến tập tin khởi động khác. Khi chọn các dòng này sẽ khởi động HĐH tương ứng. Trong ví dụ trên Linux cũng được khởi động theo các này. Trong tập tin C:\BOOTSECT.LNX cần ghi nội dung của bản ghi khởi động, tạo bởi Linux (nói đúng hơn – LILO, trình khởi động tiêu chuẩn của Linux).

2.3.4 Vấn đề với các đĩa lớn

Trên MS-DOS và các phiên bản đầu tiên của Windows truy cập tới đĩa (trong đó có cả bước khởi động đầu tiên của HĐH) được tổ chức qua Int 13 (Int 13h) của BIOS. Khi này sử dụng sự đánh địa chỉ sector trên đĩa trên cơ sở C/H/S (xem trên). Chính xác hơn:

- AH — chọn thao tác;
- CH — 8 bit nhỏ hơn của số cylinder;
- CL — 7-6 bit tương ứng bit lớn của số cylinder, 5-0 tương ứng số sector;
- DH — số của đầu đọc;
- DL — số của đĩa(80h hay 81h).

(Cần lưu ý rằng việc đánh số cylinder vật lý và rãnh thường bắt đầu từ 0, còn sector trên rãnh đánh số bắt đầu từ 1). Tuy nhiên trên thực tế số đầu đọc không quá 16, còn số sector trên rãnh – không quá 63, và dù có dùng 10 bit để chỉ ra cylinder, BIOS vẫn không thể làm việc với đĩa dung lượng lớn hơn 1024*63*16*512 = 528 Mbyte.

Để vượt qua hạn chế này, người ta áp dụng nhiều cách "láu cá" khác nhau. Ví dụ, Extended CHS (ECHS) hay "Large disk support" (đôi khi còn gọi là "Large") sử dụng ba bit chưa dùng đến của số thứ tự đầu đọc để tăng số cylinder. Cách này cho pehsp sử dụng cấu trúc "hình học giả mạo của đĩa" với 1024 cylinder, 128 đầu đọc và 63 sector/rãnh. Biến đổi Extended CHS thành địa chỉ CHS thực (có thể chứa đến 8192 cylinder) được BIOS thực hiện. Cách này cho phép làm việc với đĩa có dung lượng đến 8192*16*63*512 = 4 227 858 432 byte hay 4,2 Gbyte.

Nhưng các nhà phát triển càng ngày càng tăng mật độ ghi của đĩa, số đĩa và số rãnh, và còn phát minh ra các phương pháp khác để tăng dung lượng đĩa. Ví dụ, số sector trên rãnh không còn cố định mà trở thành khác nhau trên các rãnh

khác nhau (trên các rãnh nằm gần rìa ngoài của đĩa, dài hơn, số sector được tăng lên). Kết quả là bộ ba số C/H/S không còn phản ánh đúng cấu trúc "hình học của đĩa", và các phiên bản BIOS cũ không thể hỗ trợ truy cập tới toàn bộ không gian đĩa.

Khi đó người ta nghĩ ra phương pháp khác để làm việc với các đĩa lên qua Int 13h - *đánh địa chỉ các khối theo đường thẳng* ("Linear Block Addressing" hay LBA). Không đi sâu vào chi tiết, có thể nói rằng tất cả sector trên đĩa được đánh số một cách tuần tự, bắt đầu từ sector đầu tiên trên rãnh số 0 của cylinder số 0. Thay vào chỗ địa chỉ CHS mỗi sector nhận được một địa chỉ lôgíc – số thứ tự của sector trong tổng số tất cả sector. Việc đánh số sector lôgíc bắt đầu từ 0, trong đó sector số 0 chứa bản ghi khởi động chính (MBR). Trong Setup BIOS hỗ trợ biến đổi số thứ tự theo đường thẳng thành địa chỉ CHS có dạng "Hỗ trợ LBA". Như vậy, trong các phiên bản BIOS mới thường có lựa chọn với ba phương án: "Large", "LBA", và "Normal" (phương án cuối cùng có nghĩa là không thực hiện biến đổi địa chỉ).

Tuy nhiên trong chế độ LBA việc sử dụng đĩa vật lý vẫn được thực hiện qua Int 13h, mà Int 13h vẫn sử dụng bộ 3D (C,H,S). Vì nguyên nhân này xuất hiện hạn chế lên dung lượng của đĩa: BIOS, và theo đó, MS-DOS và các phiên bản Windows đầu tiên không thể đánh địa chỉ các đĩa có dung lượng lớn hơn 8,4 Gbyte.

Cần chú ý rằng hạn chế nói trên chỉ áp dụng với các đĩa có giao diện IDE. Trong các controller của đĩa SCSI, số của sector được chuyển vào các lệnh SCSI, và sau đó tự đĩa tìm ra vị trí cần thiết, vì thế hạn chế lên dung lượng đĩa không xuất hiện.

Một lần nữa muốn nhắc lại rằng, tất cả những hạn chế nói trên chỉ có ý nghĩa trong giai đoạn khởi động HĐH. Bởi vì Linux và các phiên bản Windows mới nhất khi làm việc với đĩa đã không còn sử dụng Int 13 của BIOS, mà sử dụng driver riêng của mình. Nhưng trước khi có thể sử dụng driver của mình, hệ thống phải được nạp. Vì thế trong giai đoạn khởi động đầu tiên bất kỳ hệ thống nào cũng cần sử dụng BIOS. Điều này hạn chế việc đặt nhiều hệ thống ra ngoài vùng 8 Gbyte đĩa đầu tiên: chúng không thể khởi động từ đó, mặc dù sau khi khởi động thì có thể làm việc với các đĩa có dung lượng lớn hơn nhiều. Để có thể hiểu cách thoát khỏi những hạn chế này, chúng ta cần một chút kiến thức về quá trình khởi động của HĐH Linux.

2.4 Lựa chọn trình khởi động

2.4.1 Trình khởi động GRUB[1]

GRUB (**GR**and **U**nified **B**ootloader) – trình khởi động hết sức mạnh có khả năng khởi động rất nhiều HĐH miễn phí cũng như HĐH thương mại. GRUB được Erich Boleyn viết vào năm 1995 để khởi động hệ thống GNU Mach, vì không thể sử dụng những trình khởi động khác. Sau đó vào năm 1999 Gordon Matzigkeit và Yoshinori K. Okuji chuyển GRUB thành một gói chương trình GNU, đưa chương trình này thành một phần mềm mã nguồn mở. Mặc dù mới ra đời và số phiên bản

[1] Phần về GRUB này do người dịch viết.

còn rất nhỏ[2] nhưng đây là sự lựa chọn tốt đối với phần lớn người dùng máy tính cá nhân. Nếu không có nhu cầu đặc biệt nào đó thì bạn nên chọn trình khởi động này. Các bản phân phối Linux lớn (Debian, SuSE, Fedora,...) cũng đã chuyển sang sử dụng GRUB làm lựa chọn theo mặc định.

Một trong những tính năng quan trọng của GRUB là tính mềm dẻo. GRUB có thể hiểu các hệ thống tập tin và định dạng thực thi của nhân, vì thế bạn có thể nạp HĐH theo cách ưa thích. Ngoài ra, nếu không muốn dùng giao diện dòng lệnh, thì bạn có thể cài đặt và sử dụng giao diện thực đơn và thay đổi giao diện thực đơn theo mong muốn của mình. Một điểm mạnh khác của GRUB đó là "grub shell" có thể chạy khi bắt đầu khởi động hoặc sau khi đã khởi động xong hệ thống. Bằng grub shell bạn có thể "giả lập" (emulate) trình khởi động này và cài đặt GRUB.

Bây giờ chúng ta xem xét ngắn gọn về tên gọi thiết bị dùng trong GRUB, vì cú pháp thiết bị trong trình khởi động này có khác một chút so với những gì mà bạn đã thấy trên những hệ thống của mình. Bạn cần hiểu cú pháp này để biết cách chỉ ra một ổ đĩa hay phân vùng nào đó. Ví dụ một cú pháp là:

```
(fd0)
```

Trước tiên cần nói GRUB yêu cầu tất cả các tên thiết bị phải đặt trong ngoặc '(' và ')'. Phần `fd` có nghĩa là đĩa mềm. Số '0' chỉ ra số thứ tự của ổ, đếm bắt đầu từ *0*.

Ví dụ 2:

```
(hd0,1)
```

Ở đây 'hd' có nghĩa là ổ cứng. Số nguyên '0' đầu tiên cho biết số thứ tự của ổ, tức là ổ cứng thứ nhất. Số nguyên thứ hai ('1') cho biết số thứ tự của phân vùng (chúng ta không xem xét các HĐH khác Linux). Xin nhắc lại một lần nữa là các số đếm đều bắt đầu từ số không '0'. Trong trường hợp này đây là phân vùng thứ hai của ổ cứng thứ nhất. GRUB sử dụng một phân vùng của đĩa chứ không phải toàn bộ đĩa.

Ví dụ 3:

```
(hd0,4)
```

Đây là phân vùng mở rộng ("extended partition") thứ nhất của đĩa cứng thứ nhất. Chú ý rằng các phân vùng mở rộng được đếm bắt đầu từ '4' không phụ thuộc vào số phân vùng chính ("primary partition") thực tế có trên đĩa. Cần chú ý thêm là GRUB không phân biệt IDE và SCSI. Nó đếm số thứ tự ổ bắt đầu từ '0' không phụ thuộc vào dạng đĩa.

Làm sao để chỉ ra một tập tin? Hãy xem ví dụ sau đây:

```
(hd0,0)/vmlinuz
```

Dòng này chỉ ra tập tin '`vmlinuz`' nằm trên phân vùng đầu tiên của ổ cứng đầu tiên. Hết sức đơn giản! Thông tin trong phần này sẽ giúp bạn hiểu được cấu hình của GRUB sẽ nói đến sắp tới đây.

[2] hãy so sánh số phiên bản của GRUB và LILO

2.4.2 Trình khởi động LILO

Trình khởi động LILO được viết bởi Werner Almesberber. LILO có thể khởi động nhân Linux từ đĩa mềm, đĩa cứng, và cũng có thể khởi động các hệ điều hành khác: PC/MS-DOS, DR DOS, OS/2, Windows 95/98, Windows NT/2000/XP, 386BSD, SCO UNIX, Unixware v.v... LILO cho phép chọn đến 16 hệ điều hành khác nhau để khởi động.

LILO không phải là chương trình đơn lẻ mà là một bộ gồm nhiều chương trình: trình khởi động, các chương trình sử dụng để cài đặt và cấu hình trình khởi động, và các tập tin phục vụ:

- chương trình `/sbin/lilo`, chạy dưới Linux, phục vụ để ghi tất cả thông tin cần thiết trong giai đoạn khởi động vào các chỗ tương ứng. Cần chạy chương trình này sau mỗi lần có thay đổi trong nhân hay trong tập tin cấu hình LILO;
- các tập tin phục vụ, cần cho LILO trong thời gian khởi động. Những tập tin này thường nằm trong thư mục `/boot`. Quan trọng nhất trong số chúng – đó là bản thân trình khởi động (xem phía dưới) và tập tin map (`/boot/map`); trong tập tin này có chỉ ra vị trí của nhân. Một tập tin quan trọng khác – tập tin cấu hình LILO; thường có tên `/etc/lilo.conf`;
- trình khởi động – đây là phần LILO được nạp vào bộ nhớ đầu tiên qua Int của BIOS; trình khởi động nạp nhân Linux hay sector khởi động của hệ điều hành khác. Trình khởi động gồm có hai phần. Phần thứ nhất được ghi vào sector khởi động và phục vụ để nạp phần thứ hai, có kích thước lớn hơn rất nhiều. Cả hai phần thường được ghi trên đĩa trong tập tin `/boot/boot.b`.

Cần nhớ rằng, định dạng của sector khởi động tạo ra bởi LILO khác với định dạng MBR của DOS. Vì thế nếu ghi sector khởi động LILO vào MBR, thì các hệ điều hành đã cài của Microsoft sẽ ngừng khởi động (nếu như không có các biện pháp bổ sung).

Sector khởi động của LILO có thể được thiết kế để sử dụng như sector khởi động của phân vùng, trong đó có chỗ cho bảng phân vùng. Sector khởi động của LILO trong khi cài đặt có thể đặt vào những chỗ sau:

- sector khởi động của đĩa mềm trong định dạng Linux (`/dev/fd0`, ...);
- MBR của đĩa cứng đầu tiên (`/dev/hda, /dev/sda`, ...);
- sector khởi động của phân vùng chính với hệ thống tập tin Linux trên đĩa cứng đầu tiên (`/dev/hda1, /dev/hda2`, ...);
- sector khởi động của phân vùng lôgíc trong phân vùng mở rộng đĩa cứng đầu tiên (`/dev/hda5`, ...). Sự thật là phần lớn chương trình dạng `fdisk` không đề ra khả năng khởi động khởi động từ phân vùng mở rộng và từ chối việc kích hoạt phân vùng này. Vì vậy trong thành phần LILO có chứa một chương trình đặc biệt (`activate`) cho phép vượt qua hạn chế này. Tuy nhiên chương trình `fdisk` của bản phân phối Linux hỗ trợ khả năng kích hoạt phân vùng mở rộng. Cần sử dụng tùy chọn `-b` hoặc biến `BOOT`.

Sector khởi động của LILO không thể đặt vào các chỗ sau:

- sector khởi động của đĩa mềm hay phân vùng chính, với định dạng hệ thống tập tin khác Linux;
- trong phân vùng swap của Linux;
- trên đĩa cứng thứ hai.

Ngoài ra, cần nhớ rằng, LILO trong thời gian khởi động cần những tập tin sau:

- `/boot/boot.b`;
- `/boot/map` (tạo ra bởi lệnh `/sbin/lilo`);
- tất cả phiên bản nhân khởi động (nếu bạn đọc chọn phiên bản nhân khi khởi động);
- sector khởi động của các hệ điều hành khác mà bạn đọc muốn khởi động qua LILO;
- tập tin chứa các thông báo đưa ra khi khởi động (nếu được xác định).

Như vậy, sector khởi động LILO cũng như những tập tin đã liệt kê (trong số đó có các tập tin bạn đọc sẽ cài đặt sau này) cần nằm trong phạm vi 1024 cylinder đầu tiên của đĩa cứng, bởi vì chúng cần được truy cập qua BIOS. Xem phần nói về hạn chế của BIOS ở trên.

Bắt đầu từ phiên bản 21, LILO đưa ra màn hình trình đơn (thực đơn) cho phép chọn hệ thống để khởi động (trước đây cần nhấn phím Tab để gọi trình đơn này).

2.4.3 Các trình khởi động khác

Ngoài GRUB và LILO để khởi động Linux có thể khởi động các trình khởi động khác.

- Nếu như trước khi cài đặt Linux đã có HĐH Windows NT/2000/XP, thì trình khởi động bạn đọc có thể sử dụng là OS Loader của NT. So sánh với LILO thì trình khởi động OS Loader có ít nhất hai ưu thế. Thứ nhất, tất cả cấu hình cũ không bị mất (chúng ta có thể chọn khởi động Windows hay Linux theo lựa chọn), và thứ hai, có thể cài đặt Linux lên đĩa mà LILO không thể khởi động, ví dụ, ổ đĩa thứ hai trên controller thứ hai (Secondary Slave).
- Nếu như trước khi cài đặt Linux bạn đọc chỉ có HĐH Windows 95 hay Windows 98 và không có Windows NT/2000 hay XP, thì OS Loader không được cài đặt. Và nếu như vì một lý do nào đó bạn đọc không muốn cài đặt LILO, thì có thể sử đụng chương trình khởi động loadlin.exe (thường đi kèm với bản phân phối Linux);

- Thời gian gần đây trong thành phần bản phân phối Linux thường có chương trình khởi động GRUB.
- Trong thành phần OS/2 của công ty IBM có chương trình khởi động Boot Manager. Trong rất nhiều hướng dẫn người ta khuyên dùng chương trình này để tổi chức khởi động nhiều HĐH.
- Trong các nguồn thông tin khác nhau còn nhắc đến chương trình System Commander;
- Thêm một trình khởi động khác có trong thành phần gói PartitionMagic của công ty Power Quest. Chúng ta sẽ nói về chương trình này trong phần nhỏ tiếp theo.

Ngoài ra tác giả còn thấy nói đến một loạt các trình khởi động khác (một số có thể tìm thấy trong thư mục `/public/ftp/pub/Linux/system/boot/loaders` trên trang ftp://metalab.unc.edu/. Nhưng vì tác giả không sử dụng những chương trình này, nên không thể nói cụ thể cách sử dụng chúng. Và tất cả những lời khuyên dùng sau của tác giả sẽ dựa trên việc sử dụng LILO, NT Loader và loadlin.exe. Nếu như có ý muốn cài đặt chương trình khởi động khác, thì bạn đọc cần đọc hướng dẫn cài đặt và sử dụng của nó.

2.4.4 Các phương án khởi động

Như vậy, theo ý kiến của tác giả có các phương án khởi động sau:

- Nếu trên máy chỉ có một hệ điều hành Linux duy nhất, hãy dùng GRUB.
- Nếu đã cài đặt Windows NT hay Windows 2000/XP, thì hãy sử dụng GRUB.
- Nếu có Windows 95 hay Windows 98 trên FAT16, và bạn đọc không muốn cài đặt chương trình khởi động nào khác, thì có thể sử dụng GRUB hoặc LILO, hoặc đầu tiên chạy DOS và sau đó khởi động Linux nhờ chương trình loadlin.exe (hay một chương trình tương tự, có vài chương trình như vậy, nhưng chúng ta sẽ không xét đến).
- Nếu đã cài đặt Windows 95 OSR2 hay Windows 98 trên FAT32, và bạn đọc không muốn cài đặt thêm chương trình khởi động, thì cần sử dụng loadlin.exe. Rất nhiều HOWTO khẳng định rằng không cần sử đụng LILO, nếu như phân vùng kích hoạt có định dạng FAT32, mặc dù tác giả không rõ nguyên nhân. Tuy nhiên thí nghiệm khởi động Linux qua NT Loader, cài đặt trên phân vùng FAT32, của tác giả đã kết thúc không thành công. Vì thế, trong trường hợp này tác giả đã phải sử dụng chương trình loadlin.exe. Chương trình này đã hoàn thành tốt nhiệm vụ, và tạo cho tác giả một ấn tượng tốt, vì thế tác giả khuyên bạn đọc sử dụng loadlin.exe để khởi động Linux.

Trong những phần tiếp theo tác giả sẽ cho biết cách cài đặt Linux, sử dụng tất cả bốn phương án khởi động: qua trình khởi động NT Loader, trình khởi động GRUB, trình khởi động LILO và trình khởi động loadlin.exe. Tuy nhiên trước khi cài đặt trình khởi động cần chuẩn bị các phân vùng trên đĩa, hay ít nhất là nghĩ cách tổ chức chúng.

2.5 Chuẩn bị các phân vùng trên đĩa

2.5.1 Lời khuyên khi tạo phân vùng

Đưa ra lời khuyên ở đây không phải là việc dễ dàng, vì phân vùng đĩa phục thuộc rất nhiều vào ý thích và nhu cầu của chủ nhân đĩa. Nhưng cũng xin thử đưa ra vài đề nghị sau. Tác giả sẽ đặt tên đĩa và phân vùng theo "tiêu chuẩn" của Linux, tức là `/dev/hda`, `/dev/hdb`, v.v... đối với đĩa và `/dev/hda1`, `/dev/hda2`, v.v... – đối với các phân vùng.

Việc phân chia đĩa thành các phân vùng là cần thiết, bởi vì Windows và Linux sử dụng các cách lưu trữ thông tin trên đĩa và sau đó đọc chúng từ đĩa khác nhau. Chính vì thế tốt hơn hết là dành cho mỗi hệ điều hành một (hoặc thậm chí một vài như chúng ta sẽ thấy ở dưới) phân vùng riêng.

Đầu tiên chúng ta hãy xem xét một trường hợp đơn giản – dung lượng ổ đĩa của bạn đọc không vượt quá 8,4 Gbyte (nói chính xác hơn – số cylinder không vượt quá 1024). Trong trường hợp này mọi thứ đều đơn giản: bạn đọc chỉ việc chia đĩa làm sao để đủ chỗ cho hệ điều hành sẽ cài đặt. Có thể sử dụng dữ liệu cho biết kích thước đĩa nhỏ nhất cần thiết để cài đặt hệ điều hành với cấu hình cơ bản trong bảng 2.2.

Bảng 2.2: Nhu cầu sử dụng không gian đĩa của HĐH

Hệ điều hành	**Yêu cầu**
Windows 95	100 Mbyte
Windows 98	200 Mbyte
Windows NT	200 Mbyte
Windows 2000	700 Mbyte
Linux Red Hat 6.2 (Workstation với KDE)	700 Mbyte

Tuy nhiên xin hãy nhớ rằng, không những phải tính kích thước các tập tin của bản thân hệ điều hành, mà còn phải tính cả kích thước của các chương trình bạn đọc dự tính chạy. Và còn phải dành một phần dự trữ không nhỏ cho các chương trình sẽ cài đặt sau này (không thể tránh khỏi!). Hãy tính rằng, 700 Mbyte dành cho Linux ở trong bảng nói trên chỉ dành cho các chương trình cài đặt cùng với Linux theo mặc định, trong số đó có, ví dụ, chương trình soạn thảo rất mạnh Lyx. Đối với Windows cũng tương tự như vậy.

Theo kinh nghiệm của tác giả thì để làm việc với Windows 95/98, Windows NT và Linux các phân vùng với kích thước 800-1000 Mbyte là đủ (tất nhiên, nếu bạn đọc không cài đặt các gói chương trình lớn, như OpenOffice.Org), còn đối với Windows 200 thì cần phân vùng lớn hơn.

Bây giờ chúng ta sẽ xem xét vấn đề chia các phân vùng cho Linux. Ở đây không thể chỉ chia một phân vùng. Thứ nhất, cần chia một *phân vùng swap* riêng biệt cho Linux. Khi xác định dung lượng của phân vùng swap Linux cần tính đến những yếu tố sau:

- Trong Linux, RAM và không gian swap hợp lại tạo thành bộ nhớ ảo chung. Ví dụ, nếu bạn đọc có 256 MByte RAM và 128 Mbyte không gian swap, thì sẽ có 384 Mbyte bộ nhớ ảo.

- Để làm việc với Linux cần ít nhất 16 Mbyte bộ nhớ ảo, vì thế nếu bạn đọc chỉ có 4 Mbyte RAM, thì cần phân vùng swap không nhỏ hơn 12 Mbyte.
- Kích thước của phân vùng swap có thể lớn bao nhiêu tùy thích, tuy nhiên không cần cấu hình quá nhiều nếu không cần thiết. Thông thường chỉ cần dùng swap khi máy ít bộ nhớ RAM hoặc chạy máy chủ với nhiều ứng dụng nặng. Trong tất cả mọi trường hợp tốt nhất tránh dùng swap vì nó chậm hơn RAM nhiều.[3]
- Khi tính kích thước của không gian swapping, cần nhớ rằng kích thước quá lớn có thể là vô ích. Trên máy tính với 16 Mbyte RAM khi cài đặt Linux với cấu hình chuẩn và các chương trình ứng dụng chuẩn thì 48 Mbyte không gian swapping là đủ. Còn nếu cài đặt Linux với cấu hình nhỏ nhất, thì không cần đến không gian swap. Tất nhiên, kích thước chính xác của không gian swap phụ thuộc lớn vào chương trình sẽ được cài đặt.

Nói chung, chỉ nên suy nghĩ về vấn đề dung lượng của phân vùng swap khi có một đĩa nhỏ và ít bộ nhớ RAM. Trong trường hợp ngược lại hãy phân chia để tổng số dung lượng của bộ nhớ ảo (gồm RAM và phân vùng swap) không nhỏ hơn 128 Mbyte. Còn nếu như bạn đọc có 128 Mbyte RAM hay nhiều hơn, thì phân vùng này có thể không cần thiết.

Tất cả các phần còn lại của Linux và các chương trình hỗ trợ theo nguyên tắc có thể đặt vào một phân vùng. Tuy nhiên, việc đặt hệ thống tập tin Linux lên vài phân vùng riêng rẽ là có ý nghĩa. Ví dụ, có nhà chuyên gia khuyên nên dành cho hệ thống tập tin Linux ba phân vùng (nếu tính cả swap thì thành 4). Phân vùng thứ nhất (theo ý kiến cá nhân tác giả, 1 Gbyte là đủ) sẽ chứa hệ thống tập tin gốc (`/`). Phân vùng thứ hai dành cho thư mục `/home`. Còn phân vùng thứ ba được gắn vào thư mục `/usr`. Việc phân chia như vậy dựa trên những lý lẽ sau. Dù HĐH Linux có ổn định và đáng tin cậy đến đâu, thì thỉnh thoảng cũng cần cài đặt lại. Ví dụ, bạn đọc muốn cập nhật phiên bản mới của bản phân phối, hoặc vì ít kinh nghiệm sử dụng nên làm hỏng tập tin hệ thống quan trọng, hoặc đơn giản là muốn cài đặt một bản phân phối khác. Nếu như tất cả được cài đặt vào một phân vùng, thì khi cài đặt lại những dữ liệu đã làm ra và ghi nhớ trong thư mục cá nhân sẽ bị mất (nếu không có bản sao chép). Ngoài ra, sẽ bị mất cả những chương trình đã cài từ mã nguồn, hay cài bằng phương pháp khác. Phần lớn những gói chương trình này được cài vào thư mục `/usr`. Nếu dành cho thư mục này một phân vùng riêng và khi cài đặt không định dạng lại chúng, thì những chương trình nói trên sẽ được giữ lại và **có thể** sẽ làm việc (rất có thể cần vài cấu hình nhỏ) sau khi cài đặt lại hệ thống. Trong tiêu chuẩn về hệ thống tập tin của Linux FHS (cụ thể xin xem ở chương **??**) cũng có lời khuyên về việc đặt thư mục `/usr` lên một phân vùng riêng.

Theo tác giả thấy, những ý kiến nói trên đã đủ để bạn đọc tự tìm ra phương án phân chia ổ đĩa của mình, trong trường hợp chỉ có một ổ đĩa nhỏ. Bây giờ chúng ta xem xét trường hợp đĩa với số cylinder lớn hơn 1024.

Từ những gì đã nói đến ở phần trước (hạn chế dung lượng đĩa cứng), cần đặt chương trình khởi động trong phạm vi 1024 cylinder đầu tiên. Nhân tiện, NT Loader không nhất thiết phải đặt vào phân vùng NTFS, cũng như không nhất

[3] Xin cám ơn bác Tony Lý về mục này

thiết phải đặt vào phân vùng chứa các tập tin khác của HĐH. Như đã nói ở trên, đối với Linux có thể đặt thư mục gốc cùng với thư mục con `/boot` vào các cylinder "thấp" (trong vòng 1024 đầu tiên), còn các thư mục khác – ở chỗ nào tùy thích.

Như vậy trong trường hợp này, những đề nghị của tác giả cho ra bảng tổng kết sau:

- phần khởi động của tất cả các hệ thống Microsoft đặt vào phân vùng chính đầu tiên của đĩa, với định dạng FAT16 (DOS);
- phân vùng chính tiếp theo dành cho thư mục gốc (`/`), kích thước khoảng 1 Gbyte;
- phân vùng chính thứ ba dành cho swap của Linux (lời khuyên về kích thước của phân vùng này xem ở trên);
- phần còn lại của đĩa đặt thành phân vùng mở rộng;
- trong phân vùng mở rộng tạo các phân vùng lôgíc cho mỗi HĐH sẽ cài đặt: Windows 98, Windows NT/2000/XP, và đồng thời cho các hệ thống tập tin `/home` và `/usr` của HĐH Linux (trong `/home` sẽ đặt các tập tin riêng của người dùng, còn trong `/usr` – chương trình sẽ cài đặt).

Tất nhiên, nếu như bạn đọc chỉ có Windows 95 với FAT16, thì có thể để Windows trên phân vùng đầu tiên. Nếu như trên máy đã cài đặt Windows NT hay có FAT32, thì một phân vùng FAT16 cũng không thừa. Thứ nhất, kể cả trong trường hợp hệ thống có vấn đề, bạn đọc có thể khởi động từ đĩa mềm DOS (tạm thời khi chưa làm quen với Linux một cách "tường tận") và thấy được rằng đĩa cứng làm việc bình thường. Thứ hai, hệ thống tập tin FAT16 được hỗ trợ trên mọi HĐH, trong đó có Linux, vì thế phân vùng này có thể phục vụ cho việc trao đổi tập tin giữa các hệ thống. Nhưng không nên để phân vùng này lớn, vì FAT16 sử dụng không gian đĩa rất không hợp lý. Chính vì vậy hãy dành cho phân vùng này khoảng 256 hoặc 512 Mbyte.

Những lời khuyên này đưa ra với giả thiết rằng, bạn đọc chỉ có một đĩa cứng. Nếu như bạn đọc có 2, thì vẫn sử dụng được những lời khuyên này, chỉ có điều phân vùng swap tốt hơn đặt trên đĩa khác với đĩa dành cho Linux. Người ta nói rằng như vậy tăng tốc độ làm việc trong Linux (cũng dễ hiểu vì đầu đọc ít phải chạy hơn).

2.5.2 Chương trình để phân chia ổ đĩa

Sau khi hoàn thành kết hoạch chia ổ đĩa, cần lựa chọn công cụ để đưa kế hoạch này thành hiện thực. Chương trình phân chia đĩa được biết đến nhiều nhất là `fdisk`; trên mọi hệ điều hành đều có phiên bản riêng của chương trình này. Và không cần gì hơn ngoài chương trình này, nếu như phân chia ổ đĩa trắng, không chứa bất ký dữ liệu nào. Nhưng chúng ta đang xem xét trường hợp đã có HĐH nào đó trên đĩa và cần phân chia ổ đĩa mà không làm mất thông tin. `fdisk` không thích hợp cho những mục đích như vậy.

Trong thành phần các bản phân phối Red Hat và BlackCat (rất có thể trong các bản phân phối khác) có chương trình `fips`, phục vụ cho phân chia ổ đĩa.

Tuy nhiên, theo ý kiến của người dùng thì không nên sử dụng chương trình này. Vì thế lời khuyên của tác giả với bạn đọc, những người dùng Linux mới – nếu như muốn phân chia lại ổ đĩa mà không làm mất thông tin, thì hãy tìm chương trình Partition Magic của công ty Power Quest (http://www.powerquest.com) và sử dụng chương trình này.

Thứ nhất, chương trình này cho phép phân chia lại ổ đĩa mà không làm mất thông tin (tức là, tất cả những cài đặt và cấu hình trước đó sẽ được ghi lại). Khi này, không chỉ tạo được phân vùng mới từ chỗ trống trên đĩa, mà còn có thể di chuyển các phân vùng đã có theo ý muốn.

Thứ hai, chương trình này (thậm chí trong phiên bản dành cho DOS) cung cấp một giao diện đồ họa dễ sử dụng có hỗ trợ chuột, và mọi thao tác cũng như thay đổi đều thấy rõ ràng. Điều này rất quan trọng với người dùng mới.

Khi tạo phân vùng cần để ý không cho ranh giới giữa các phân vùng cắt lẫn nhau.

Tác giả cho rằng, những thông tin đã đưa đủ để bạn đọc lập kế hoạch và thực hiện việc phân chia ổ đĩa thành các phân vùng. Vì thế tiếp theo chúng ta sẽ xem xét các phương án cài đặt hai HĐH trên một máy tính.

2.6 Windows NT và Linux: khởi động qua NT OS Loader

Trong phần này, khi nói về Windows NT xin ngầm hiểu cả Windows 2000 và NT, vì "quan hệ" của chúng đối với việc cài đặt Linux hoàn toàn giống nhau. Chúng ta giả thiết là Windows NT đã được cài vào phân vùng `/dev/hda2` (nếu như bạn đọc nhớ, `/dev/hda1` sẽ dành cho phân vùng FAT16). Nếu HĐH Windows NT đã được cài đặt, nghĩa là trình khởi động OS Loader cũng đã được cài đặt. Và như thế có thể sử dụng chương trình này để khởi động Linux. Tác giả hy vọng rằng bạn đọc đã sao lưu những thông tin có giá trị của mình. Các bước cài đặt có thể mô tả như sau:

1. Nếu như bạn đọc chưa cài đặt Linux bao giờ, thì trước khi bắt đầu cần chuẩn bị đĩa mềm khởi động và phục hồi Windows NT. Để tạo đĩa mềm khởi động chỉ cần định dạng lại đĩa mềm, rồi sao chép lên đó các tập tin ntldr, ntdetect.com và boot.ini từ thư mục gốc của ổ đĩa khởi động NT. Chương trình tạo đĩa phục hồi Windows 2000/XP có thể chạy từ trình đơn hệ thống (lệnh Backup trong Start/Program/Accessories).

2. Dùng chương trình Partition Magic để lấy một phần đĩa trống và từ đó tạo ra phân vùng với dạng ext2(3) (hệ thống tập tin Linux) và phân vùng swap. Cách tính kích thước của chúng đã nói ở trên.

3. Cài đặt Linux theo chỉ dẫn của bản phân phối. Trong khi cài đặt cần chú ý đến những điểm sau:

   - thứ nhất, trong quá trình cài đặt nhất định phải tạo ra các đĩa mềm khởi động Linux. Tức là cần trả lời "Yes, make a BOOT DISK" (hay tương tự thế, tùy thuộc vào bản phân phối) khi được hỏi có tạo đĩa mềm khởi động hay không. Đĩa mềm này sẽ được dùng đến ở sau. Ngoài ra, có thể sử dụng đĩa mềm này để khởi động Linux. Đây cũng

là một phương án khởi động, và hơn nữa khác với đĩa mềm khởi động DOS, sau khi khởi động hệ thống không còn yêu cầu đĩa mềm nữa, có thể bỏ nó ra khỏi ổ, sử dụng ổ để đọc các đĩa mềm khác. Tuy nhiên cách khởi động này cũng có điều tiện, vì thế không nên sử dụng thường xuyên. Chỉ sử dụng trong trường hợp "bất đắc dĩ". Đĩa mềm này còn cần thiết cho cấu hình để khởi động nhiều HĐH.

- thứ hai, khi cài đặt Linux cần cài LILO vào sector đầu tiên của phân vùng dành cho thư mục gốc (/) của Linux, chứ không phải vào sector khởi động chính của đĩa (MBR). Chúng ta giả thiết Linux được cài vào phân vùng `/dev/hda3`. Như vậy LILO sẽ được cài vào sector đầu tiên của `/dev/hda3`

Theo nguyên tắc, nếu như bạn đọc cài LILO vào MBR, thì không phải mọi thứ đã hỏng hết. Kết quả cuối cùng (khởi động qua NT Loader) vẫn có thể đạt được nhưng cần bỏ ra một chút công sức. Vấn đề ở chỗ, định dạng MBR tạo bởi LILO và Windows (DOS) khác nhau. Vì thế nếu bạn đọc cài LILO vào MBR, thì cầu phục hồi lại MBR của Windows. Tác giả cũng sẽ nói cách phục hồi, nhưng tốt hơn hết là bạn đọc cài LILO ngay lập tức vào sector đầu tiên của phân vùng đã cài Linux.

4. Sau khi cài đặt xong, khởi động Linux bằng đĩa mềm (nếu như bạn đọc cài LILO vào phân vùng của Linux và không động gì đến MBR, thì đây là khả năng duy nhất).

5. Sao chép sector khởi động của Linux vào một tập tin; tập tin này sẽ cần để trình khởi động Windows NT/2000 có thể khởi động Linux. Việc sao chép thực hiện như sau: đầu tiên gắn một đĩa mềm trắng (mới mua thì càng tốt),

   ```
   [root]# mount -t vfat /dev/fd0 /mnt/floppy
   ```

   chuyển vào thư mục /mnt/floppy

   ```
   [root]# cd /mnt/floppy
   ```

   và thực hiện câu lệnh

   ```
   [root]# dd if=/dev/hda3 of=/mnt/floppy/bootsect.lnx bs=512 count=1
   ```

   để ghi nội dung sector khởi động của đĩa `/dev/hda3` vào tập tin `/mnt/floppy/bootsect.lnx`[4]

6. Tiếp theo cần khởi động lại để vào Windows NT, bằng câu lệnh:

   ```
   [root]# shutdown -h now
   ```

[4]Ghi chú: nếu đĩa C: (/dev/hda1) có định dạng FAT, thì có thể tạo tập tin bootsect.lnx trong thư mục gốc của đĩa C:. Tác giả không biết (chưa thử) có thể khởi động không cần đĩa mềm không, nếu phân vùng chính đầu tiên có định dạng NTFS. Tuy nhiên ở đây cũng không có vấn đề gì, chỉ cần sao chép sector khởi động qua đĩa mềm như đang trình bày. Tạm thời nhân Linux còn chưa hỗ trợ tốt việc ghi lên phân vùng NTFS.

Vì MBR chưa có gì thay đổi, nên Windows NT sẽ khởi động. Trong NT cần sao chép tập tin bootsect.lnx vào thư mục gốc của đĩa C:, hay chính xác hơn là vào thư mục gốc của phân vùng mà từ đó khởi động Windows NT. Đây có thể là phân vùng FAT16 hay phân vùng NTFS. Đặc điểm để nhận ra phân vùng này là hai tập tin ntldr và boot.init chứa trong đó (những tập tin này có thể ẩn!). Tập tin bootsect.lnx có thể đặt thuộc tính chỉ đọc (read-only).

7. Sau đó tìm tập tin boot.ini và thêm vào dòng sau:

   ```
   C:\bootsect.lnx="LINUX"
   ```

   (tất nhiên, trong dấu ngoặc kép bạn đọc có thể đặt tên bất kỳ.)

8. Việc còn lại là khởi động lại máy tính một lần nữa, và trong trình đơn chọn hệ điều hành sẽ có LINUX. Nếu chọn LINUX, thì LILO sẽ được chạy và sau đó nó (LILO) sẽ nạp Linux.

Còn bây giờ chúng ta sẽ xem xét trường hợp bạn đọc (do vô tình hay cố ý) cài đặt LILO vào bản ghi khởi động chính của đĩa (Master Boot Record, MBR). Trong trường hợp này bản ghi khởi động Windows NT (hay 2000) sẽ bị xóa, và việc khởi động Windows NT (bước thứ 6 ở trên) là không thể. Nếu như bạn đọc vẫn còn muốn sử dụng trình khởi động OS Loader của NT, chứ không muốn dùng LILO, thì những bước trên có thay đổi một chút: thay cho bước thứ 6 cần làm các thao tác sau.

1. Khởi động Windows NT từ đĩa mềm khởi động (đã tạo trước khi cài đặt Linux, nếu không có thì bạn đọc cần tìm một máy khác đang chạy Windows NT rồi tạo). Trong trình đơn (thực đơn) của trình khởi động cần chọn lệnh Recover, rồi chọn chế độ Command mode. Sau đó đăng nhập vào tài khoản nhà quản trị (administrator).

2. Phục hồi lại bản ghi khởi động chính của đĩa. Sử dụng câu lệnh fdisk /mbr. Tác giả dùng lệnh này thành công, mặc dù trong một số bài báo nói cách phục hồi MBR như vậy không phải lúc nào cũng làm việc. Trong Windows 2000 có các lệnh chuyên dùng fixboot và fixmbr (chạy từ console phục hồi hệ thống). Chạy hai lệnh này theo thứ tự đã chỉ ra. Sau đó Windows 2000 sẽ khởi động bình thường.

3. Khởi động lại máy tính từ đĩa mềm khởi động Linux và đăng nhập vào hệ thống với quyền người dùng root.

4. Nhập lệnh `cd /etc` và mở tập tin `lilo.conf`. Ở đầu tập tin này có liên kết đến phân vùng khởi động theo mặc định, ví dụ, `/dev/hda`.

5. Dùng bất kỳ trình soạn thảo nào, ví dụ, CoolEdit của Midnight Commander, để thay thế giá trị này thành phân vùng đã cài Linux lên (chính xác hơn là thành phân vùng được gắn như gốc (/) của Linux). Nếu Linux được cài vào phân vùng `/dev/hda3`, thì cần ghi cái đó, tức là thay thế `/dev/hda` thành `/dev/hda3`. Nếu như bạn đọc không nhớ đã cài Linux vào đâu, thì hãy chạy câu lệnh `mount` và tìm kết quả tương tự như sau[5]:

[5] có nghĩa là tìm phân vùng đã gắn vào thư mục gốc /, trong ví dụ này là /dev/hda3

```
/dev/hda3 on / type reiserfs (rw)
```

6. Chạy lệnh `/sbin/lilo` để ghi trình khởi động vào phân vùng /dev/hda3 (cần chạy lệnh `lilo` không có tham số). Sẽ có cảnh báo về việc phân vùng không phải là đầu tiên trên đĩa. Đây chính là điều chúng ta cần, bản ghi khởi động của Windows được giữ nguyên vẹn.

7. Thực hiện các bước 6-8 như ở trên.

Dễ dàng đoán ra rằng, "quy trình" phức tạp với hai lần khởi động lại chỉ để chuyển sector khởi động Linux từ MBR vào sector đầu tiên của phân vùng dành cho Linux, và phục hồi MBR của Windows.

Quá trình cài đặt Linux kết thúc ở đây. Bạn đọc đã có thể chọn HĐH sẽ khởi động và điều khiển máy tính của mình.

2.7 Sử dụng trình khởi động GRUB[6]

Như đã nói trong phần lựa chọn chương trình khởi động, nếu trên máy đã cài Windows 98 với hệ thống tập tin FAT16, thì lựa chọn tốt hơn cho trình khởi động là chương trình có trong thành phần của mọi bản phân phối HĐH Linux – GRUB (**GR**and **U**nified **B**ootloader).

2.7.1 Cài đặt GRUB

Cách dễ nhất để có thể cài đặt trình khởi động GRUB là chọn dùng trình khởi động này trong quá trình cài đặt hệ thống Linux của bạn. Trong những phiên bản mới của các bản phân phối Linux mới khi này có thể chọn cấu hình cho những hệ điều hành đã có trên máy. Một số bản phân phối, ví dụ Xandros còn tự động tìm thấy những hệ điều hành đã có này và thêm chúng vào tập tin cấu hình của GRUB. Nhờ đó sau khi cài đặt bạn có thể chọn khởi động Linux hoặc hệ điều hành (thường là Windows) đã có. Mặc dù việc tự động cấu hình này hết sức thuận tiện, nhưng sẽ có ích nếu bạn biết được cách cấu hình Linux để xử lý những lỗi có thể xảy ra và hơn thế nữa biết cách cấu hình để khởi động hệ điều hành mong muốn khi không có tự động cấu hình. Chúng ta sẽ xem xét cấu hình GRUB trong phần tiếp theo.

2.7.2 Cấu hình GRUB

GRUB có một giao diện trình đơn để từ đó người dùng có thể chọn một mục (một hệ điều hành) bằng các phím mũi tên rồi nhấn <Enter> để khởi động. Để dùng được trình đơn đó, bạn cần một tập tin cấu hình '`menu.lst`' nằm trong thư mục khởi động `/boot`. Thông thường tập tin này được tạo ra khi cài đặt. Hãy xem ví dụ một tập tin như vậy dưới đây:

```
# GRUB example configuration file on the teppi's openSUSE system.
# Modified by YaST2. Last modification on Sun Apr 2 22:22:11 MSD 2006
```

[6]Phần về GRUB này do người dịch viết

```
color white/blue black/light-gray
default 0
timeout 8

###Don't change this comment - YaST2 identifier: Original name:linux###
title SUSE LINUX 9.3
    kernel (hd0,0)/vmlinuz root=/dev/hda5 vga=0x31a selinux=0
splash=silent resume=/dev/hda7  showopts
    initrd (hd0,0)/initrd

###Don't change this comment - YaST2 identifier: Original name:
windows###
title Windows
    map (hd0)    (hd1)
    map (hd1)    (hd0)
    rootnoverify (hd1,0)
    makeactive
    chainloader +1

###Don't change this comment - YaST2 identifier: Original name: xen###
title XEN
    kernel (hd0,0)/xen.gz dom0_mem=196608
    module (hd0,0)/vmlinuz-xen root=/dev/hda5 vga=0x31a selinux=0
splash=silent resume=/dev/hda7 showopts
    module (hd0,0)/initrd-xen

###Don't change this comment - YaST2 identifier: Original name: floppy###
title Floppy
    root (fd0)
    chainloader +1
```

Tác giả nghĩ rằng cần giải thích cụ thể hơn một chút về tập tin này. Có thể bạn đã đoán ra những dòng bắt đầu bằng ký tự '#' là những dòng chú thích. Bạn có thể ghi bất kỳ câu gì mong muốn vào dòng này và không ảnh hưởng gì đến công việc của chương trình. Ngoài dòng chú thích GRUB cũng bỏ qua những dòng trống.

Ở đầu tập tin là những thiết lập chung (global options), những tùy chọn có liên quan đến giao diện của trình đơn. Chúng được đặt trước các mục bắt đầu bằng '`title`'. Tùy chọn `default` chỉ ra hệ điều hành khởi động theo mặc định (nếu không động vào bàn phím khi khởi động). Số '0' chỉ ra đó là hệ điều hành thứ nhất trong danh sách (tức là SuSE Linux 9.3). Hãy nhớ lại, trong GRUB việc đánh số được bắt đầu từ '0' chứ không phải '1'. Tùy chọn `timeout` cho biết GRUB sẽ khởi động một cách tự động sau khoảng thời gian chỉ ra (tính theo giây) nếu không nhấn phím nào. Trong trường hợp này là sau 8 giây. Tùy chọn `color` dùng để chọn màu cho trình đơn (hãy thử nó!).

Sau những thiết lập chung là phần thiết lập cho từng hệ điều hành cụ thể. Mỗi phần đều bắt đầu bằng lệnh '`title`' theo sau là tên sẽ hiển thị trong trình đơn. Và vì lệnh này hiển thị tham số một cách nguyên vẹn, nên bạn có thể nhập bất kỳ thứ gì vào phía sau. Như bạn thấy trong tập tin cấu hình ở trên, phần đầu tiên là để khởi động "SUSE LINUX 9.3", phần thứ hai – "Windows", v.v...

Khác với trường hợp LILO, sau khi thay đổi cấu GRUB không cần phải chạy bất kỳ lệnh nào để những thay đổi này có hiệu lực. Chúng sẽ được cập nhật một

cách tự động trong lần khởi động sau. Đây là một điểm mạnh của GRUB so với LILO.

Tất nhiên đây không phải là tất cả những khả năng sử dụng của GRUB, nhưng đối với người dùng mới thì như vậy là đủ. Sau một thời gian làm việc với Linux và có đủ kinh nghiệm bạn sẽ biết cách tìm thêm thông tin về trình khởi động tuyệt vời này.

2.8 Sử dụng trình khởi động LILO

2.8.1 Cài đặt và cấu hình LILO

Giống như trường hợp Windows NT, chúng ta sẽ đưa ra các bước cần thực hiện để có thể khởi động nhiều HĐH.

1. Trước khi cài đặt Linux hãy chuẩn bị đĩa mềm khởi động Windows.

2. Dùng chương trình Partition Magic để lấy phần không gian đĩa còn trống và trên đó tạo ra một phân vùng ext2(3) (hệ thống tập tin Linux) và một phân vùng swap. Cách chia ổ đĩa đã nói ở trên. Nếu dung lượng ổ đĩa cứng vượt quá 8,4 Gbyte thì hãy đọc kỹ các phần 2.3 và 2.5.

3. Cài đặt Linux theo chỉ dẫn đi kèm với bản phân phối. Cần nhớ rằng, nếu bạn muốn sử dụng trình khởi động LILO, thì trong quá trình cài dặt Linux cần chọn phương án cài LILO vào bản ghi khởi động chính (Master Boot Record). Tạo các đĩa mềm khởi động theo nguyên tắc là không bắt buộc, nhưng tác giả khuyên bạn đọc nên làm.[7]

4. Bước tiếp theo cần cấu hình LILO để có thể khởi động các HĐH theo lựa chọn. LILO được cấu hình bằng tập tin `/etc/lilo.conf` và câu lệnh `/sbin/lilo`. Câu lệnh này dùng để cài đặt (hay cài đặt lại) LILO.

   Chúng ta xem xét một ví dụ nhỏ của tập tin cấu hình LILO. Trong ví dụ này chúng ta sẽ coi như thiết bị `/dev/hda1` là phân vùng với DOS/Windows, còn phân vùng `/dev/hda2` chứa Linux. Trong trường hợp đó `/etc/lilo.conf` có dạng gần như sau:

```
boot = /dev/hda2
compact
delay = 50
# message = /boot/bootmesg.txt
root = current
image = /boot/vmlinuz-2.4.22
label = linux
```

[7] Ghi chú. Trình khởi động LILO không bắt buộc phải cài đặt vào bản ghi khởi động chính của đĩa, LILO có thể nằm ở bản ghi khởi động của phân vùng chính được kích hoạt và chứa thư mục gốc của Linux hoặc thậm chí trên phân vùng lôgíc trong phân vùng mở rộng. Trong trường hợp đó MBR cần phải có khả năng nạp LILO, ví dụ khi MBR là trình khởi động của MS-DOS hay Windows. Tuy nhiên tác giả chưa nhìn thấy sự cần thiết của ứng dụng này (nếu đã chọn LILO làm trình khởi động chính thì hãy sử dụng cho "trọn bộ"), vì thế chúng ta sẽ không xem xét đến.

```
read-only
other = /dev/hda1
table = /dev/hda
label = dos
```

Vài lời giải thích cho ví dụ: Dòng `boot` cho biết thiết bị khởi động.

Dòng `compact` bật chế độ nén tập tin map – tập tin chứa đặc tính của nhân được khởi động; tính năng (nén) này tăng tốc độ của khởi động đầu.

Câu lệnh `message` dùng để đưa ra thông báo theo ý muốn khi khởi động.

Bắt đầu từ dòng `image` là các phần nhỏ của tập tin cấu hình, mỗi phần tương ứng với một hệ điều hành sẽ khởi động theo lựa chọn của người dùng. Trong mỗi phần như vậy có một dòng `label`. Trên dòng này ghi tên cần nhập vào dấu nhắc LILO hay tên sẽ hiển thị trong trình đơn của LILO để có thể chọn HĐH muốn khởi động. Nếu như tên không được nhập sau khoảng thời gian chỉ trên dòng `delay` (tính theo phần mười giây – cần nhân với 0,1 giây), thì sẽ khởi động HĐH theo mặc định. Trong ví dụ này, sẽ khởi động Linux theo mặc định, vì phần cấu hình tương ứng với Linux nằm đầu tiên trong tập tin. Có thể chỉ ra hệ điều hành được khởi động theo mặc định khi thêm một dòng có dạng `default=dos`, tức là sử dụng tên đã đặt trên dòng `label`.

Dòng `table=<device>` cho biết tên thiết bị chứa bảng phân chia đĩa. LILO sẽ không đưa thông tin về phân chia đĩa cho hệ điều hành được khởi động nếu biến này không được đưa ra. (Một số hệ điều hành có công cụ khác để xác định là đã được khởi động từ phân vùng nào.) Đừng quên rằng, cần thực hiện câu lệnh `/sbin/lilo`, sau khi thay đổi chỉ dẫn đến bảng phân chia đĩa, tức là thay đổi biến `table`. Nếu đặt dòng (gọi là phần nhỏ thì tốt hơn) `other = /dev/hda1` trong tập tin `/etc/lilo.conf`, thì trong thư mục gốc của đĩa `/dev/hda1` (đĩa C: trong hệ thống thuật ngữ Microsoft) cần có trình khởi động phụ (không phải là chính). Trên một máy của tác giả ở đó nằm trình khởi động NT Loader (vì Windows NT được cài đặt trước Linux), và LILO khởi động thành công Windows NT. Chỉ cần đặt thời gian chờ khởi động trong tập tin boot.ini bằng không, để không thấy trình đơn khởi động của NT Loader. Tuy nhiên, nếu vì một lý do nào đó bạn muốn thấy trình đơn này thì giá trị timeout trong tập tin boot.ini cần đặt khác không (thời gian chờ được tính theo giây). Điều này có thể cần thiết khi muốn khởi động cả Windows 98 từ trình đơn của NT Loader (trong trường hợp này sẽ có 3 HĐH: Linux, Windows NT và Windows 98, trong trình đơn của LILO nếu chọn dos thì sẽ hiện ra trình đơn của NT Loader rồi từ đó chọn một trong hai HĐH Windows để khởi động).

Nếu bạn đọc muốn khởi động Windows trực tiếp từ LILO, thì hãy thêm phần nhỏ sau vào `/etc/lilo.conf`:

```
other = /boot/bootsect.dos
label = win
```

trong đó bootsect.dos lấy từ thư mục gốc của ổ đĩa chứa NT Loader.

5. Sau khi sửa xong tập tin `/etc/lilo.conf` theo ý muốn, cần chạy câu lệnh `/sbin/lilo` để những thay đổi có hiệu lực. Câu lệnh này (trong tài liệu hướng dẫn gọi là map-installer) cài đặt trình khởi động phụ, mà sẽ được kích hoạt trong lần khởi động tiếp theo. Trước khi chạy `/sbin/lilo` để thay đổi bước khởi động, hãy thực hiện câu lệnh này với tham số `-t`. Khi có tham số này sẽ thực hiện tất cả các thủ tục cài đặt trình khởi động, trừ việc thay đổi tập tin map, bản ghi sector khởi động, và bảng phân chia ổ đĩa, tức là chỉ chạy thử cấu hình mới. Nếu cho thêm tùy chọn `-v`, thì bạn đọc sẽ được biết thêm thông tin chi tiết về những gì lệnh `/sbin/lilo` sẽ thực hiện.

   Khi `/sbin/lilo` ghi đè nội dung mới lên sector khởi dộng, thì nội dung cũ của sector này sẽ tự động được ghi nhớ vào một tập tin. Theo mặc định đó là tập tin `/boot/boot.NNNN`, trong đó NNNN tương ứng với số của thiết bị, ví dụ, 0300 – tương ứng `/dev/hda`, 0800 – `/dev/sda`, v.v... Nếu tập tin này đã có trên đĩa, thì nó không bị ghi đè lên. Tuy nhiên có thể đặt một tên khác đẻ ghi sector khởi động, không nhất thiết phải dùng `/boot/boot.NNNN`.

   Tập tin `/boot/boot.NNNN` có thể sử dụng để phục hồi nội dung cũ của sector khởi động, nếu không còn cách phục hồi nào khác đơn giản hơn. Câu lệnh để thực hiện có dạng:

   ```
   [root:~#] dd if=/boot/boot.0300 of=/dev/hda bs=446 count=1
   ```

   hay

   ```
   [root:~#] dd if=/boot/boot.0800 of=/dev/hda bs=446 count=1
   ```

   (bs=446 vì chỉ phục hồi chương trình khởi động, và không động gì đến bảng phân chia đĩa).

   Cũng có thể phục hồi MBR cũ khi cần thiết bằng câu lệnh `/sbin/lilo` với tùy chọn `-u`. Nhưng cần biết rằng, câu lệnh này chỉ làm việc đúng với điều kiện là thư mục LILO (tức là `/boot`) không thay đổi kể từ khi cài đặt.

   MBR của MS-DOS có thể được phục hồi bằng cách khởi động vào DOS từ đĩa mềm (CD) rồi chạy câu lệnh fdisk /mbr (xem trên). Lệnh này chỉ thay đổi mã chương trình khởi động nằm trong MBR, mà không thay đổi bảng phân chia đĩa.

6. Sau khi cài đặt lại trình khởi động cần khởi động lại máy tính và thử các phương án khởi động khác nhau để kiểm tra.

Để kết thúc phần nói về LILO này chúng ta sẽ xem xét vài khó khăn có thể xuất hiện khi sử dụng LILO, và cách khắc phục (nếu có thể).

Khi LILO được nạp, nó đưa ra màn hình từ “LILO”. Khi này mỗi chữ cái biểu thị sự kết thúc một hành động nào đó hay kết thúc một bước nạp LILO. Nếu khởi động bị bị ngưng giữa chừng, thì qua số chữ cái đưa ra có thể nhận định về nguyên nhân xuất hiện vấn đề.

- Không chữ cái nào hiện ra – không có phần nào của LILO được nạp. Hoặc LILO không được cài đặt, hoặc phân vùng chứa LILO chưa được kích hoạt.

- L [mã lỗi] – trình khởi động chính đã được nạp và đã chạy (tức là đã nhận được quyền điều khiển), nhưng nó không thể nạp trình khởi động phụ. Mã lỗi hai ký tự cho biết nguyên nhân cụ thể của vấn đề (cách giải mã cần tìm trong tài liệu kỹ thuật của LILO). Thông thường thì vấn đề nảy sinh do ổ đĩa xấu (có khuyết tật) hay không đặt đúng cấu trúc hình học của đĩa. Nếu LILO không dừng lại ở đây, mà tiếp tục đưa ra một chuỗi vô tận các mã lỗi, thì vấn đề thường dễ giải quyết.

- LI – trình khởi động chính đã nạp được trình khởi động phụ, nhưng không chạy được nó. Có thể là lỗi đưa cấu trúc hình học của đĩa, hoặc tập tin `boot/boot.b` bị di chuyển mà người dùng quên không chạy `/sbin/lilo`.

- LIL — trình khởi động phụ đã được chạy, nhưng nó không thể nạp bảng mô tả từ tập tin map. Nguyên nhân thường do khuyết tật của ổ đĩa hoặc không đưa đúng cấu trúc hình học của đĩa.

- LIL? – trình khởi động phụ đã được nạp vào địa chỉ không đúng. Thông thường do lỗi đưa ra cấu trúc hình học của đĩa hoặc tập tin `/boot/boot.b` bị di chuyển mà người dùng quên không chạy `/sbin/lilo`.

- LIL- – bảng mô tả trong tập tin map bị phá hủy. Thông thường do lỗi đưa ra cấu trúc hình học của đĩa hoặc tập tin `/boot/boot.b` bị di chuyển mà người dùng quên không chạy `/sbin/lilo`.

- LILO – tất cả các phần của LILO được nạp thành công.

2.8.2 Cài đặt các hệ điều hành khác sau Linux

Khi cài đặt MS-DOS và Windows 95/98, trình khởi động của chúng (không phụ thuộc vào ý muốn của bạn đọc) được ghi vào Master Boot Record (MBR), và dấu kích hoạt trong bảng phân vùng sẽ được chuyển sang phân vùng MS-DOS (Windows 95/98). Mà trình khởi động MS-DOS và Windows 95/98 chỉ "biết" chuyển quyền điều khiển cho sector đầu tiên của phân vùng kích hoạt. Như thế, nếu như bạn đọc đầu tiên cài đặt Linux, và sau đó mới cài đặt Windows 95/98 hay MS-DOS, thì Linux sẽ không thể khởi động. Có thể phục hồi lại LILO bằng cách chạy lệnh `/sbin/lilo` (nếu LILO được cài vào MBR), hoặc kích hoạt phân vùng chứa LILO (nếu LILO được cài vào phân vùng chính).

Khi có vấn đề sau khi cài đặt một hệ điều hành khác sau Linux, thường có thể giải quyết bằng cách khởi động vào Linux bằng đĩa mềm khởi động, sửa lại tập tin cấu hình LILO (thêm phần nhỏ cho HĐH mới), rồi chạy `/sbin/lilo`.

2.8.3 Chuyển thư mục /boot lên phân vùng DOS

Những phiên bản nhân Linux mới nhất hỗ trợ khả năng đặt các tập tin cần thiết trên giai đoạn khởi động vào hệ thống tập tin MS-DOS (hay UMSDOS). Vì trong phần lớn các trường hợp phân vùng DOS chiếm các vùng đĩa ở đầu (không có hạn chế của BIOS), nên cho phép giải quyết nhiều vấn đề của ổ đĩa lớn, khi mà thư mục `/boot` không thể nằm trên phân vùng dành cho Linux.

Để thực hiện phương án khởi động này, cần phân vùng DOS ở chế độ đọc/ghi, tạo trong đó một thư mục (ví dụ, `/dos/linux`), và chuyển tất cả các tập tin từ thư mục `/boot` vào đó; thư mục `/boot` được chuyển thành liên kết tượng trưng đến thư mục `/dos/linux`; vị trí mới của thư mục `/boot` cần chỉ ra trong tập tin `/etc/lilo.conf`, và cuối cùng chạy lệnh `/sbin/lilo`.

2.9 Khởi động Linux từ MS-DOS bằng loadlin.exe

Không chỉ các tập tin khởi động và nhân có thể nằm trong phân vùng DOS, mà cả tiến trình khởi động Linux nói chung có thể được tổ chức từ DOS. Khả năng này thực hiện nhờ chương trình đặc biệt loadlin.exe, do Hans Lermen (lermen@elserv.ffm.fgan.de) viết. Chương trình này thường có trên các đĩa của bản phân phối.[8]

Loadlin.exe cung cấp cách khởi động Linux từ ổ cứng an toàn nhất, khi có phân vùng kích hoạt là DOS hay Windows. Phương án khởi động này được khuyên sử dụng cho người dùng Linux mới. Phần lớn người dùng mới (và không chỉ họ) không đủ kiên nhẫn để đọc tài liệu rất hay nhưng rất dài (và lại viết trên tiếng Anh nữa) của LILO. Vì thế họ thường sử dụng LILO không đúng cách, và kết quả là không thể khởi động được bất kỳ hệ điều hành nào (tác giả cũng đã rơi vào trường hợp như vậy). Đối với những người dùng mới thì sẽ thuận tiện hơn khi sử dụng loadlin.exe để khởi động và bắt đầu tìm hiểu Linux. Tuy nhiên, *"không vào hang cọp sao bắt được cọp"*.

Chương trình loadlin.exe không yêu cầu phải cài đặt, chỉ cần đặt chương trình – tập tin loadlin.exe và nhân (ảnh của nhân) lên một trong các đĩa mà DOS nhận ra. Có thể dùng chương trình này để khởi động Linux từ CD hoặc từ ổ đĩa trong mạng mà không cần sử dụng đĩa mềm khởi động. Khả năng này biến loadlin.exe thành một công cụ tuyệt vời để khởi động Linux khi có vấn đề với LILO.

Phiên bản 1.6 của loadlin.exe làm việc với mọi cấu hình DOS và có rất ít hạn chế. Phiên bản này có thể sử dụng bộ nhớ mở rộng; có thể nạp các nhân lớn (các bzImage) và các ảnh đĩa ảo (initrd) trực tiếp vào vùng bộ nhớ nằm trên.

Việc sử dụng loadlin.exe không có nghĩa là Linux làm việc dưới DOS, vì chương trình này hỗ trợ "khởi động lôgíc" của máy tính, và sau đó DOS được thay thế hoàn toàn bằng Linux. Nếu như bạn đọc muốn quay trở lại DOS, thì phải khởi động lại máy tính, ví dụ, bằng câu lệnh `reboot`.

Như vậy, cần làm gì để có thể sử dụng chương trình loadlin.exe.

- Trên máy tính của bạn (tất nhiên với bộ xử lý 386 hoặc cao hơn) cần cài đặt DOS hay WINDOWS 95.
- Cần có ảnh nén của nhân Linux (zImage, bzImage)[9].
- Chương trình loadlin.exe. Có thể tìm thấy trên đĩa phân phối ở dạng không nén hay trong gói LOADLIN16.TGZ. Gói nén này còn chứa hướng dẫn

[8] một số bản phân phối còn có gói cài đặt cho chương trình này, thông thường ghi tập tin loadlin.exe vào thư mục của người dùng root

[9] Ghi chú. zImage là định dạng nhị phân cũ của nhân, bzImage là định dạng mới hơn (số phiên bản nhân lớn hơn 1.3.73) có kích thước lên tới 1 Mbyte, do đó, nhân khi giải nén có kích thước lên tới 2 Mbyte. Tiếp theo chúng ta sẽ chỉ nói về các tập tin zImage, mặc dù bạn đọc có thể thay thế zImage thành bzImage

sử dụng DOC\MANUAL.TXT, tập tin tham số ví dụ DOC\TEST.PAR, và hướng dẫn cách đưa các tham số DOC\PARAMS.DOC (đừng quên rằng chúng là những tập tin DOS).

Nếu bạn chạy loadlin.exe không có tham số

```
C:\LOADLIN> loadlin
```

thì sẽ nhận được hướng dẫn sử dụng ngắn gọn. Thuận tiện hơn để đọc hướng dẫn này khi chạy chương trình với tham số `more` (giống trong Linux):

```
C:\LOADLIN> loadlin | more
```

Bây giờ chúng ta có thể xem xét các bước cài đặt Linux khi sử dụng loadlin.exe để khởi động.

1. Chia các phân vùng cho Linux (cách làm xem phần 2.5)
2. Cài đặt Linux vào phân vùng đã chia. LILO cần cài đặt vào sector đầu tiên của phân vùng Linux để không ghi đè lên MBR và không bị mất khả năng khởi động vào Windows.
3. Sau khi kết thúc phần cài đặt hãy khởi động Linux (nếu không có cách nào khác, hãy sử dụng đĩa mềm khởi động). Gắn phân vùng DOS (giả thiết là `/dev/hda1`, còn phân vùng Linux là `/dev/hda3`):

   ```
   [root]# mount -t vfat /dev/hda1 /mnt/C
   ```

   Tạo thư mục `/mnt/C/loadlin` và giải nén tập tin LOADLIN16.TGZ vào đó. Ngoài ra sao chép từ thư mục `/boot` vào thư mục đó cả tập tin chứa ảnh của nhân Linux. Có thể tìm tập tin chứa ảnh của nhân cần thiết nhờ tập tin `/etc/lilo.conf`: tìm trong tập tin cấu hình dòng "image=..." và bạn đọc sẽ thấy tên cần thiết ở phía bên phải dấu bằng. Ví dụ tên của tập tin này là vmlinuz-2.4.22. Sao chép tập tin vmlinuz-2.4.22 từ `/boot` vào `/mnt/C/loadlin` và đổi tên thành vmlinuz (việc đổi tên là không nhất thiết, và chỉ có ý nghĩa thuận tiện cho sử dụng):

   ```
   [root]# cp /boot/vmlinuz-2.4.22 /mnt/C/loadlin/vmlinuz
   ```

4. Bây giời khởi động lại máy tính vào DOS trực tiếp (nếu bạn đọc có Windows 95/98 thì cần nhấn phím <F8> khi khởi động để hiển thị trình đơn cho phép chọn DOS), hoặc qua hộp thoại lựa chọn khi nhấn lệnh tắt máy (shutdown) Windows 95/98.

   Sau khi vào DOS hãy chuyển sang thư mục C:\LOADLIN:

   ```
   CD \LOADLIN
   ```

   rồi thực hiện lệnh:

   ```
   C:\LOADLIN> LOADLIN vmlinuz /dev/hda3 ro vga=ask
   ```

hoặc, nếu bạn đọc muốn nạp nhân cùng với ổ đĩa RAM:

```
C:\LOADLIN> LOADLIN vmlinuz /dev/ram rw initrd=diskimage
```

Còn có thể ghi tất cả các tham số của lệnh loadlin.exe vào tập tin (ví dụ với tên params) và gọi câu lệnh đó ở dạng sau:

```
C:\LOADLIN> LOADLIN @params
```

Khả năng này đặc biệt có ích khi đưa nhiều tham số dòng lệnh và khi độ dài của dòng lệnh lớn hơn 127 ký tự. Mô tả đầy đủ tất cả các tham số của câu lệnh loadlin.exe có thể tìm thấy trong tập tin PARAMS.DOC hoặc Internet trên trang http://sunsite.unc.edu/mdw/HOWTO/BootPrompt-HOWTO.html và http://rsphy1.anu.edu/~gpg109/BootPrompt-HOWTO.html.

Bây giờ bạn đọc có thể sử dụng cách này để khởi động Linux. Công việc duy nhất còn lại là làm sao để không phải gõ lệnh loadlin với tất cả các tham số sau mỗi lần khởi động lại. Có thể viết thêm lệnh gọi loadlin vào tập tin autoexec.bat hoặc tạo một tập tin lệnh (ví dụ, linux.bat), và khi chạy tập tin này, máy tính sẽ chuyển vào chế độ DOS trước, sau đó thì chạy Linux. Tác giả cho rằng những thông tin đã đưa ra ở trên đủ để tạo tập tin bat cần thiết. Cần nói thêm rằng, không được khởi động Linux từ giao diện đồ họa DOS/Windows và cần tắt một vài tùy chọn trong tập tin ẩn C:\MSDOS.SYS (đây là tập tin văn bản thông thường), bằng cách thêm vào hai dòng sau (nếu chưa có):

```
BootGUI=0
Logo=0
```

Dòng đầu tiên tắt giao diện đồ họa, và DOS sẽ được khởi động thay cho Windows 95/98. (Để chạy giao diện đồ họa, cần nhập câu lệnh C:\> win). Dòng `Logo=0` tắt việc hiển thị biểu tượng Windows (cửa sổ). Vấn đề ở chỗ, đối với một số cạc màn hình Linux sẽ đưa ra một màn hình trống rỗng sau khi khởi động, nếu như có hiển thị biểu tượng Windows.

Chương 3

Khởi động Linux lần đầu

Vạn sự khởi đầu nan – Trung Quốc

Người dịch: *Sau khi cài đặt rất có thể bạn đọc sẽ không biết bước tiếp theo phải làm gì. Bật máy tính lên, đợi hệ thống khởi động xong và ... ngồi nhìn. Phải làm gì để có thể tiếp tục làm việc với hệ thống Linux? Sau khi làm việc xong thì tắt máy thế nào? Có những chương trình nào nên biết? Trong khi làm việc có vấn đề nảy sinh thì tìm câu trả lời ở đâu? Trong chương này chúng ta sẽ tìm thấy câu trả lời cho những câu hỏi trên.*

3.1 Khởi động HĐH Linux

Như vậy là việc cài đặt Linux đã hoàn thành[1]. Nếu người dùng đã cài đặt GRUB làm trình khởi động, thì sau những dòng chữ BIOS thường đưa ra sẽ xuất hiện màn hình dạng như trong hình 3.1.

Nếu không chạm vào bàn phím, thì trên màn hình sẽ xuất hiện rất nhiều thông báo mà tạm thời chúng ta sẽ không xem xét ý nghĩa của chúng[2]. Cuối cùng xuất hiện màn hình cho phép người dùng đăng nhập vào hệ thống. Nếu bạn đã chọn tự động chạy chế độ đồ hoạ thì hãy nhấn <Ctrl>+<Alt>+<F1> để thấy màn hình đăng nhập như sau:

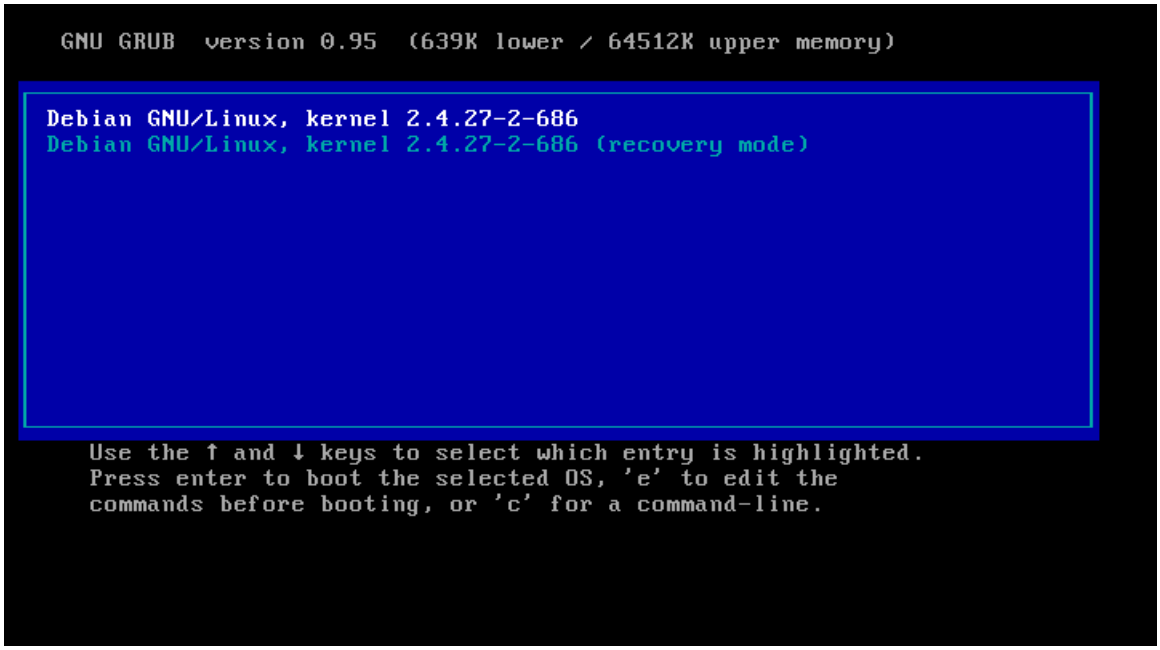
```
ThinhQuyen login:
```

Màn hình đăng nhập này xuất hiện trên máy của tác giả khi khởi động Debian Linux. Tất nhiên màn hình đăng nhập của bạn đọc sẽ khác. Nếu có đủ kiên nhẫn để đọc hết cuốn sách này, thì bạn sẽ biết cách thay đổi màn hình này, ví dụ đưa ra những câu chào dạng "*Xin chào! Hôm nay là 02 tháng 09 năm 2006. Hãy nhập vào tên người dùng và mật khẩu*". Tuy nhiên bây giờ vẫn còn sớm để nói về vấn đề này. Nếu khởi động bằng đĩa mềm thì quá trình cũng tương tự nhưng chậm hơn một chút.

Nếu Linux không phải là hệ điều hành (HĐH) duy nhất trên máy tính, thì trên màn hình khởi động GRUB sẽ xuất hiện danh sách các hệ điều hành tìm

[1] Người dịch: Thông tin trong phần này của bản gốc tiếng Nga đã hơi cũ và trình khởi động bản gốc sử dụng là LILO. Bản dịch đã cập nhật lại và sử dụng trình khởi động GRUB

[2] Trong những bản phân phối Linux mới, ví dụ SuSE Linux, có sử dụng màn hình flash. Do đó có thể bạn đọc sẽ không thấy những thông báo này. Trong trường hợp đó, hãy nhấn phím tương ứng để chuyển về màn hình khởi động bình thường. Phím này thường là <Esc>.



Hình 3.1: Màn hình khởi động của GRUB

thấy. Có thể chọn HĐH muốn khởi động bằng các phím <↑> và <↓> rồi nhấn phím <Enter>. Nếu đã chọn khởi động Linux thì cuối cùng người dùng phải thấy dòng `login:`, tức là dòng mời nhập vào tên đăng nhập.

3.2 Đăng nhập vào hệ thống

Như bạn đã biết, khi hiện ra dòng mời này thì đầu tiên cần nhập vào tên người dùng, sau đó nhập vào mật khẩu khi có yêu cầu để vào hệ thống. Nếu đây là lần đăng nhập đầu tiên vào hệ thống ngay sau khi cài đặt, và trong khi cài đặt bạn quên tạo thêm người dùng, thì hãy nhập vào tên người dùng "**root**" (người dùng cao cấp). Đây là người dùng duy nhất luôn luôn có tài khoản được tạo ra trong quá trình cài đặt. Người dùng này là chủ sở hữu có toàn quyền đối với hệ thống bây giờ cũng như sau này, tức là có quyền truy cập không giới hạn đến các tài nguyên, có thể thêm, xóa những người dùng khác, dừng hệ thống v.v... Những thao tác không cẩn thận của người dùng này có thể dễ dàng dẫn đến những hậu quả khó lường, thậm chí làm hỏng hệ thống. Vì thế thường chỉ đăng nhập dưới tên người dùng này khi thực hiện các công việc quản trị hệ thống. Tuy nhiên chúng ta đang ở trong trường hợp này, vì thế hãy nhập "`root`" vào dòng `login:` rồi nhấn phím <Enter> (<Return>). Hệ thống sẽ đưa ra câu hỏi mật khẩu:

```
Password:
```

Tất nhiên là bạn cần nhập vào mật khẩu tương ứng với người dùng đã đưa ra, ở đây là mật khẩu của root. Mật khẩu này là mật khẩu đã đưa ra cho người dùng root trong quá trình cài đặt. Sau khi nhập mật khẩu cũng nhấn <Enter>. Nếu sau khi nhập vào tên người dùng rồi rất lâu sau không vào mật khẩu, thì hệ thống sẽ tự động quay trở lại dòng hỏi tên người dùng `login:`. Sau khi nhập

đúng mật khẩu, bạn sẽ thấy dòng như sau:

```
[root@ThinhQuyen /root]#
```

Dòng này được gọi là *dấu nhắc*. Xuất hiện dấu nhắc có nghĩa là hệ thống đã sẵn sàng tiếp nhận và thực hiện câu lệnh của người dùng. Ở thời điểm này nó có nghĩa là bạn đã đăng nhập vào hệ thống một cách thành công. Trong MS-DOS và Windows(TM) màn hình đen và dấu nhắc hệ thống thường được gọi là chế độ dòng lệnh. Chúng ta sẽ gọi chế độ này là *văn bản* để phân biệt với chế độ đồ hoạ của hệ thống X Window.

Trong ví dụ đưa ra dấu nhắc gồm tên người dùng (`root`), tên hệ thống (*ThinhQuyen*) và thư mục hiện thời (`/root`). Sau này bạn có thể thay đổi dạng của dấu nhắc này. Trong tất cả các ví dụ tiếp theo chúng ta sẽ sử dụng dấu nhắc chỉ gồm có tên người dùng.

Trước khi đề nghị bạn đọc nhập vào câu lệnh đầu tiên, cần nói rằng trên bất kỳ hệ thống UNIX nào kiểu chữ cũng đều đóng vai trò quan trọng, tức là có phân biệt chữ hoa và chữ thường. Vì thế cần nhập vào tất cả các câu lệnh cũng như tham số của chúng như chỉ ra trong ví dụ, kể cả kiểu chữ. Câu lệnh đầu tiên chúng ta sẽ nhập vào là `useradd`. Sau tên của câu lệnh cần khoảng trắng và sau đó là tên người dùng, ví dụ, `nhimlui`:

```
[root]# useradd nhimlui
```

Ngay sau khi nhấn phím <Enter> để chạy lệnh này, trên hệ thống sẽ biết rằng người dùng `nhimlui` tồn tại (chúng ta nói, "mở tài khoản cho người dùng `nhimlui`"). Tuy nhiên vẫn chưa thể vào hệ thống (thường nói, "đăng nhập") dưới tên người dùng này. Để hệ thống cho phép người dùng mới `nhimlui` làm việc, cần phải "tạo" thêm cho người dùng này mật khẩu bằng câu lệnh:

```
[root]# passwd nhimlui
```

Sẽ xuất hiện dòng:

```
Changing password for nhimlui.
New Password:
```

Hãy nhập vào mật khẩu. Sau khi nhập xong, cũng nhấn phím <Enter>, hệ thống sẽ hỏi nhập mật khẩu một lần nữa để kiểm tra.

```
Reenter New Password:
```

Đừng ngạc nhiên, bạn sẽ không nhìn thấy mật khẩu vào vì hệ thống sẽ không hiển thị gì ra, kể cả dấu sao '*' như thường thấy. Nếu bạn đã nhập đúng tất cả, thì sẽ xuất hiện thông báo thành công dạng:

```
Password changed.
```

và sẽ lại xuất hiện dấu nhắc của hệ thống. Nếu nhập vào mật khẩu không tốt (mật khẩu quá ngắn hoặc quá đơn giản), thì sẽ xuất hiện dòng cảnh báo (dạng `Bad password: too short`), nhưng hệ thống vẫn tiếp nhận mật khẩu và cho phép người dùng mới đăng nhập vào hệ thống.

Như vậy là bạn đọc đã làm quen với hai câu lệnh đầu tiên của hệ thống Linux: `useradd` và `passwd`. Câu lệnh tiếp theo mà bất kỳ người dùng Linux nào cũng cần phải biết đó là câu lệnh `man`. `man` là hệ thống trợ giúp luôn luôn đi kèm với hệ thống Linux. Cần phải nhập vào câu lệnh này với một tham số – tên của câu lệnh khác hoặc từ khóa. Ví dụ:

```
[root]# man passwd
```

Kết quả là bạn sẽ nhận được một văn bản mô tả câu lệnh tương ứng hoặc thông tin về đề tài mà từ khoá đưa ra. Vì thông tin thường không thể nằm gọn trên một màn hình nên khi xem cần sử dụng các phím <PageUp>, <PageDown>, và phím trắng. Hãy nhấn phím <Q> ở bất kỳ thời điểm nào để thoát ra khỏi màn hình xem `man` và quay lại dòng nhập lệnh. Bây giờ xin bạn đọc hãy thử xem thông tin về hai câu lệnh đã đề cập đến ở trên – `login` và `passwd`. Chú ý là có thể xem thông tin về chính bản thân câu lệnh `man`. Hãy chạy lệnh:

```
[root]# man man
```

Rất tiếc theo như tác giả biết thì hiện thời các trang thông tin `man` còn chưa được dịch sang tiếng Việt. Do đó bạn sẽ nhận được những trang này bằng tiếng Anh. Nếu như bạn không có khả năng đọc tiếng Anh, thì hay kiên nhẫn đọc cuốn sách này hoặc một cuốn sách khác cùng đề tài.

Bạn đọc còn có thể thử chạy thêm một vài câu lệnh nữa và theo dõi xem hệ thống làm gì. Ví dụ, hãy thử những câu lệnh liệt kê trong bảng 3.1 (hãy nhập chúng cùng với những tham số có trong bảng).

Bảng 3.1: Những câu lệnh đơn giản của Linux

Câu lệnh	**Mô tả ngắn gọn**
`whoami`	Thông báo tên đăng nhập của bạn
`w` hoặc `who`	Cho biết những người dùng nào hiện đang làm việc trên hệ thống
`pwd`	Cho biết tên của thư mục hiện thời
`ls -l`	Hiển thị danh sách tập tin và thư mục con của thư mục hiện thời
`cd tên_thư_mục`	Chuyển thư mục hiện thời đến `tên_thư_mục`
`ps ax`	Hiển thị danh sách các tiến trình đang chạy

Hãy xem mô tả về những câu lệnh này bằng `man`.

Chúng ta sẽ không xem xét chi tiết danh sách tất cả những câu lệnh có thể gặp. Vì thứ nhất, người dùng sẽ dần dần làm quen với chúng trong quá trình đọc cuốn sách này và thực hành trên HĐH Linux của mình. Thứ hai, có thể đọc về những câu lệnh chính trong bất kỳ cuốn sách nào nói về UNIX.

3.3 Console, terminal ảo và shell

Như vậy là bạn đọc đã có kinh nghiệm làm việc đầu tiên ở chế độ văn bản (hay còn gọi là kênh giao tác "console") của hệ thống Linux. Chúng ta sẽ còn gặp thường xuyên những khái niệm trình lệnh "terminal" và "console", do đó tác giả giải thích kỹ hơn những khái niệm này.

Khi hệ thống UNIX đầu tiên mới được tạo ra, máy tính còn rất to (mainframe) và người dùng làm việc trên những máy tính này qua một hệ thống những giao diện kế tiếp nhau dùng để kết nối các *terminal* ở xa. Terminal – thiết bị dùng để giao tiếp giữa người dùng và máy tính, thường gồm màn hình và bàn phím. Máy tính cá nhân của bạn đọc là máy tính thế hệ mới, do đó chắc sẽ không có terminal ở xa nào kết nối tới, nhưng vẫn có bàn phím và màn hình thực hiện vai trò terminal đối với người dùng. Ngoài ra thêm vào thành phần của terminal bây giờ còn có chuột.

Mainframe có một terminal đặc biệt dành cho nhà quản trị (system administrator) hệ thống, được gọi là *console*. Console thường kết nối tới máy tính không qua giao diện kế tiếp nhau mà qua một ổ cắm riêng (đôi khi thiết bị đưa ra không phải là màn hình mà là một thiết bị in). Vì trên các hệ thống UNIX người ta thường tuân theo các truyền thống, do đó bàn phím và màn hình máy tính cá nhân ngày nay làm việc giống như console ngày xưa. Điểm mạnh của giải pháp này ở chỗ tất cả những chương trình cũ đã dùng để quản trị hệ thống UNIX có thể làm việc không có vấn đề gì trên dạng console mới này.

Tuy nhiên, ngoài console Linux còn cho phép kết nối các terminal ở xa tới máy tính. Và hơn thế nữa còn cung cấp khả năng làm việc với một vài terminal ảo từ một console duy nhất. Hãy nhấn tổ hợp phím <Ctrl>+<Alt>+<F2>. Bạn đọc sẽ lại thấy dòng mời đăng nhập `login:`. Nhưng đây không phải là làm việc lại từ đầu với hệ thống mà bạn đọc chỉ chuyển sang một terminal ảo khác. Ở đây bạn đọc có thể đăng nhập dưới tên người dùng khác. Hãy thử vào hệ thống bằng tên người dùng vừa mới tạo ra (`nhimlui`). Sau đó hãy nhấn tổ hợp phím <Ctrl >+< Alt>+<F1>. Bạn sẽ quay lại màn hình ban đầu. Theo mặc định, hầu hết các bản phân phối Linux lớn (Red Hat, SuSE, Debian,...) mở ra 6 phiên làm việc song song (terminal ảo) như vậy khi khởi động. Như vậy rất thuận tiện cho sử dụng. Để chuyển giữa các terminal ảo chúng ta sử dụng các tổ hợp phím <Ctrl>+<Alt>+<F1> — <Ctrl>+<Alt>+<F6>. Cần nói thêm là khi làm việc ở chế độ văn bản (không phải đồ hoạ) thì cũng có thể sử dụng các tổ hợp phím <Alt>+<F1> — <Alt>+<F6> và thu được kết quả tương tự. Tuy nhiên nếu làm việc ở chế độ đồ hoạ thì không thể không dùng phím <Ctrl>. Do đó, tốt hơn hết là làm quen ngay từ đầu với tổ hợp 3 phím. Nhân tiện cũng cần nói luôn, nếu trong quá trình làm việc bạn đọc không nhớ đang nằm trong terminal ảo nào thì hãy sử dụng câu lệnh `tty`. Lệnh này sẽ đưa ra tên của terminal ở dạng: `/dev/tty2`.

Xin nói ngay lập tức nếu bạn muốn thoát khỏi một trong các terminal thì có thể thực hiện bằng tổ hợp phím <Ctrl>+<D>. Thao tác này không tắt máy và cũng không khởi động lại hệ thống. Đừng quên rằng Linux là HĐH nhiều tiến trình (multiproccess) và nhiều người dùng (multiuser). Một người dùng nào đó dừng công việc không có nghĩa là cần phải tắt máy, còn có những người dùng khác vẫn tiếp tục làm việc. Tổ hợp phím nói trên chỉ đơn giản là dừng phiên làm việc hiện thời của một người dùng, và hệ thống sẽ hiển thị lại trên terminal này

dấu mời đăng nhập (`login:`) mà bạn đọc đã thấy. Cũng có thể dừng phiên làm việc bằng một trong hai câu lệnh `logout` hoặc `exit`.

Bây giờ khi đã biết cách bằng đầu và kết thúc phiên làm việc trên hệ thống, xin bạn hãy thực hiện những lời khuyên ở trên, tức là vào hệ thống dưới tên người dùng bình thường (không có quyền cao "cấp" của người dùng `root`). Hãy đóng tất cả những phiên làm việc mà `root` mở ra, rồi vào hệ thống dưới tên người dùng mới tạo ra.

Còn bây giờ cần nói vài dòng về hệ vỏ. Hệ vỏ hay `shell` (từ này thường không dịch mà để nguyên tiếng Anh) là chương trình thực hiện việc giao tiếp của hệ thống với người dùng. Chính `shell` nhận tất cả những câu lệnh mà người dùng nhập vào từ bàn phím và tổ chức việc thực hiện những câu lệnh này. vì thế `shell` còn có thể gọi là bộ xử lý lệnh (thuật ngữ quen thuộc đối với người dùng DOS). Nói một cách chặt chẽ thì câu "hệ thống hiển thị dấu nhắc" là không đúng, vì dấu nhắc này là do `shell` đưa ra để đợi người dùng nhập vào câu lệnh tiếp theo. Mỗi lần người dùng nào đó vào hệ thống, câu lệnh `login` sẽ chạy cho người dùng này một bộ xử lý lệnh – `shell`. Nếu bạn đọc đăng nhập vào hệ thống từ terminal thứ hai dưới tên người dùng `nhimlui` (hoặc dưới tên người dùng bạn đã chọn), thì hãy chú ý đến sự khác nhau trong dấu nhắc của hai người dùng `root` và `nhimlui`. Dấu nhắc của người dùng `root` có ký tự `#` ở cuối, dấu nhắc của tất cả những người dùng còn lại – ký tự $.

Không chỉ duy nhất lệnh `login` có khả năng chạy `shell`. Bạn chỉ cần nhập vào lệnh `bash` (đây cũng là tên của chương trình hệ vỏ trên phần lớn các hệ thống Linux) và như vậy là đã chạy một `shell` mới. Khi thoát khỏi hệ vỏ mới này (bằng câu lệnh `exit` hoặc tổ hợp phím <Ctrl>+<D>), bạn sẽ quay lại hệ vỏ ban đầu (hệ vỏ mà từ đó bạn đã chạy lệnh `bash`).

Hệ vỏ `bash` không chỉ là bộ xử lý lệnh mà còn là một ngôn ngữ lập trình mạnh. Trong `bash` có đầy đủ các câu lệnh tích hợp (nội bộ) và các toán tử, và ngoài ra còn có thể sử dụng các chương trình khác nằm trên đĩa làm câu lệnh. Có thể xem danh sách các câu lệnh tích hợp trong `bash` bằng lệnh `help`. Hãy thử lệnh này! Xem thông tin chi tiết về một lệnh nào đó cũng bằng lệnh `help` này với tham số là tên của lệnh, ví dụ: `help cd`. Vì hệ vỏ `bash` đóng một vai trò hết sức quan trọng trong Linux, nên tác giả sẽ dành riêng một chương của cuốn sách này để nói về nó. Tất nhiên là bạn có thể tìm thấy những thông tin tương tự trong bất kỳ cuốn sách nào về UNIX. Cũng cần lưu ý là đối với các hệ thống UNIX các nhà phát triển đã viết ra nhiều hệ vỏ khác thay thế cho `bash`. Cũng có thể sử dụng những hệ vỏ này trên Linux, nhưng theo mặc định sẽ chạy `bash`.

Bây giờ chúng ta sẽ xem xét thêm một câu lệnh nữa mà bạn đọc cần biết. Máy tính của người dùng thông thường là máy cá nhân (personal computer hay nói gọn là PC) dù ở nhà hay ở cơ quan. Có nghĩa bạn đọc cũng là người dùng `root` của hệ thống. Nhưng như đã nói ở trên, đăng nhập dưới tên người dùng cao cấp này là không nên, vì mỗi thao tác không cẩn thận của người dùng này có thể dẫn đến những hậu quả không mong muốn. Khi đăng nhập dưới tên người dùng thông thường, ít nhất bạn đọc cũng không thể xóa hoặc làm hỏng các tập tin hệ thống (system files) do sự không cẩn thận của mình. Trong khi đó, có một loạt các thao tác, ví dụ gắn hệ thống tập tin, chỉ có người dùng cao cấp mới có thể thực hiện. Đừng khởi động lại máy tính mỗi lần như vậy! Câu lệnh `su` giúp đỡ giải quyết những trường hợp như vậy. Chỉ cần nhập câu lệnh `su` và `shell` hiện

thời (hay nói không đúng là "hệ thống") sẽ chạy một `shell` mới mà khi vào trong đó bạn sẽ chạy tất cả các lệnh với quyền của `root`. Tất nhiên là để có quyền này bạn cần nhập mật khẩu của `root` vào dòng yêu cầu hiện ra (`Password:`). Sau khi thực hiện xong các công việc quản trị hệ thống, hãy thoát khỏi hệ vỏ và bạn sẽ trở thành người dùng bình thường với những quyền của mình.

Bằng cách tương tự như vào hệ thống dưới tên `root` ở trên, còn có thể vào hệ thống dưới tên một người dùng bất kỳ mà bạn biết mật khẩu[3] (nói cách khác là "chạy một hệ vỏ `shell` mới dưới tên người dùng khác"). Nhưng cần chỉ ra tên của người dùng này trên dòng lệnh, ví dụ:

```
[user]$ su nhimlui
```

Câu lệnh `su` trước không kèm theo tên nào, theo mặc định sẽ đặt tên người dùng cao cấp `root` vào.

Nhưng trong HĐH Linux còn có thêm một khả năng chuyển tạm thời vào tài khoản của người dùng `root` để thực hiện các chức năng quản trị. Hãy nhớ rằng Linux là hệ thống nhiều người dùng, trên hệ thống có thể làm việc cùng lúc nhiều người dùng. Vì thế có thể làm việc dưới tên người dùng `root` trên terminal ảo thứ nhất, còn trên terminal ảo thứ hai – dưới tên người dùng bình thường. Những công việc thường ngày (soạn thảo văn bản, đọc thư,...) bạn có thể thực hiện bằng tài khoản bình thường, còn khi cần thực hiện các công việc quản trị, bạn sẽ dùng tài khoản người dùng cao cấp (`root`). Để thực hiện lựa chọn này bạn chỉ cần nhấn <Ctrl>+<Alt>+<F1> và sẽ có ngay quyền của nhà quản trị. Sau khi làm xong những thao tác mà chỉ có người dùng cao cấp mới có thể làm, hãy quay lại tài khoản của người dùng bình thường ngay lập tức bằng tổ hợp phím <Ctrl>+<Alt>+<F2>. Như vậy bạn đọc sẽ không có nguy cơ làm hỏng hệ thống khi còn chưa có nhiều kinh nghiệm sử dụng Linux.

3.4 Soạn thảo dòng lệnh. Lịch sử lệnh

Trong những phần trước tác giả đã đề nghị bạn đọc thực hiện một vài lệnh của HĐH Linux. Tác giả cho rằng nếu trong quá trình nhập lệnh có xảy ra lỗi thì bạn có thể đoán được cách sửa chúng. Nhưng dù sao cũng có ích nếu đưa ra danh sách ngắn gọn những câu lệnh (hay nói đúng hơn là phím và tổ hợp phím) dùng để soạn thảo dòng lệnh, cũng như gây ảnh hưởng đến cách làm việc của `shell` bằng bàn phím (bảng 3.2, chúng ta sẽ nói về chuột ở một phần riêng).

Ghi chú: Nếu bạn làm việc trong chương trình Midnight Commander, thì **có thể** sẽ không thể sử dụng những phím như <→>, <←>, <Home>, <End>, <Del> để làm các công việc như bảng trên, vì chúng được Midnight Commander dùng để di chuyển dòng chiếu sáng trong bảng[4] hiện thời. Nhưng một số tổ hợp phím dùng với <Ctrl> và <Esc> thì vẫn dùng được bình thường.

Danh sách những câu lệnh (tổ hợp phím) có thể dùng không chỉ giới hạn trong phạm vi bảng 3.2, nhưng chúng ta chỉ xem xét những câu lệnh đơn giản và cần thiết trong lần làm quen đầu tiên với Linux này. Để có thêm thông tin hãy sử dụng câu lệnh `man bash` hoặc `info bash`.

[3] Người dùng root không cần phải biết mật khẩu để làm việc này.

[4] panel

Bảng 3.2: Những phím soạn thảo dòng lệnh

Phím	**Phản ứng của hệ thống**
<→>, <Ctrl>+<F>	Di chuyển sang phải một ký tự (trong khuôn khổ những ký tự đã nhập cộng thêm một ký tự sẽ nhập)
<←>, <Ctrl>+<B>	Di chuyển sang trái một ký tự
<Esc>+<F>	Di chuyển sang phải một từ
<Esc>+<B>	Di chuyển sang trái một từ
<Home>, <Ctrl>+<A>	Di chuyển về đầu dòng lệnh
<End>, <Ctrl>+<E>	Di chuyển về cuối dòng lệnh
<Del>, <Ctrl>+<D>	Xóa ký tự nằm tại vị trí con trỏ
<Backspase>	Xóa ký tự nằm bên trái con trỏ
<Enter>, <Ctrl>+<M>	Bắt đầu thực hiện câu lệnh
<Ctrl>+<L>	Dọn màn hình và đưa dòng lệnh hiện thời lên dòng đầu tiên
<Ctrl>+<T>	Đổi chỗ hai ký tự: ký tự nằm tại vị trí con trỏ và ký tự nằm bên trái con trỏ, sau đó di chuyển con trỏ sang phải một ký tự
<Esc>+<T>	Đổi chỗ hai từ: từ nằm tại vị trí con trỏ và từ nằm bên trái con trỏ
<Ctrl>+<K>	Cắt phần dòng lệnh bắt đầu từ ký tự nằm tại vị trí con trỏ đến cuối dòng (phần dòng lệnh cắt ra được lưu trong bộ đệm và có thể đặt vào vị trí khác)
<Ctrl>+<U>	Cắt phần dòng lệnh bắt đầu từ ký tự nằm bên trái con trỏ đến đầu dòng (phần dòng lệnh cắt ra được lưu tron bộ đệm và có thể đặt vào vị trí khác)
<Esc>+<D>	Cắt phần dòng lệnh bắt đầu từ vị trí con trỏ đến cuối từ (nếu tại vị trí con trỏ là dấu cách thì cắt toàn bộ từ nằm bên phải nó)
<Esc>+<Del>	Cắt phần dòng lệnh bắt đầu từ vị trí con trỏ đến đầu từ (nếu tại vị trí con trỏ là dấu cách thì cắt toàn bộ từ nằm bên trái nó)
<Ctrl>+<W>	Cắt phần dòng lệnh bắt đầu từ vị trí con trỏ đến dấu cách ở bên trái
<Ctrl>+<Y>	Đặt (dán) đoạn dòng lệnh đã cắt cuối cùng vào vị trí con trỏ
<Esc>+<C>	Chuyển ký tự tại vị trí con trỏ thành viết hoa rồi di chuyển con trỏ tới dấu cách đầu tiên ở bên phải so với từ hiện thời
<Esc>+<U>	Chuyển tất cả các ký tự bắt đầu từ vị trí con trỏ thành viết HOA rồi di chuyển con trỏ tới dấu cách đầu tiên ở bên phải
<Esc>+<L>	Chuyển tất cả các ký tự bắt đầu từ vị trí con trỏ tới cuối từ thành viết `thường` rồi di chuyển con trỏ tới dấu cách đầu tiên ở bên phải
<Shift>+ <PgUp>, <Shift>+ <PgDown>	Những tổ hợp phím này cho phép xem các trang màn hình đã hiển thị. Số lượng những trang này phụ thuộc vào bộ nhớ của cạc màn hình. Có ích khi có câu lệnh nào đó đưa ra màn hình rất nhiều thông tin chạy nhanh qua màn hình, người dùng không kịp thấy chúng
<Ctrl>+<C>	Dừng thực hiện câu lệnh vừa chạy (mà vẫn đang chạy)
<Ctrl>+<D>	Thoát ra khỏi hệ vỏ `bash`

Xin lưu ý bạn đọc trong hệ vỏ `bash` có chương trình tích hợp giúp dễ dành nhập câu lệnh trên dòng lệnh. Gọi chương trình con này bằng một hoặc hai lần nhấn phím <Tab> sau khi nhập một vài ký tự. Nếu những ký tự này là phần đầu của ít nhất một trong những câu lệnh mà `bash` biết, thì có hai khả năng xảy ra. Nếu chúng là phần đầu của duy nhất một câu lệnh, tức là `bash` chỉ tìm thấy có một câu lệnh này, thì hệ vỏ sẽ thêm phần còn lại của câu lệnh này vào dòng lệnh. Nếu `bash` tìm thấy nhiều câu lệnh có phần đầu này, thì sẽ hiển thị danh sách tất cả những phương án có thể chọn. Nhờ đó người dùng có khả năng nhập thêm một vài ký tự nữa làm giảm số phương án chọn xuống còn 1 rồi là dùng phím <Tab> một lần nữa. Nếu số phương án chọn là rất nhiều (ví dụ nhấn phím <Tab> hai lần khi dòng lệnh trống rỗng) thì bạn đọc sẽ nghe thấy tiếng bíp sau lần nhấn <Tab> đầu tiên, và sau lần nhấn <Tab> thứ hai sẽ xuất hiện một dòng dạng `Display all 2627 possibilities? (y or n)` (Hiển thị tất cả 2627 khả năng? cần chọn `y` – có hoặc `n` – không).

Nếu nhấn hai lần phím <Tab> ở sau tên của một câu lệnh và một khoảng trắng, thì hệ vỏ `bash` sẽ coi như bạn đang tìm tên tập tin để dùng làm tham số cho lệnh này, và `bash` sẽ đưa ra danh sách tập tin của thư mục hiện thời. Đây là tính năng trợ giúp của `bash` trong trường hợp người dùng quên tên tập tin trong khi làm việc căng thẳng. Cũng giống như trường hợp câu lệnh, nếu đã nhập vào một phần tên tập tin thì phần còn lại sẽ được tự động thêm vào. Tương tự như vậy có thể đoán phần còn lại của các biến môi trường[5], chỉ cần sử dụng tổ hợp phím <Esc>+<$> thay cho <Tab>.

Trong khi làm việc với hệ vỏ sẽ có ích nếu biết rằng, `bash` ghi nhớ một số câu lệnh (theo mặc định là 1000 lệnh, giá trị này được đặt trong biến `HISTSIZE`, xem chương 5) và cho phép gọi lại chúng bằng cách chọn từ danh sách. Đây được gọi là *lịch sử lệnh*. Có thể xem lịch sử lệnh bằng câu lệnh `history`. Ở đây bạn cần sử dụng các tổ hợp phím <Shift>+<PgUp> và <Shift>+<PgDown> để xem danh sách (có thể) rất dài này. Lịch sử lệnh được lưu trong tập tin xác định bởi biến `HISTFILE` (thường là `$HOME/.bash_history`). Để làm việc với lịch sử câu lệnh trong hệ vỏ `bash` người ta sử dụng những tổ hợp phím trong bảng 3.3.

3.5 Ngừng làm việc với Linux

Mặc dù máy tính làm việc dưới sự điều khiển của HĐH Linux có thể để chạy suốt ngày đêm, nhưng phần lớn người dùng máy tính cá nhân đã quen với việc tắt máy sau khi làm việc xong. Khi làm việc với HĐH Linux không thể tắt máy bằng cách ngắt nguồn điện như đối với MS-DOS. Vì trong bất kỳ thời điểm nào trên hệ thống cũng có rất nhiều quá trình đang làm việc. Bạn có thể thấy điều này bằng cách thực hiện lệnh:

```
[nhimlui]$ ps ax
```

Thực hiện lại lệnh này một lần nữa để xem lại. Nhưng nguyên nhân quan trọng hơn là ở chỗ một số tiến trình này có thể đang làm việc với các tập tin, và hệ thống còn chưa ghi nhớ tất cả các thay đổi với những tập tin lên đĩa mà chỉ lưu

[5] environment variable

Bảng 3.3: Tổ hợp phím điều khiển lịch sử lệnh

Phím	Phản ứng của hệ thống
<↑> hoặc <Ctrl>+<P>	Chuyển tới (gọi vào dòng lệnh) câu lệnh trước trong danh sách (di chuyển ngược lại danh sách)
<↓> hoặc <Ctrl>+<N>	Chuyển tới câu lệnh tiếp theo trong danh sách (di chuyển theo danh sách)
<PgUp>	Chuyển tới câu lệnh đầu tiên trong danh sách lịch sử lệnh
<!>, <N>	Thực hiện (không cần nhấn <Enter> câu lệnh thứ n trong danh sách
<!>, <->, <N>	Thực hiện câu lệnh thứ n tính từ cuối danh sách
<!>, `dòng_ký_tự`	Thực hiện dòng lệnh, có phần đầu trùng với `dòng_ký_tự`. Việc tìm dòng lệnh cần thiết sẽ được thực hiện từ cuối tập tin lịch sử và dòng lệnh đầu tiên tìm thấy sẽ được thực hiện
<Ctrl>+<O>	Cũng giống như nhấn phím <Enter>, sau đó hiển thị câu lệnh tiếp theo trong lịch sử lệnh

tạm chúng trong bộ nhớ (cache). Nếu ngắt nguồn điện thì những thay đổi này sẽ không được lưu và sẽ bị mất, đôi khi có thể dẫn đến không khởi động được máy trong lần sau. Do đó cần biết dừng hệ thống một cách đúng đắn trước khi tắt máy. Công việc này do câu lệnh (chương trình) `shutdown` đảm nhiệm.

Chỉ có người dùng `root` mới có thể thực hiện câu lệnh `shutdown` này[6], do đó bạn cần đăng nhập vào hệ thống dưới tên người dùng này, hoặc dùng câu lệnh `su` để có đủ quyền tương ứng. Câu lệnh `shutdown` có cú pháp như sau:

```
[root]# shutdown <tùy_chọn> <thời_gian> <dòng_thông_báo>
```

Ghi chú: Rất có thể khi chạy lệnh, bạn sẽ nhận được câu trả lời "`bash: shutdown: command not found`". Điều đó có nghĩa là `bash` không biết tìm chương trình ở đây. Trong trường hợp đó bạn cần nhập vào đường dẫn đầy đủ đến chương trình, ở đây là `/sbin/shutdown`, vì tập tin chương trình của `shutdown` nằm tại `/sbin`.

Thường sử dụng hai trong số các tùy chọn của chương trình `shutdown`:

- -h – dừng hoàn toàn hệ thống (**h**alt, sẽ tắt máy)
- -r – khởi động lại hệ thống (**r**eboot).

Tham số `thời_gian` dùng để "hẹn giờ" thực hiện câu lệnh (không nhất thiết phải thực hiện câu lệnh ngay lập tức). Thời gian hẹn giờ được tính từ lúc nhấn phím <Enter>. Ví dụ, nếu bạn muốn khởi động lại sau 5 phút thì hãy nhập vào câu lệnh:

```
[root]# shutdown -r +5
```

Câu lệnh này có nghĩa là "dừng hệ thống sau 5 phút và khởi động lại sau khi hoàn thành công việc". Đối với chúng ta thì tạm thời câu lệnh sau sẽ thích hợp hơn:

[6]Cũng có thể cấu hình để những người dùng khác thực hiện được `shutdown`, ví dụ qua `sudo`.

```
[root]# shutdown -h now
```

Câu lệnh này sẽ tắt máy ngay lập tức. Tương đương với câu lệnh này là lệnh `halt`. Sau khi nhấn tổ hợp phím "nổi tiếng" <Ctrl>+<Alt>+<Del> trên Linux sẽ thực hiện các hành động tương tự với lệnh

```
[root]# shutdown -r now
```

Bằng cách này cũng có thể tắt máy, nhưng cần ngắt nguồn điện trong khi hệ thống bắt đầu khởi động lại.

3.6 Trợ giúp khi dùng Linux

Như vậy là bạn đọc đã kết thúc phiên làm việc đầu tiên với HĐH Linux và tôi mong rằng bạn chưa cần trợ giúp trong một tình huống nào đó. Tác giả cũng mong cuốn sách này sẽ thực hiện vai trò trợ giúp trong thời gian đầu tiên này, nhưng có thể nó không giải quyết được tất cả những vấn đề của bạn. Vì thế tác giả sẽ đưa ra ngay lập tức những nguồn thông tin khác. Nhưng tôi sẽ phải "đi trước" kể về cách nhận thông tin trợ giúp trong giao diện đồ hoạ (X Window). Tác giả coi cách giải quyết này là đúng vì người dùng cần biết trước cách thoát khỏi những trường hợp khó khăn.

3.6.1 Các nguồn thông tin trợ giúp

Nếu rơi vào tình huống mà bạn không biết phải làm gì để có được kết quả mong muốn, thì tốt nhất hãy tìm trợ giúp ở ngay trong hệ thống. Các phiên bản Linux có hàng nghìn trang tài liệu ở dạng tập tin, do đó câu trả lời cho câu hỏi của bạn đã nằm trong lòng bàn tay. Có một vài nguồn độc lập chứa thông tin về hầu hết các mặt của hệ thống Linux:

- các trang trợ giúp `man`
- trợ giúp siêu văn bản `info`
- tài liệu đi kèm với phần mềm
- HOWTO và FAQ của dự án **T**he **L**inux **D**ocument **P**roject (`http://www.tldp.org`)
- câu lệnh `locate`

Cần nói ngay lập tức là phần lớn thông tin từ những nguồn này bằng tiếng Anh. Các dự án dịch chúng sang tiếng Việt còn chưa được tổ chức hoặc còn chưa được hoàn chỉnh. Do đó chúng ta sẽ xem xét từng nguồn thông tin này một cách cụ thể hơn.

3.6.2 Các trang trợ giúp man

Như đã nói ngắn gọn ở trên về câu lệnh man, bằng câu lệnh này người dùng trong hình huống khó khăn luôn luôn có thể tìm trợ giúp về bất kỳ câu lệnh nào của hệ thống, về định dạng tập tin, và về các *gọi hệ thống* (system call). Đây là cách nhận trợ giúp chính trong tất cả các hệ thống UNIX. Các trang trợ giúp man chia thành các phần sau:

Bảng 3.4: Các phần chính của trợ giúp man

Phần	Nội dung
0	Các tập tin header (thường nằm trong /usr/include)
1	Chương trình hoặc câu lệnh của người dùng
8	Câu lệnh dùng để quản trị hệ thống
2	Gọi hệ thống (system call, hàm do nhân cung cấp)
3	Gọi thư viện (library call, chương trình con, hàm trong thư viện của ứng dụng)
4	Thiết bị (tập tin đặc biệt, thường nằm trong /dev)
5	Định dạng tập tin và quy ước, ví dụ /etc/passwd
6	Trò chơi
7	Khác (bao gồm các gói macro và quy ước, ví dụ man(7), groff(7)
9	Nhân (kernel routines)
n	các lệnh Tcl/Tk

Thứ tự liệt kê ở đây không phải là sự nhầm lẫn. Vấn đề ở chỗ các tập tin chứa thông tin của trợ giúp man nằm trong các thư mục con của thư mục /usr/share/man và khi câu lệnh man tìm kiếm thông tin cần thiết, thì nó sẽ xem các thư mục con này theo thứ tự đã chỉ ra trong bảng 3.4. Nếu bạn chạy lệnh

```
[user]$ man swapon
```

thì sẽ nhận được trợ giúp về câu lệnh swapon nằm trong phần 8. Vì thế nếu muốn xem trợ giúp về gọi hệ thống swapon cần chạy lệnh

```
[user]$ man 2 swapon
```

để chỉ ra số thứ tự của phần trợ giúp cần tìm kiếm thông tin.

Các trang man được xem bằng chương trình less (hoặc chương trình xác định bởi biến PAGER), do đó có khả năng xem thông tin theo từng màn hình và di chuyển màn hình này xuống dưới và lên trên và để di chuyển có thể sử dụng các phím như trong chương trình less. Những phím thường dùng nhất là:

Nếu bạn không thích đọc từ màn hình mà cầm tay đọc, thì có thể in ra trang man tương ứng bằng lệnh

```
[user]$ man tên_câu_lệnh | lpr
```

hoặc nếu máy in là postscript thì dùng:

```
[user]$ man -t tên_câu_lệnh | lpr
```

Bảng 3.5: Phím sử dụng để xem trang `man`

Phím	Chức năng
<Q>	Thoát khỏi chương trình
<Enter>	Xem từng dòng
<Space>	Hiển thị màn hình thông tin tiếp theo
<B>	Quay lại màn hình trước
</>, dòng ký tự, <Enter>	Tìm kiếm dòng ký tự chỉ ra
<N>	Lặp lại tìm kiếm vừa thực hiện.

Tuy nhiên để có thể nhận được thông tin mong muốn thì còn cần phải biết chỗ tìm thông tin đó. Trong trường hợp này có thể dùng hai câu lệnh `whatis` và `apropos`. Câu lệnh `whatis` tìm kiếm từ khóa đưa ra trong cơ sở dữ liệu bao gồm danh sách các câu lệnh và mô tả ngắn gọn của chúng. Lệnh này chỉ đưa ra những trùng lặp chính xác với từ khóa tìm kiếm. Câu lệnh `apropos` thực hiện tìm kiếm theo các phần của từ khóa. Tương tự như lệnh `apropos` là câu lệnh `man` với tham số `-k`. Hãy thử chạy lệnh sau:

```
[user]$ man -k net
```

Cần phải nói luôn là để cho các câu lệnh `man -k`, `whatis` và `apropos` làm việc, thì đầu tiên cần tạo ra cơ sở dữ liệu về các câu lệnh có trên máy bằng cách chạy lệnh `makewhatis`. Trong trường hợp ngược lại khi tìm kiếm bạn sẽ nhận được thông báo "nothing appropriate". Chỉ có người dùng `root` mới có quyền chạy câu lệnh `makewhatis`. Nếu bạn đọc để máy chạy cả đêm thì tốt nhất chạy câu lệnh này ở dạng công việc cho tiến trình `cron`[7].

Cuối cùng tác giả muốn nói rằng, các trang trợ giúp `man` không dành cho thời gian làm quen đầu tiên với Linux. Chúng dành cho những người dùng có kinh nghiệm cần có "sổ tay tra cứu" về định dạng, tuỳ chọn và cú pháp của lệnh trong quá trình làm việc để không phải nhớ một số lượng lớn những thông tin này trong đầu.

3.6.3 Câu lệnh `info`

Câu lệnh `info` là dạng trợ giúp thay thế và tương đương với `man`. Để nhận thông tin về một câu lệnh nào đó, thì cũng giống như `man`, cần nhập vào `info` cùng với một tham số là tên của câu lệnh quan tâm. Ví dụ:

```
[user]$ info man
```

Thông tin màn bạn sẽ thấy trên màn hình trong phần lớn trường hợp sẽ khác với những gì mà câu lệnh `man` đưa ra. Và theo ý kiến của nhiều người dùng là về chiều hướng tốt hơn. Nhưng sự khác nhau cơ bản nhất ở chỗ `info` đưa ra thông tin dạng siêu văn bản (hypertext) giống như các trang web. Nhờ đó bạn có khả năng xem các phần khác nhau của trợ giúp mà không cần phải thoát ra khỏi chương trình xem này. Trong khi làm việc ở chế độ văn bản, bạn có thể

[7] `cron` là chương trình để chạy tự động các công việc theo thời gian đã định.

chạy câu lệnh `info` trên một trong các terminal ảo (hãy nhớ đến các tổ hợp phím <Ctrl>+<Alt>+<F2>, <Ctrl>+<Alt>+<F3> v.v...) để có thể chuyển sang terminal ảo đó tìm trợ giúp khi cần thiết. Trong trường hợp bạn không biết tìm thông tin cần thiết ở đâu thì có thể chuyển sang các phần khác nhau bằng các siêu liên kết (hyperlink) mà `info` tạo ra. Những liên kết này được đánh dấu bằng ký tự sao (*), khác với cách đánh dấu liên kết trên các trang Web nhưng vẫn giữ nguyên được sự thuận lợi. Có thể di chuyển qua các liên kết bằng phím <Tab>. Sau khi di chuyển đến liên kết mong muốn, hãy nhấn phím <Enter>. Phím <P> đưa người dùng trở lại trang vừa xem, phím <N> đưa đến trang tiếp theo, còn <U> chuyển lên trên một bậc trong cấu trúc phân bậc của các trang tài liệu này.

Ngoài ra, còn có thể chuyển theo liên kết bằng cách khác tương tự như hệ thống trình đơn. Đầu tiên cần nhấn phím <M>, sau đó nhập vào dòng `Menu item:` ở cuối màn hình một vài ký tự đầu tiên của tên của phần trợ giúp cần thiết. Tên của những phần trợ giúp này được hiển thị trên màn hình. Số ký tự phải đủ sao cho chỉ tương ứng với một phần trợ giúp, nếu không thì chương trình sẽ yêu cầu nhập thêm vào. Thoát ra khỏi `info` bằng phím <Q>.

3.6.4 Câu lệnh `help`

Như đã nhắc đến ở trên, hệ thống trợ giúp về các lệnh tích hợp của hệ vỏ `bash` là câu lệnh `help`. Nếu chạy lệnh `help` không có tham số thì sẽ nhận được danh sách của tất cả các lệnh tích hợp của `bash`. Nếu chạy `help tên`, trong đó `tên` là tên của một trong những câu lệnh nói trên, thì bạn sẽ nhận được giới thiệu ngắn gọn về cách sử dụng câu lệnh này.

3.6.5 Tài liệu đi kèm với bản phân phối và chương trình ứng dụng

Nếu trong quá trình cài đặt không bỏ đi những gói tài liệu, thì sau khi kết thúc bạn sẽ tìm thấy trong thư mục `/usr/share/doc` (hoặc `/usr/doc`) các thư mục con `HOWTO`, `FAQ`,... Những thư mục này chứa tài liệu đầy đủ về hệ thống Linux nói chung cũng như những phần riêng rẽ của nó. Những tài liệu này có ở dạng văn bản ASCII và có thể xem chúng bằng các câu lệnh `more tên` hoặc `less tên` hoặc bằng chương trình xem có trong Midnight Commander.

Phần lớn các chương trình ứng dụng có kèm theo tài liệu hướng dẫn cài đặt và sử dụng. Nếu cài đặt chương trình từ gói (package) dạng `rpm` (Fedora Core, SuSE, Mandriva,...) thì tài liệu sẽ nằm trong thư mục con tương ứng của thư mục `/usr/share/doc`. Tên của những thư mục con này tương ứng với tên của chương trình và phiên bản của nó. Ví dụ, chương trình nhập tiếng Việt mà tôi đang dùng để gõ những dòng này `xvnkb` phiên bản 0.2.9 có thư mục con tương ứng `xvnkb-0.2.9` nằm trong `/usr/share/doc` sau khi cài đặt.

Đôi khi để tìm tập tin trợ giúp mong muốn bạn sẽ cần đến câu lệnh `locate`. Câu lệnh này trong một chừng mực nào đó tương tự với các lệnh `whatis` và `apropos`. Khi chạy `locate` nó sẽ tìm tất cả những tập tin có tên chứa từ khóa đưa ra. Ví dụ `locate net` sẽ tìm tất cả những tên tập tin có tên chứa "net". Những tập tin này có rất nhiều trên máy. Trong từ khóa (mẫu) có thể sử dụng các ký tự thay thế *, ?, []. Tuy nhiên câu lệnh `locate` không tìm kiếm theo các

thư mục của hệ thống tập tin, mà theo cơ sở dữ liệu đặc biệt chứa tên các tập tin được tạo ra (và đôi khi cần cập nhật) bằng lệnh `updatedb`.

Trong một số bản phân phối `locate` được thay thế bởi `slocate` (**secure locate**). `slocate` tự tạo cho mình cơ sở dữ liệu nói trên sau khi chạy với tham số tương ứng.

3.6.6 Câu lệnh `xman`

Đây là chương trình cho phép xem các trang trợ giúp `man` khi làm việc ở giao diện đồ hoạ (GUI). Việc tìm kiếm và hiển thị các trang trợ giúp được thực hiện bằng cách nhấn các nút và trình đơn. Còn lại (theo thông tin hiển thị) `xman` cũng giống như `man`.

3.6.7 Câu lệnh `helptool`

Sau khi chạy lệnh `helptool` sẽ hiện ra một cửa sổ đồ hoạ, có một ô nhập vào để người dùng đưa ra thuật ngữ đang quan tâm. Chương trình sẽ xem tất cả các tập tin tài liệu (bạn có thể cấu hình để chọn những tài liệu nào cần xem khi tìm kiếm). Sau khi hoàn thành tìm kiếm chương trình sẽ hiện ra danh sách những tập tin có chứa thuật ngữ này. Nếu nhấn chuột vào một tập tin trong danh sách thì sẽ hiện ra một cửa sổ khác nội dung của tập tin đã chọn. Khi này tập tin sẽ được hiển thị ở dạng lưu trên đĩa: trang `info`, trang `man`, v.v...

3.6.8 Sách và Internet

Tất nhiên, học Linux dễ dàng và đơn giản nhất khi có một cuốn sách tốt. Trước tiên bạn cần đọc tài liệu hướng dẫn đi kèm với bản phân phối của mình. Các bản phân phối lớn như Debian, SuSE, Fedora,... đều đã có những tài liệu này (rất có thể đã được dịch sang tiếng Việt). Rất tiếc người dịch chưa đọc cuốn sách tiếng Việt nào về Linux do đó không thể giới thiệu với bạn đọc. Tất nhiên nếu bạn có kết nối Internet (bây giờ không còn quá xa xỉ) và một chút tiếng Anh thì có thể tìm được câu trả lời cho mọi câu hỏi của mình. Tôi xin đưa ra một số địa chỉ sau làm bước khởi đầu cho bạn đọc trong biển thông tin vô bờ bến này.

Các trang tiếng Việt

1. `http://vnoss.org` – trang web dành cho người dùng mã nguồn mở (MNM) Việt Nam. Có nhiều thông tin về Linux, tài liệu về Linux, diễn đàn cho phép bạn đặt câu hỏi của mình. Trang web do bác Nguyễn Đại Quý đang sống và làm việc tại Bỉ quản lý.

2. `http://vnoss.net` – tin tức về Linux và MNM.

3. `http://vnlinux.org` – đây là trang web dành cho nhóm người dùng Linux Việt Nam (vietlug). Bạn sẽ tìm thấy nhiều thông tin có ích ở đây và có thể đăng ký tham gia nhóm thư vietlug để đặt câu hỏi. Trang này do anh Larry Nguyễn, một Việt Kiều ở Mỹ, quản lý.

4. http://vnoss.net/dokuwiki/doku.php?id=linux:tailieutiengviet – trên trang này tổng hợp tất cả những tài liệu tiếng Việt về Linux, rất có thể bạn sẽ tìm thấy tài liệu về đề tài mình cần tìm hiểu trên trang này.

5. http://kde-vi.org – trang web của nhóm dịch giao diện KDE sang tiếng Việt.

Các trang tiếng Anh

1. http://www.linux.com

2. http://www.linux.org

3. http://www.linux.org.uk – Trang web Linux của Châu Âu. Do Allan Cox một trong các nhà phát triển Linux hỗ trợ.

4. http://www.tldp.org – Trang web chính chứa tài liệu về Linux. Rất nhiều tài liệu bao gồm HOWTO, FAQ, sách...

5. http://freshmeat.net/ – Thông báo hàng ngày về những chương trình ứng dụng mới ra dành cho Linux. Kho phần mềm khổng lồ cho Linux.

6. http://www.li.org – Tổ chức **L**inux **I**nternational.

7. http://www.linuxstart.com

8. http://oreilly.linux.co – Ở đây bạn sẽ tìm thấy rất nhiều tài liệu.

9. http://www.linuxplanet.com

10. http://www.kde.org – Trang chủ của môi trường làm việc KDE.

11. http://www.gnu.org – Các ứng dụng dành cho Linux, trong đó nổi tiếng nhất là trình soạn thảo `Emacs` (GNU's Not UNIX).

12. http://slashdot.org – Những tin tức mới nhất về công nghệ máy tính trong đó có Linux. Có các bài báo và lời bình của người đọc (không qua kiểm duyệt).

13. http://www.linuxtoday.com – Danh sách dài những tin tức, thông báo quảng cáo và các thông tin khác. Xem trang này bạn sẽ biết phần lớn những sự kiện trong thế giới Linux.

14. http://www.lwn.net – Tin tức hàng tuần về Linux. Thông tin chia thành từng hạng mục: thông tin chung, thương mại, thông tin về nhân Linux, công cụ phát triển chương trình mới, chương trình cho Linux, v.v... Nếu bạn muốn tin tưởng rằng Linux phát triển rất nhanh và muốn nhận trợ giúp của các công ty thương mại lớn thì nhất định phải thăm trang này. Tin tức trên tuần báo này được ban biên tập chú thích rất tốt.

15. http://www.linuxnewbie.org – Trang web tốt cho những người dùng mới.

16. http://www.linuxjournal.com – tạp chí Linux. Thường đăng những bài báo về nhiều đề tài.

17. http://www.linuxgazette.com – tờ báo Linux này sẽ thú vị đối với cả người dùng mới và người dùng có kinh nghiệm.

18. http://www.linuxfocus.org – Tạp chí phi thương mại toàn cầu.

19. http://www.linuxworld.com – Một trang web tốt có rất nhiều bài báo hay.

20. http://www.linux-mag.com – Linux Magazin, một tờ tạp chí rất thú vị.

21. http://www.penguinmagazine.com.

Tất nhiên đây không phải là danh sách đầy đủ những trang web nói về Linux. Những trang nói trên chỉ là điểm khởi đầu để từ đó bạn sẽ tìm được những trang web khác trong biển thông tin Internet. Đi đến đâu là phụ thuộc vào ý muốn của bạn.

Nếu có vấn đề trong lúc cài đặt, thì hãy hỏi dịch vụ khách hàng của phân phối đĩa. Nếu bạn mua đĩa của một công ty chuyên môn thì sẽ có địa chỉ liên lạc của dịch vụ này. Nếu bạn mua đĩa ghi lại hoặc tự ghi đĩa từ tập tin ISO nhận được qua Internet thì rất có thể lỗi cài đặt là do khi ghi đĩa gây ra.

Hãy sử dụng hộp thư điện tử. Bạn nên đăng ký với một vài nhóm thư chung (mailing list) nào đó, ví dụ mailto:vietlug-users@userforge.net. Cách đăng ký còn phụ thuộc vào từng nhóm thư (nói chính xác hơn là phụ thuộc vào máy chủ điều khiển nhóm thư này). Nhưng hiện nay thường có hai cách đăng ký: gửi thư đến một địa chỉ xác định để yêu cầu, đăng ký qua giao diện web. Thông tin này bạn có thể tìm thấy trên trang web giới thiệu về nhóm thư chung. Tuy nhiên bạn cần biết là để đọc được tất cả thư chung thì cần rất nhiều thời gian, và còn phải đọc rất nhiều thư của những người dùng mới khác (ví dụ “Console là gì?”), hoặc thậm chí có cả những lá thư “ngớ ngẩn” (ví dụ “Hôm nay dùng Debian thật vui”) và tất nhiên là phải đọc cả những thư trả lời cho những câu hỏi này của những ai biết một chút gì đó. Do đó nếu muốn bạn có thể xem kho lưu trữ những lá thư này bằng trình duyệt, rất có thể đã có câu trả lời cho câu hỏi của bạn ở đó. Và như vậy bạn không cần phải đăng ký cũng như viết thư vào nhóm thư chung nữa.

Tất nhiên nếu không tìm thấy thì đừng ngại ngần đặt câu hỏi. Người dùng Linux hết sức vui lòng trả lời thư của bạn. Bảo đảm là bạn sẽ nhận được câu trả lời, nếu không hiểu thì còn có thể yêu cầu giải thích thêm.

Khi đặt câu hỏi có liên quan đến hệ thống Linux của bạn, cần luôn luôn thêm vào thư của mình càng nhiều chi tiết càng tốt (nhưng đừng thêm thông tin thừa) bao gồm: tên của bản phân phối (Debian, SuSE, Fedora, hay một cái nào khác), phiên bản nhân, có vấn đề với phần cứng nào (phiên bản, dòng chữ ghi trên mạch điện tử), thông báo nào hiện ra khi có vấn đề. Đừng đòi hỏi người dùng khác gửi câu trả lời thẳng đến địa chỉ của bạn, “viết thư vào nhóm thư chung là tự thể hiện, viết thư điện tử cũng là sự hỗ trợ kỹ thuật. Viết thư thì miễn phí, nhưng sự hỗ trợ kỹ thuật thì không”. Xin hãy luôn nhớ điều đó!

Chương 4

Làm quen với hệ thống tập tin ext3fs[1]

Bây giờ bạn đã biết cách khởi động và dừng hệ thống Linux, đã đến lúc làm quen với một trong những thành phần chính và quan trọng của Linux – đó là hệ thống tập tin. Hệ thống tập tin – là cấu trúc nhờ đó nhân của hệ điều hành có thể cung cấp cho người dùng và các tiến trình tài nguyên của hệ thống ở dạng bộ nhớ lâu dài trên các đĩa lưu[2] thông tin: đĩa cứng, đĩa từ, CD, DVD, v.v...
Mỗi hệ thống tập tin, giống như một cái đĩa ăn, có hai mặt. Một mặt của nó luôn quay về phía người dùng (hay nói chính xác hơn là quay về phía ứng dụng), chúng ta tạm gọi nó là mặt trước. Từ phía mặt trước này người dùng thấy hệ thống tập tin là một cấu trúc lôgíc của các thư mục và tập tin. Mặt còn lại, mà người dùng không thấy, quay về phía chính bản thân đĩa lưu tạo thành một vùng bên trong của hệ thống tập tin đối với người dùng, chúng ta tạm gọi là mặt sau. Mặt này của hệ thống tập tin có cấu trúc không đơn giản chút nào. Vì ở đây thực hiện các cơ chế ghi tập tin lên các đĩa lưu khác nhau, thực hiện việc truy cập (chọn thông tin cần thiết) và nhiều thao tác khác.
Trong chương hiện tại chúng ta sẽ xem xét mặt quay về phía người dùng của hệ thống tập tin. Mặt còn lại sẽ dành cho một chương sách ở sau. Cần nói thêm là chúng ta sẽ xem xét một hệ thống tập tin cụ thể ext3fs, hệ thống tập tin cơ bản của Linux đến thời điểm hiện nay. Còn có những hệ thống tập tin khác nhưng chúng ta sẽ đề cập đến chúng muộn hơn.

4.1 Tập tin và tên của chúng

Máy tính chỉ là công cụ để làm việc với thông tin không hơn không kém. Mà thông tin trên mỗi HĐH được lưu ở dạng tập tin trên các đĩa lưu. Từ phía của HĐH thì tập tin là một chuỗi liên tục các byte với chiều dài xác định. Hệ điều hành không quan tâm đến định dạng bên trong của tập tin. Nhưng nó cần đặt cho tập tin một cái tên nào đó để người dùng (hay nói đúng hơn là chương trình ứng dụng) có thể làm việc với tập tin. Làm sao để người dùng có thể làm việc với tập tin đó là công việc của hệ thống tập tin, người dùng thường không cần quan tâm đến. Vì thế, đối với người dùng thì hệ thống tập tin là một cấu trúc lôgíc của các thư mục và tập tin.

Tên tập tin trong Linux có thể dài 255 ký tự bao gồm bất kỳ ký tự nào trừ ký tự có mã bằng 0 và ký tự dấu gạch chéo (/). Tuy nhiên còn có nhiều ký tự nữa có

[1] Chương này do người dịch viết
[2] Một số tác giả thích dùng thuật ngữ "vật chứa" ở đây.

ý nghĩa đặc biệt trong hệ vỏ `shell` và do đó không nên dùng để đặt tên tập tin. Đó là những ký tự sau:

```
! @ # $ & ~ % * ( ) [ ] { } ’ " \ : ; > < ‘ dấu cách
```

Nếu tên tập tin chứa một trong những ký tự này (không khuyên dùng nhưng vẫn có thể) thì trước nó phải đặt một dấu gạch chéo ngược (\) (điều này vẫn đúng trong trường hợp có chính bản thân dấu gạch chéo ngược, tức là phải lặp lại dấu này hai lần). Ví dụ:

```
[user]$ mkdir \\mot\&hai
```

sẽ tạo thư mục `\mot&hai`. Còn có thể đặt tên tập tin hoặc thư mục với những ký tự nói trên vào dấu ngặc kép. Ví dụ, để tạo thư mục có tên “mot hai ba” chúng ta cần dùng câu lệnh sau:

```
[user]$ mkdir "mot hai ba"
```

vì câu lệnh

```
[user]$ mkdir mot hai ba
```

sẽ tạo ba thư mục: “mot”, “hai” và “ba”.

Làm tương tự như vậy đối với những ký tự khác, tức là có thể thêm chúng vào tên tập tin (thư mục) nếu đưa tên vào trong dấu ngoặc kép hoặc dùng dấu gạch chéo ngược để bỏ đi ý nghĩa đặc biệt của chúng. Tuy nhiên tốt nhất là không sử dụng những ký tự này kể cả dấu cách trong tên tập tin và thư mục, bởi vì có thể gây ra vấn đề cho một số ứng dụng khi cần sử dụng những tập tin như vậy và cả khi di chuyển những tập tin đó lên hệ thống tập tin khác.

Đối với dấu chấm thì không phải như vậy. Trong Linux người dùng thường đặt nhiều dấu chấm trong tên của tập tin, ví dụ `xvnkb-0.2.9.tar.gz`. Khi này khái niệm phần mở rộng tập tin (thường dùng trong DOS) không còn có ý nghĩa gì, mặc dù vẫn dùng phần cuối cùng của tên tập tin sau dấu chấm để làm ký hiệu về các dạng tập tin đặc biệt (`.tar.gz` dùng để ký hiệu các tập tin nén[3]). Trên Linux các tập tin chương trình và tập tin bình thường không phân biệt theo phần mở rộng của tên (trong DOS tập tin chương trình có phần mở rộng exe) mà theo các dấu hiệu khác, chúng ta sẽ đề cập đến ở sau. Dấu chấm có ý nghĩa đặc biệt trong tên tập tin. Nếu nó là dấu chấm đầu tiên trong tên, thì tập tin này sẽ là ẩn (thuộc tính hidden) đối với một số câu lệnh, ví dụ, lệnh `ls` không hiển thị những tập tin như vậy.[4]

Như đã nói ở chương trước trong Linux có phân biệt các ký tự viết hoa và viết thường. Điều này cũng đúng đối với tên tập tin. Vì thế `l4u-0.9.2.tar.gz` và `L4U-0.9.2.tar.gz` có thể nằm trong cùng một thư mục và là tên của các tập tin khác nhau. Điều này lúc đầu có thể gây khó khăn cho người dùng Windows nhưng sau khi quen thì bạn sẽ thấy nó thật sự có ích.

Chúng ta đã quen với việc tập tin được xác định hoàn toàn theo tên của nó. Tuy nhiên nếu nhìn từ phía hệ điều hành và hệ thống tập tin thì không phải

[3] thường gọi theo tiếng lóng là tarball, quả bóng tar

[4] Nhưng lệnh ls -a sẽ hiển thị. Đọc thêm ls(1) để biết chi tiết.

như vậy. Chúng ta sẽ nói kỹ về *mặt sau* của hệ thống tập tin trong một số chương sách sắp tới, nhưng bây giờ cũng cần đề cập đến một chút về chỉ số "inode".

Vấn đề ở chỗ mỗi tập tin trong Linux có một "*chỉ số ký hiệu*" (index descriptor) tương ứng, hay còn gọi là "*inode*" (tạm thời chưa có thuật ngữ tiếng Việt chính xác nên xin để nguyên từ tiếng Anh). Chính inode lưu tất cả những thông tin cần thiết cho hệ thống tập tin về tập tin, bao gồm thông tin về vị trí của các phần của tập tin trên đĩa lưu, thông tin về dạng tập tin và nhiều thông tin khác. Các chỉ số inode nằm trong một bảng đặc biệt gọi là *inode table*. Bảng này được tạo ra trên đĩa lưu cùng lúc với hệ thống tập tin. Mỗi đĩa lưu dù là thật sự hay lôgíc thì đều có một bảng các chỉ số inode của riêng mình. Các inode trong bảng được đánh số theo thứ tự, và chính chỉ số này mới là tên thực sự của tập tin trên hệ thống. Chúng ta sẽ gọi chỉ số này là chỉ số của tập tin. Tuy nhiên đối với người dùng thì những tên như vậy thật sự không thuận tiện. Không phải ai cũng có khả năng nhớ đã ghi gì trong tập tin với số 12081982 (nói chính xác hơn là chỉ có một số rất ít người có khả năng này). Vì thế các tập tin còn được đặt thêm một tên thân thiện với người dùng và hơn thế nữa còn được nhóm vào các thư mục.

Tác giả đưa ra những thông tin ở trên chỉ để nói rằng tên của bất kỳ tập tin nào trong Linux không phải gì khác mà chính là liên kết đến chỉ số inode của tập tin. Vì thế mỗi tập tin có thể có bao nhiêu tên tùy thích. Những tên này còn được gọi là liên kết "cứng" (hard link) (chúng ta sẽ làm quen kỹ hơn với khái niệm liên kết và cách tạo những liên kết này trong chương sau). Khi bạn đọc xóa một tập tin có nhiều tên (liên kết cứng) thì trên thực tế chỉ xóa đi một liên kết (mà bạn chỉ ra trên dòng lệnh xóa). Thậm chí cả khi bạn đọc đã xóa đi liên kết cuối cùng thì cũng không có nghĩa là đã xóa nội dung của tập tin: nếu tập tin đang được hệ thống hay một ứng dụng nào đó sử dụng, thì nó được lưu đến lúc hệ thống (ứng dụng) giải phóng nó.

Để có thể thêm tên khác cho tập tin hoặc thư mục (tạo liên kết cứng), chúng ta sử dụng câu lệnh `ln` ở dạng sau:

```
ln tên_đã_có tên_mới
```

Ví dụ:

```
[user]$ ln projects/l4u/l4u-0.9.2.pdf ~/l4u.pdf
```

Ký tự ~ có ý nghĩa đặc biệt, nó chỉ thư mục cá nhân (home directory) của người dùng, chúng ta sẽ nói kỹ hơn về ký tự này ngay sau đây. Bây giờ có thể dùng `~/l4u.pdf` để thay cho đường dẫn dài hơn `projects/l4u/l4u-0.9.2.pdf`. Chi tiết về câu lệnh `ln` bạn có thể đọc trong trang `man` của nó.

Có thể tìm ra số lượng liên kết cứng đến tập tin (tức là số lượng tên của tập tin) bằng lệnh `ls` với tham số `-l`[5]. Ngay phía sau quyền truy cập đến tập tin là một số cho biết số lượng những liên kết cứng này:

```
[user]$ ls -l
tổng 1280
-rw-r--r-- 1 teppi82 users 81409 2006-09-06 03:43 bash.tex
drwxr-xr-x 2 teppi82 users  4096 2006-09-06 02:16 images
-rw-r--r-- 2 teppi82 users 82686 2006-09-06 14:32 l4u-0.9.2.pdf
-rw-r--r-- 1 teppi82 users  3069 2006-09-06 13:52 l4u.tex
```

[5]Nếu bạn dùng SuSE Linux thì có thể nhập vào lệnh ll.

(Danh sách bị cắt bớt vì không cần thiết).

4.2 Thư mục

Nếu như cấu trúc tập tin không cho phép sử dụng gì khác ngoài tên tập tin (tức là tất cả các tập tin nằm trên một danh sách chung giống như các hạt cát trên bãi biển) thì thậm chí cả khi không có giới hạn về độ dài của tên, rất khó có thể tìm đến tập tin cần thiết. Hãy tưởng tượng bạn có một danh sách khoảng vài nghìn tập tin! Xin đừng nghi ngờ, một hệ thống Linux hoàn chỉnh sẽ có số lượng tập tin còn lớn hơn thế. Vì thế mà các tập tin được tổ chức vào các thư mục, các thư mục có thể nằm trong các thư mục khác, v.v... Kết quả là chúng ta thu được một cấu trúc thư mục có phân bậc bắt đầu từ một thư mục gốc. Mỗi thư mục (con) có thể chứa các tập tin riêng lẻ và các thư mục con của nó.

Cấu trúc phân bậc của thư mục thường được minh hoạ bằng "*cây thư mục*", trên đó mỗi thư mục đó là một nút của "cây", còn tập tin – là các "lá". Trên MS Windows hoặc DOS cấu trúc thư mục như vậy có trên mỗi ổ đĩa (tức là chúng ta có không phải một "cây" mà một "rừng" thư mục) và thư mục gốc của mỗi cấu trúc tập tin được đánh dấu bằng một chữ cái Latinh (và do đó đã có một số hạn chế). Trên Linux và UNIX nói chung chỉ có một cấu trúc thư mục duy nhất cho tất cả các đĩa lưu, và thư mục gốc duy nhất của cấu trúc này được ký hiệu bằng dấu gạch chéo "/". Có thể đưa vào thư mục gốc này một số lượng không hạn chế các thư mục nằm trên các đĩa lưu khác nhau (thường nói là "gắn hệ thống tập tin" hoặc "gắn đĩa lưu").

Tên của thư mục cũng được đặt theo những quy định như đối với tên tập tin. Và nói chung ngoài cấu trúc bên trong của mình thì thư mục không khác gì so với những tập tin thông thường, ví dụ tập tin văn bản (text file).

Tên đầy đủ của tập tin (hoặc còn gọi là "*đường dẫn*"[6] đến tập tin) là danh sách tên của các thư mục bao gồm thư mục chứa tập tin đó và các thư mục mẹ, bắt đầu từ thư mục gốc "/" và kết thúc là bản thân tên của tập tin. Trong đường dẫn này tên của các thư mục con cách nhau bởi dấu gạch chéo "/" dùng để ký hiệu thư mục gốc như đã nói ở trên. Ví dụ `/home/teppi82/projects/l4u/ext3fs.tex` là tên đầy đủ của tập tin tôi đang nhập vào trên máy của mình.

Hệ vỏ `shell` lưu giá trị của "*thư mục hiện thời*", tức là thư mục mà người dùng đang làm việc trong đó. Có một câu lệnh cho biết tên của thư mục hiện thời, đó là lệnh `pwd`. Ghi chú: nếu nói một cách chính xác, thì thư mục hiện thời luôn đi liền với mỗi tiến trình đã chạy (trong đó có hệ vỏ shell), vì thế đôi khi chạy một chương trình nào đó trong `shell` có thể dẫn đến việc thay đổi thư mục hiện thời sau khi chương trình đó hoàn thành công việc.

Ngoài thư mục hiện thời mỗi người dùng còn có một "thư mục nhà" (home directory, phương án dịch "*thư mục cá nhân*" được ưu tiên hơn, và chúng ta sẽ dùng thuật ngữ này trong cuốn sách l4u). Đó là thư mục trong đó người dùng có toàn quyền[7]: có thể tạo và xóa các tập tin, thay đổi quyền truy cập đến chúng, v.v... Trong cấu trúc thư mục của Linux những thư mục cá nhân của người dùng thường nằm trong thư mục `/home` và thường có tên trùng với tên đăng nhập của

[6] ở đây là đường dẫn tuyệt đối

[7] Nói chính xác hơn là: có toàn quyền đến khi nào root chưa thay đổi chúng :).

người dùng đó. Ví dụ: `/home/nhimlui`. Mỗi người dùng có thể làm việc với thư mục của mình bằng ký hiệu `~`, tức là người dùng `nhimlui` có thể làm việc với thư mục `/home/nhimlui/hinhanh` bằng `~/hinhanh`. Khi người dùng vào hệ thống, thư mục cá nhân sẽ trở thành thư mục hiện thời của người dùng này.

Câu lệnh `cd` dùng để thay đổi thư mục hiện thời. Tham số của lệnh này là đường dẫn đầy đủ hoặc đường dẫn tương đối đến thư mục mà bạn muốn dùng làm hiện thời. Khái niệm đường dẫn đầy đủ (tuyệt đối) đã giải thích ở trên, bây giờ chúng ta sẽ nói rõ hơn về khái niệm đường dẫn tương đối. Đường dẫn tương đối đó là danh sách các thư mục cần phải đi qua trong cây thư mục để có thể chuyển từ thư mục hiện thời đến thư mục khác (chúng ta gọi nó là *thư mục đích*). Nếu thư mục đích nằm phía **dưới** trong cấu trúc thư mục, tức là nằm trong một thư mục con, hoặc "cháu", "chắt" nào đó của thư mục hiện thời, thì đơn giản: chỉ cần chỉ ra thư mục con của thư mục hiện thời, sau đó thư mục con của thư mục con (thư mục "cháu"),... cho đến khi nào tới được thư mục đích. Nếu như thư mục đích nằm **cao** hơn trong cấu trúc thư mục, hoặc nằm hoàn toàn trên một "cành" khác của cây thư mục, thì phức tạp hơn một chút. Tất nhiên trong bất kỳ trường hợp nào cũng có thể sử dụng đường dẫn tuyệt đối, nhưng khi đó cần phải nhập vào một đường dẫn rất dài.

Vấn đề này được giải quyết như sau: mỗi thư mục (trừ thư mục gốc) có duy nhất một thư mục mẹ trong cây thư mục. Trong mỗi thư mục có hai bản ghi đặc biệt. Một trong số chúng có ký hiệu là dấu chấm ('.') và chỉ đến chính bản thân thư mục này, còn bản ghi thứ hai có ký hiệu là hai dấu chấm đơn ('..'), nó chỉ đến thư mục mẹ. Chính những dấu hai chấm này được dùng để ghi đường dẫn tương đối. Ví dụ, để dùng thư mục mẹ làm thư mục hiện thời, thì chỉ cần chạy lệnh:

```
[user]$ cd ..
```

Còn để chuyển "leo" lên hai bậc của cây thư mục, rồi từ đó hạ xuống thư mục `vnoss/doc` thì cần chạy lệnh:

```
[user]$ cd ../../vnoss/doc
```

Câu lệnh `ls` dùng để đưa ra màn hình danh sách các tập tin và thư mục con của thư mục hiện thời. Cần lưu ý là trên thực tế câu lệnh `ls` chỉ đưa ra nội dung của tập tin mô tả thư mục này, và không xảy ra bất kỳ nào thao tác làm việc với tập tin của thư mục. Như đã nói ở trên, mỗi thư mục chỉ là một tập tin bình thường, trong đó có liệt kê tất cả những tập tin và thư mục con của thư mục này. Tức là không có các hộp đặc biệt chứa các tập tin, chỉ có các danh sách tập tin thông thường xác định tập tin hiện thời thuộc về một thư mục nào đó.

Nếu chạy câu lệnh `ls` không có tham số thì chúng ta chỉ thấy tên của các tập tin của thư mục hiện thời. Nếu muốn xem nội dung của một thư mục khác, thì cần phải đưa cho câu lệnh `ls` đường dẫn tuyệt đối hoặc tương đối đến thư mục đó. Ví dụ:

```
[user]$ ls projects
BanTin       drupal-vn  KDE-vi  mrtg      Xfce
bashscripts  fluxbox    l4u     others    vim
chem-tex     gnomevi    manvi   SuSE      vnlinux
debian       HocTap     mc      syslinux  vnoss
```

Bản ghi về tập tin trong thư mục tương ứng ngoài tên còn có rất nhiều thông tin về tập tin này. Để thấy được những thông tin chi tiết đó, thì cần dùng các tham số mở rộng khác của câu lệnh `ls`. Nếu chạy câu lệnh `ls` với tham số `-l` thì không chỉ có tên tập tin mà sẽ hiển thị cả dữ liệu về quyền truy cập đến tập tin (chúng ta sẽ nói đến ở sau); số lượng liên kết cứng hay số lượng tên (nếu là thư mục thì ngay từ đầu đã có hai liên kết như vậy là . và .., do đó số này bằng số thư mục con công thêm 2); tên chủ sở hữu tập tin, tên nhóm sở hữu tập tin (xin được gọi tắt là "nhóm tập tin" mặc dù tối nghĩa); kích thước tập tin và thời gian sửa đổi cuối cùng. Một ví dụ minh họa khác:

```
[user]$ ls -l
tổng 1316
-rw-r--r-- 1 teppi82 users  81629 2006-09-08 22:11 bash.tex
-rw-rw-r-- 1 teppi82 users  98135 2006-09-08 13:54 caidat.tex
-rw-r--r-- 1 teppi82 users    783 2006-09-08 21:58 ChangeLog
-rw-r--r-- 1 teppi82 users  20778 2006-09-09 02:48 ext3fs.tex
-rw-r--r-- 1 teppi82 users   2013 2006-09-08 21:34 gioithieu.tex
drwxr-xr-x 2 teppi82 users   4096 2006-09-08 14:25 images
-rw-r--r-- 1 teppi82 users   3267 2006-09-08 23:13 l4u.tex
```

Nếu đưa thêm tham số `-i` thì trong cột đầu tiên sẽ hiển thị chỉ số inode của tập tin. Khi dùng tham số `-t` việc sắp xếp được thực hiện không theo tên mà theo thời gian sửa đổi tập tin. Tham số `-u` dùng để hiển thị thời gian truy cập cuối cùng thay vào chỗ thời gian sửa đổi. Tham số `-r` đảo ngược lại trật tự của sắp xếp (cần phải sử dụng cùng với các tham số `-l` hoặc `-t`). Cần chú ý rằng có thể liệt kê các tham số một cách riêng rẽ như thế này:

```
[user]$ ls -l -i - r
```

hoặc gộp lại như thế này:

```
[user]$ ls -lir
```

Chúng ta dừng mô tả ngắn gọn về câu lệnh `ls` ở đây (chi tiết về lệnh này có thể xem trên các trang `man` hoặc `info` tương ứng) và chuyển sang xem xét các thư mục chính của cấu trúc tập tin trong Linux.

4.3 Công dụng của các thư mục chính

Nếu như bạn đọc đã từng dùng Windows (ví dụ 2000 hay XP), thì biết rằng mặc dù người dùng có toàn quyền tổ chức cấu trúc thư mục, nhưng một số truyền thống vẫn được tuân theo. Ví dụ các tập tin hệ thống thường nằm trong thư mục C:\Windows, các chương trình thường được cài đặt vào C:\Program Files, v.v... Trong Linux cũng có một cấu trúc thư mục kiểu như vậy và thậm chí còn nghiêm ngặt hơn. Hơn nữa có một tiêu chuẩn xác định cấu trúc thư mục cho các HĐH dòng UNIX. Tiêu chuẩn này được gọi là **F**ilesystem **H**ierarchy **S**tandart (FHS). Nếu có mong muốn bạn có thể đọc toàn bộ tiêu chuẩn này tại địa chỉ

http://www.pathname.com/fhs/. Các bản phân phối Linux lớn đều tuân theo tiêu chuẩn này.

Bảng 4.1 dưới đây đưa ra danh sách ngắn gọn những thư mục chính được tạo ra trong cấu trúc tập tin theo tiêu chuẩn nói trên. Ở cột bên trái liệt kê các thư mục con của thư mục gốc, còn cột thứ hai liệt kê một vài (không phải tất cả) thư mục con, còn cột thứ ba cuối cùng đưa ra mô tả ngắn gọn về công dụng của những thư mục này. Mô tả trong bảng này là hết sức ngắn gọn, chi tiết hơn bạn có thể đọc trong tiêu chuẩn FHS có trên http://www.pathname.com/fhs/.

Bảng 4.1: Cấu trúc thư mục của Linux

Thư mục	Công dụng
/bin	Thư mục này gồm chủ yếu các chương trình, phần lớn trong số chúng cần cho hệ thống trong thời gian khởi động (hoặc trong chế độ một người dùng khi bảo trì hệ thống). Ở đây có lưu rất nhiều những câu lệnh thường dùng của Linux.
/boot	Gồm các tập tin cố định cần cho khởi động hệ thống, trong đó có nhân (kernel). Tập tin trong thư mục này chỉ cần trong thời gian khởi động[8].
/dev	Thư mục các tập tin đặc biệt hoặc các tập tin thiết bị phần cứng. Chúng ta sẽ nói đến những tập tin này ở ngay sau trong một phần riêng. Bạn đọc có thể xem qua `man mknod` (mknode(1)).
/etc	Thư mục này và các thư mục con của nó lưu phần lớn những dữ liệu cần cho quá trình khởi động ban đầu của hệ thống và lưu những tập tin cấu hình chính. Ví dụ, trong `/etc` có tập tin `inittab` xác định cấu hình khởi động, và tập tin người dùng `passwd`. Một phần các tập tin cấu hình có thể nằm trong các thư mục con của `/usr`. Thư mục `/etc` không được lưu các tập tin chương trình (cần đặt chúng trong `/bin` hoặc `/sbin`. Dưới đây chúng ta sẽ xem xét công dụng của một vài(!) thư mục con của thư mục `/etc`.
/etc/rc.d	Thư mục này lưu những tập tin sử dụng trong quá trình khởi động hệ thống. Chúng ta sẽ đề cập chi tiết về những tập tin này và quá trình khởi động nói riêng trong một vài chương sắp tới.
/etc/skel	Khi tạo người dùng mới, thì những tập tin trong thư mục này sẽ được sao chép vào thư mục cá nhân của người dùng đó.
/etc/sysconfig	Thư mục lưu một vài (không phải tất cả) tập tin cấu hình hệ thống.
/etc/X11	Thư mục dành cho các tập tin cấu hình của hệ thống X11 (ví dụ, xorg.conf).
/home	Thông thường trong thư mục này là các thư mục cá nhân của người dùng (trừ root).

[8] do đó một số nhà quản trị không tự động gắn phân vùng /boot vào trong quá trình khởi động.

Thư mục	Công dụng
/lib	Thư mục này lưu các thư viện chia sẻ của các hàm mà trình biên dịch C và các môđun (các driver thiết bị) cần. Thậm chí nếu trên hệ thống không có trình biên dịch C nào, thì các thư viện chia sẻ vẫn cần thiết, vì chúng được nhiều chương trình sử dụng. Những thư viện này chỉ nạp vào bộ nhớ khi có nhu cầu thực hiện hàm nào đó, như vậy cho phép giảm kích thước mã chương trình nằm trong bộ nhớ. Trong trường hợp ngược lại thì cùng một mã lặp lại nhiều lần trong các chương trình khác nhau.
/lost+found	Thư mục này sử dụng để phục hồi hệ thống tập tin bằng lệnh `fsck`. Nếu `fsck` tìm ra tập tin mà không xác định được thư mục mẹ thì nó sẽ đưa tập tin đó vào thư mục /lost+found. Vì thư mục mẹ bị mất, nên tập tin sẽ nhận được tên trùng với chỉ số inode của nó.
/mnt	Đây là điểm gắn (mount) những hệ thống tập tin gắn tạm thời. Nếu trên máy tính có đồng thời Linux và Windows (DOS) thì thư mục này thường dùng để gắn các hệ thống tập tin FAT. Nếu bạn thường gắn một vài đĩa lưu động như đĩa mềm, CD, DVD, đĩa cứng ngoài, flash,v.v... thì có thể tạo trong thư mục này các thư mục con cho từng đĩa lưu.
/tmp	Thư mục dành cho các tập tin tạm thời. Ở bất kỳ thời điểm này người dùng root cũng có thể xóa tập tin khỏi thư mục này mà không làm ảnh hưởng lớn đến người dùng khác. Tuy nhiên không nên xóa những tập tin trong thư mục này, trừ khi khi bạn biết rằng tập tin hoặc nhóm tập tin nào đó đang gây ảnh hưởng đến công việc của hệ thống. Hệ thống sẽ tự động dọn dẹp thư mục này theo định kỳ, vì thế không nên lưu ở đây những tập tin mà bạn có thể sẽ cần đến.
/root	Đây là thư mục cá nhân của người dùng cao cấp root. Hãy chú ý là thư mục này không nằm cùng chỗ với thư mục cá nhân của những người dùng khác (trong `/home`).
/sbin	Vì thư mục `/bin` chủ yếu lưu các tập tin thực thi (chương trình và tiện ích của HĐH) sử dụng trong quá trình khởi động và do nhà quản trị chạy. Trong tiêu chuẩn FHS có nói rằng cần đặc trong thư mục này những tập tin thực thi sẽ sử dụng sau khi gắn thành công hệ thống tập tin `/usr`. Ít nhất trong thư mục này phải có `init`, `mkswap`, `swapon`, `swapoff`, `halt`, `reboot`, `shutdown`, `fdisk`, `fsck.*`, `mkfs.*`, `arp`, `ifconfig`, `route`.
/proc	Đây là điểm gắn hệ thống tập tin `proc` cung cấp thông tin về các tiến trình đang chạy, về nhân, về các thiết bị tính, v.v... Đây là hệ thống tập tin ảo. Chi tiết bạn có thể đọc trong `man 5 proc`. Các tập tin đặc biệt của thư mục này sử dụng để nhận và gửi dữ liệu đến nhân.

Thư mục	Công dụng
/usr	Thư mục này rất lớn và cấu trúc của nó nhìn chung lặp lại cấu trúc của thư mục gốc. Trong các thư mục con của `/usr` là tất cả các ứng dụng chính. Theo tiêu chuẩn FHS thì nên dành cho thư mục này một phân vùng riêng hoặc đặt hoàn toàn trên đĩa sử dụng chung trong mạng. Phân vùng hoặc đĩa đó thường gắn chỉ đọc và trên đĩa (phân vùng) là các tập tin cấu hình cũng như tập tin thực thi dùng chung, các tập tin tài liệu, các tiện ích hệ thống và cả các tập tin thêm vào (tập tin dạng include).
/usr/bin	Các chương trình (tiện ích và ứng dụng) thường được người dùng bình thường sử dụng. `/usr/bin/X11` là nơi thường dùng để lưu các chương trình chạy trên X Window. Và đây cũng thường là liên kết đến `/usr/X11R6/bin`.
/usr/include	Thư mục con này lưu mã nguồn của các thư viện tiêu chuẩn của ngôn ngữ C. Người dùng cần có ít nhất là quyền đọc đối với thư mục này. Dù trong trường hợp nào cũng đừng sửa những tập tin trong thư mục này, vì chúng đã được các nhà phát triển hệ thống kiểm duyệt kỹ càng (không lẽ bạn biết về hệ thống tốt hơn các nhà phát triển).
/usr/lib	Trong thư mục này là các thư viện object của các chương trình con, các thư viện động (dynamic library), một số chương trình không thể gọi trực tiếp. Các hệ thống phức tạp (ví dụ Debian Linux) có thể có các thư mục con của mình trong thư mục này. /usr/lib/X11 – nơi thường dùng để đặt các tập tin có liên quan đến X Window và các tập tin cấu hình của hệ thống X Window. Trên Linux đó thường là liên kết mềm đến thư mục `/usr/X11R6/lib/X11`.
/usr/local	Ở đây thường đặt các chương trình và các thư mục con (nội bộ) chỉ dành cho máy tính này, bao gồm: • **/usr/local/bin**. Ở đây thường lưu những chương trình ứng dụng. • **/usr/local/doc** – các tài liệu đi kèm với chương trình ứng dụng. • **/usr/local/lib** – thư viện và tập tin của các chương trình và hệ thống nội bộ. • **/usr/local/man** – các trang trợ giúp man. • **/usr/local/sbin** – các chương trình dành cho nhà quản trị. • **/usr/local/src** – mã nguồn của các chương trình.
/usr/sbin	Thư mục này gồm các chương trình thực thi dành cho nhà quản trị và không sử dụng trong thời gian khởi động.

Thư mục	Công dụng
/usr/share	Thư mục này dùng cho tất cả các tập tin dữ liệu dùng chung và có quyền truy cập là chỉ đọc. Thường dùng để chia sẻ giữa các kiến trúc khác nhau của HĐH, ví dụ i386, Alpha, và PPC có thể dùng chung một thư mục `/usr/share` nằm trên một phân vùng hoặc đĩa chia sẻ trên mạng. Cần chú ý là thư mục này không dùng để chia sẻ giữa các HĐH khác nhau hoặc giữa các phiên bản khác nhau của cùng một HĐH. Tiêu chuẩn FHS khuyên dùng thư mục con cho mỗi chương trình. Những thư mục sau hoặc liên kết mềm sau phải có trong `/usr/share`: `man` (các trang trợ giúp man), `misc` (những giữ liệu tùy theo kiến trúc khác nhau). Chúng ta xem xét một vài thư mục con của thư mục này: • **/usr/share/dict** – các danh sách từ (word list) của tiếng Anh dùng cho các chương trình kiểm tra chính tả như `ispell`. • **/usr/share/man** – các trang trợ giúp man. Mỗi phần của man nằm trong một thư mục con riêng trong thư mục này. • **/usr/share/misc** (đã nói ở trên).
/usr/src	Mã nguồn của các thành phần khác nhau của Linux: nhân, ứng dụng...
/usr/tmp	Một nơi nữa để lưu các tập tin tạm thời. Thông thường đây là liên kết mềm đến `/var/tmp`.
/usr/X11R6	Các tập tin thuộc về hệ thống X Window. • **/usr/X11R6/bin** – các chương trình ứng dụng của hệ thống này. • **/usr/X11R6/lib** – các tập tin và thư viện có liên quan đến X-Window.
/var	Trong thư mục này là các tập tin lưu các dữ liệu biến đổi (**var**iable). Những dữ liệu này xác định cấu hình của một số chương trình trong lần chạy sau hoặc là những thông tin lưu tạm thời sẽ sử dụng sau. Dung lượng thông tin trong thư mục này có thể thay đổi trong một khoảng lớn, vì thư mục giữ các tập tin như bản ghi (log), spool, khóa locking, các tập tin tạm thời, v.v...
/var/adm	Lưu các thông tin về tài khoản và thông tin chuẩn đoán dành cho nhà quản trị.
/var/lock	Các tập tin điều khiển hệ thống dùng để dự trữ tài nguyên.
/var/log	Các tập tin bản ghi (log).

Thư mục	Công dụng
/var/run	Các tập tin biến đổi trong thời gian thực hiện các chương trình khác nhau. Chúng lưu thông tin về số tiến trình (PID) và ghi thông tin hiện ghời (utmp). Tập tin trong thư mục này thường được dọn sạch trong thời gian khởi động Linux.
/var/spool	Tập tin được đặt vào hàng đợt của các chương trình khác nhau, ví dụ: • **/var/spool/at** – các công việc mà `at` đã chạy. • **/var/spool/cron** – tập tin của hệ thống Verb+cron+. • **/var/spool/lpd** — tập tin trong hàng đợi in. • **/var/spool/mail** – tập tin thùng thư của người dùng. • **/var/spool/uucp** – tập tin của hệ thống `uucp`.
/var/tmp	Các tập tin tạm thời.

4.4 Dạng tập tin

Trong các phần trước chúng ta đã xem xét hai dạng tập tin đó là tập tin thông thường và các thư mục. Những trên Linux còn có một vài dạng tập tin nữa. Chúng ta sẽ làm quen với chúng trong phần này.

Như đã nói, đối với hệ điều hành thì tập tin chỉ là một chuỗi các byte liên tục. Nhờ vậy có thể dùng khái niệm tập tin cho các thiết bị và các đối tượng khác. Điều này đơn giản hoá sự tổ chức và trao đổi các dữ liệu, vì có thể thực hiện ghi dữ liệu vào tập tin, chuyển dữ liệu lên các thiết bị và trao đổi dữ liệu giữa các tiến trình bằng cách tương tự như nhau. Trong tất cả các trường hợp này sử dụng cùng một phương pháp dựa trên ý tưởng chuỗi các byte. Do đó ngoài các tập tin thông thường và thư mục, những thành phần sau cũng được Linux coi là tập tin:

- các tập tin thiết bị
- các ống (kênh) có tên (named pipe)
- các socket (tổ với nghĩa như tổ chim)
- các liên kết mềm (symlinks).

4.4.1 Các tập tin thiết bị

Như đã nói, đối với Linux thì tất cả các thiết bị kết nối vào máy tính (ổ cứng, ổ tháo rời, terminal, máy in, máy scan, môđem, bàn phím, chuột, v.v. . .) đều là các tập tin. Ví dụ, nếu cần đưa ra màn hình terminal thứ nhất thông tin nào đó, thì hệ thống thực hiện thao tác ghi vào tập tin `/dev/tty1`.

Có hai dạng thiết bị: *ký tự* (hay còn gọi là các thiết bị trao đổi theo byte) và **khối** (trao đổi theo khối). Sự khác nhau giữa hai dạng này nằm ở cách đọc và ghi

thông tin vào các thiết bị. Các thiết bị ký tự trao đổi thông tin theo từng ký tự (theo từng byte) trong chế độ chuỗi các byte. Ví dụ thiết bị dạng này là terminal. Còn thông tin được đọc và ghi vào các thiết bị khối theo các khối. Ví dụ các ổ cứng. Không thể đọc từ đĩa cứng và ghi lên đó từng byte, trao đổi thông tin với đĩa chỉ có thể theo từng khối.

Trao đổi dữ liệu với các thiết bị trên Linux do các driver thiết bị đảm nhiệm. Những driver này hoặc nằm trong nhân hoặc nằm riêng ở dạng môđun và có thể gắn vào nhân sau. Để trao đổi với các phần khác của hệ điều hành mỗi driver tạo ra một giao diện liên lạc có vẻ ngoài giống như tập tin. Phần lớn những tập tin như vậy đã được tạo sẵn từ trước và nằm trong thư mục `dev`. Nếu nhìn vào thư mục `/dev` (tức là chuyển vào thư mục đó bằng lệnh `cd` rồi chạy `ls`), thì bạn sẽ thấy một lượng khổng lồ những tập tin thiết bị. Bảng 4.2 cho biết những tập tin thường dùng nhất.

Bảng 4.2: Những tập tin thiết bị chính

Tập tin	Ý nghĩa
/dev/console	Console hệ thống tức là màn hình và bàn phím kết nối tới máy tính.
/dev/hd	Các ổ cứng với giao diện IDE. Thiết bị `/dev/hda1` tương ứng với phân vùng đầu tiên của đĩa cứng đầu tiên, đĩa `/dev/hda`, tức là đĩa Primary Master.
/dev/sd	Ổ cứng với giao diện SCSI.
/dev/fd	Các tập tin ổ đĩa mềm. Ổ đầu tiên là `/dev/fd0`, ổ thứ hai là `/dev/fd1`.
/dev/tty	Các tập tin hỗ trợ terminal của người dùng. Tên gọi vẫn được lưu kể từ khi kết nối teletype vào các hệ thống UNIX làm terminal. Trên Linux những tập tin này hỗ trợ các terminal ảo (hãy nhớ lại chương trước).
/dev/pty	Các tập tin hỗ trợ terminal giả. Sử dụng cho các kết nối từ xa qua telnet.
/dev/ttS	Tập tin hỗ trợ làm việc với các cổng kết tiếp nhau (các cổng COM). `/dev/ttS0` tương ứng với COM1 trong DOS, `/dev/ttS1` tương ứng với COM2.
/dev/cua	Các tập tin cho môđem.
/dev/null	Thiết bị này có thể gọi là "lỗ đen". Tất cả những gì ghi vào `/dev/null` sẽ mất vĩnh viễn. Những người viết script thường chuyển những thông báo không cần thiết vào thiết bị này. Nếu sử dụng `/dev/null` làm thiết bị nhập vào, thì sẽ thu được một chuỗi các số 0, tức là trong trường hợp này tập tin có cỡ bằng 0.

Mỗi dạng thiết bị có thể có một vài tập tin thiết bị. Vì thế các tập tin thiết bị thường có hai số: lớn (major) và nhỏ (minor). Số lớn của thiết bị cho nhân biết là tập tin này thuộc về driver nào, còn số nhỏ cho biết cần phải làm việc với thiết bị cụ thể nào của dạng này. Đối với các tập tin thiết bị, câu lệnh `ls -l` cho biết số lớn và số nhỏ đã nói thay vì kích thước của tập tin.

4.4.2 Các ống có tên (pipes)

Còn có một dạng tập tin thiết bị nữa đó là các ống có tên, hay bộ đệm FIFO (**F**irst **I**n – **F**irst **O**ut). Tập tin dạng này chủ yếu dùng để tổ chức trao đổi dữ liệu giữa

các chương trình khác nhau (pipe dịch từ tiếng Anh sang là ống, đường ống).

Ống là phương tiện hết sức thuận tiện và sử dụng rộng rãi để trao đổi thông tin giữa các tiến trình. Một tiến trình có thể đọc tất cả những gì mà một tiến trình khác đặt vào ống. Nếu có hai tiến trình được sinh ra từ cùng một tiến trình mẹ trao đổi thông tin (thường xảy ra), thì ống có thể không có tên. Trong trường hợp ngược lại cần tạo ra một ống có tên, ví dụ bằng chương trình `mkfifo`. Khi này bản thân tập tin ống chỉ tham gia vào sự khởi đầu trao đổi dữ liệu.

4.4.3 Các socket

Socket đó là kết nối giữa các tiến trình, cho phép chúng giao tiếp mà không chịu ảnh hưởng của các tiến trình khác. Nói chung socket và sự trao đổi qua socket đóng vai trò hết sức quan trọng trên tất cả các hệ thống UNIX, trong đó có Linux: socket là khái niệm then chốt của TCP/IP và như vậy là dựa trên socket đã xây dựng toàn bộ Internet. Tuy nhiên từ phía hệ thống tập tin socket thực tế không khác các ống có tên: đó chỉ là các điểm cho phép nối các chương trình với nhau. Sau khi đã tạo ra kết nối, thì trao đổi được thực hiện mà không cần đến socket: dữ liệu do nhân chuyển trực tiếp từ chương trình này đến chương trình khác.

Mặc dù những tiến trình khác có thể thấy các tập tin socket, nhưng những tiến trình không tham gia vào kết nối hiện thời không thể thực hiện bất kỳ thao tác đọc hay ghi nào lên tập tin socket. Trong số những hệ thống sử dụng socket cần kể đến X Window, hệ thống in và hệ thống `syslog`.

4.4.4 Liên kết mềm

Trong phần về tên tập tin ở chương trước chúng ta đã nói rằng tập tin trong Linux có thể có vài tên hay liên kết cứng. Liên kết cứng chỉ là một tên khác cho tập tin ban đầu. Nó được ghi trong mô tả inode của tập tin đó. Sau khi tạo liên kết cứng không thể phân biệt đâu là tên tập tin còn đâu là liên kết. Nếu bạn đọc xóa một trong số những tập tin này (nói đúng hơn là một trong số những tên này), thì tập tin vẫn còn được lưu trên đĩa cho đến khi vẫn còn ít nhất một tên.

Rất khó phân biệt tên đầu tiên của tập tin và những liên kết cứng được tạo ra sau đó. Vì vậy chỉ dùng liên kết cứng ở những nơi không cần biết sự khác nhau. Một trong những ứng dụng của liên kết cứng đó là ngăn chặn khả năng xóa tập tin một cách vô tình. Điểm đặc biệt của liên kết cứng là nó chỉ thẳng đến chỉ số inode, và do đó liên kết cứng chỉ có thể dùng cho tập tin của cùng một hệ thống tập tin, tức là trên cùng một phân vùng (đĩa lưu).

Nhưng trên Linux còn có một dạng liên kết khác gọi là (liên kết tượng trưng[9]). Những liên kết này cũng có thể coi là tên phụ cho tập tin, nhưng chúng là những tập tin khác – những tập tin liên kết mềm. Khác với liên kết cứng, liên kết mềm có thể chỉ đến những tập tin nằm trong hệ thống tập tin khác, ví dụ trên những đĩa lưu động, hoặc thậm chí trên một máy tính khác. Nếu tập tin ban đầu bị xóa, thì liên kết mềm tuy không bị xóa nhưng trở thành vô giá trị. Hãy sử dụng liên kết mềm trong những trường hợp bạn muốn tránh sự lầm lẫn mà liên kết cứng có thể gây ra, hoặc khi tập tin nằm trên một hệ thống tập tin khác.

[9] thuật ngữ này vẫn chưa được thống nhất, do đó đưa ra cả hai trường hợp có thể gặp.

Việc tạo bất kỳ một liên kết nào cũng giống như sao chép tập tin, nhưng khác ở chỗ là tên ban đầu của tập tin cũng như liên kết cùng chỉ đến một tập tin thực sự trên đĩa. Vì thế nếu bạn đọc thay đổi tập tin qua một tên nào đó của nó, thì sẽ thấy những thay đổi này khi xem tập tin qua tên khác. Để tạo liên kết mềm cần sử dụng câu lệnh `ln` đã nói đến và thêm vào tùy chọn `-s`:

```
[user]$ ln -s tên_tập_tin tên_liên_kết
```

Ví dụ:

```
[user]$ ln -s projects/l4u/l4u-0.9.2.pdf ~/l4u.pdf
```

Sau khi thực hiện câu lệnh này trong thư mục cá nhân của tôi xuất hiện tập tin `l4u.pdf`. Và bây giờ nếu xem danh sách tập tin trong thư mục bằng câu lệnh `ls -l`, thì trong danh sách sẽ có một dòng như sau:

```
lrwxrwxrwx 1 teppi82 users 20 2006-09-10 06:39 l4u.pdf -> projects/l4u/l4u-0.9.2.pdf
```

Hãy chú ý đến ký tự đầu tiên của dòng này, nó cho chúng ta biết tập tin là một liên kết mềm. Tất nhiên điều này có thể thấy rõ trong phần cuối (phần tên tập tin), ở đó sau tên tập tin của liên kết là một mũi tên → chỉ đến tập tin ban đầu.

Nếu bạn tạo trong thư mục `mot` một liên kết mềm chỉ đến một thư mục khác, thì có thể di chuyển thư mục `mot` đi đâu tùy thích, liên kết mềm khi đó vẫn làm việc đúng. Cũng như vậy đối với chính bản thân liên kết mềm. Nhưng khi tạo liên kết mềm, hãy hạn chế sử dụng "`..`" (liên kết đến thư mục mẹ) trong tên tập tin chỉ đến, bởi vì vị trí của liên kết mềm có thể thay đổi, mà "`..`" luôn luôn là thư mục mẹ của thư mục hiện thời.

4.5 Quyền truy cập đến tập tin và thư mục

Bởi vì Linux là hệ điều hành nhiều người dùng, nên yêu cầu quy định truy cập đến các tập tin và thư mục là một trong những yêu cầu thiết yếu nhất mà hệ điều hành phải giải quyết. Cơ chế quy định truy cập được phát triển cho hệ thống UNIX vào những năm 70 của thế kỷ trước rất đơn giản nhưng có hiệu quả đến nỗi đã được sử dụng hơn 30 năm, và hiện thời vẫn còn được sử dụng để giải quyết bài toán này.

Cơ sở của cơ chế quy định quyền truy cập đó là tên người dùng và tên nhóm của người dùng. Như bạn đã biết trong Linux mỗi người dùng có một tên riêng không lặp lại dùng để đăng nhập vào hệ thống. Ngoài ra, trên hệ thống còn có các nhóm người dùng, và Linux cho phép một người dùng có thể nằm trong một hoặc nhiều nhóm. Tạo và xóa các nhóm là công việc của người dùng cao cấp root, và root có thể thay đổi thành phần của một nhóm nào đó. Thành viên của các nhóm khác nhau có thể có quyền truy cập khác nhau đến tập tin, ví dụ nhóm các nhà quản trị có quyền nhiều hơn so với nhóm các nhà lập trình.

Trong mô tả inode của mỗi tập tin có ghi tên của chủ và nhóm sở hữu tập tin. Ngày từ đầu khi tạo tập tin chủ của nó là người dùng đã tạo ra nó. Nói chính xác hơn là người dùng mà tiến trình tạo tập tin đã chạy dưới tên họ. Cùng lúc với

chủ sở hữu, tên của nhóm sở hữu cũng được ghi vào theo thông tin tên nhóm của tiến trình tạo tập tin. Có thể thay đổi chủ và nhóm sở hữu trong quá trình làm việc sau này bằng hai câu lệnh `chown` và `chgrp` (chúng ta sẽ đề cập kỹ hơn về hai lệnh này ngay sau đây).

Bây giờ hãy thực hiện một lần nữa câu lệnh `ls -l`, nhưng có thêm một tham số nữa đó là tên của một tập tin cụ thể nào đó. Ví dụ tập tin chương trình của hệ vỏ bash `/bin/bash`. Nhân tiện, hãy chú ý khả năng này của câu lệnh `ls -l` – hiển thị thông tin về một tập tin cụ thể nào đó chứ không phải tất cả các tập tin trong thư mục một lúc.

```
[user]$ ls -l /bin/bash
-rwxr-xr-x 1 root root 501804 2006-04-23 05:46 /bin/bash
```

Như bạn đọc thấy, trong trường hợp này chủ sở hữu là người dùng `root`, nhóm sở hữu – `root`. Nhưng bây giờ trên dòng này chúng ta sẽ quan tâm hơn đến vùng đầu tiên, vùng xác định dạng tập tin và quyền truy cập đến nó. Vùng này trong ví dụ trên là chuỗi các ký tự tạm thời chưa nói lên điều gì "`-rwxr-xr-x`". Những ký tự này có thể tạm chia thành bốn nhóm. Nhóm thứ nhất chỉ gồm một ký tự xác định dạng tập tin (một trong bốn dạng đã nêu ở phần trên). Nó có thể là một trong số những ký tự sau:

- **–** (gạch ngang) – tập tin thông thường
- **d** – thư mục
- **b** – tập tin thiết bị khối
- **c** – tập tin thiết bị ký tự
- **s** – socket
- **p** – ống có tên (pipe)
- **l** – liên kết mềm (symbolic link).

Sau ký tự xác định dạng tập tin là ba nhóm, mỗi nhóm gồm ba ký tự xác định quyền truy cập tương ứng cho chủ sở hữu, nhóm sở hữu tập tin và cho những người dùng khác. Trong ví dụ của chúng ta quyền truy cập của chủ sở hữu là `rwx`, có nghĩa là chủ sở hữu root có quyền đọc (`r`), ghi vào tập tin (`w`) và chạy tập tin này (`x`). Thay bất kỳ ký tự nào trong số những ký tự này bằng dấu gạch ngang có nghĩa là người dùng bị tước mất quyền tương ứng. Cũng trong ví dụ ở trên chúng ta thấy, tất cả những người dùng khác (kể cả những người dùng của nhóm root) bị tước mất quyền ghi vào tập tin này, có nghĩa là họ không thể sửa tập tin và nói chung là không thể thay đổi tập tin bằng cách nào đó.

Quyền truy cập và thông tin về dạng tập tin trên các hệ thống UNIX được lưu trong mô tả inode ở dạng cấu trúc 2 byte (16 bit). Điều này là tất nhiên vì máy tính chỉ làm việc dựa trên các bit chứ không phải dựa trên các ký tự `r`, `w`, `x`. Bốn bit trong số 16 bit này được dùng cho bản ghi về dạng tập tin. Ba bit tiếp theo xác định các tính chất đặc biệt của tập tin thực thi (chúng ta sẽ nói đến một chút ở sau). Và cuối cùng 9 bit cuối cùng xác định quyền truy cập đến tập tin. 9

bit này chia thành ba nhóm, mỗi nhóm 3 bit. Ba bit đầu tiên xác định quyền của chủ sở hữu, ba bit tiếp theo – quyền của nhóm sở hữu, ba bit cuối cùng – quyền của những người dùng còn lại (tức là tất cả những người dùng, trừ chủ sở hữu và nhóm sở hữu tập tin). Khi này nếu bit tương ứng có giá trị bằng "1", thì có quyền đó, còn nếu bằng "0" thì quyền đó bị tước mất. Ở dạng chữ cái thì "1" được thay thế bằng các chữ cái tương ứng (r, w hoặc x), còn "0" thể hiện ở dạng dấu gạch ngang.

Quyền đọc `r` tập tin có nghĩa là người dùng có thể xem nội dung tập tin bằng các chương trình xem khác nhau, ví dụ `more`, hoặc bằng các trình soạn thảo văn bản. Nhưng khi soạn thảo bạn sẽ không thể lưu những thay đổi trong tập tin lên đĩa, nếu không có quyền ghi `w` vào tập tin này. Quyền thực thi (tôi thích dùng thuật ngữ *quyền gọi*) có nghĩa là bạn đọc có thể nạp tập tin vào bộ nhớ và thử chạy mã này giống như trường hợp chương trình. Tất nhiên nếu trên thực tế tập tin không phải là chương trình (hoặc các script shell, perl, ...) thì không thể gọi tập tin, nhưng ngược lại nếu tập tin là chương trình mà không có quyền gọi thì cũng không thể chạy chương trình đó.

Như vậy là chúng ta đã biết được trên Linux những tập tin nào là có thể thực thi. Bạn thấy không, phần mở rộng của tập tin ở đây không có liên quan gì, tất cả đều do tính chất "thực thi" đặt ra, và khác với các HĐH của Microsoft không phải ai cũng có quyền gọi tập tin.

Nếu vẫn thực hiện câu lệnh `ls -l` nhưng tham số không phải là tên tập tin mà là tên thư mục thì chúng ta sẽ thấy thư mục cũng có quyền truy cập và cũng vẫn những chữ cái `r, w, x` nói trên được dùng để xác định quyền truy cập đến thư mục. Ví dụ, nếu thực hiện câu lệnh:

```
[user]$ ls -l /usr
```

thì sẽ thấy dòng tương ứng với thư mục `share` như sau:

```
drwxr-xr-x 128 root root  4096 2006-09-07 02:20 share
```

Tất nhiên là đối với thư mục thì ý nghĩa của các khái niệm "*quyền đọc*", "*quyền ghi*" và "*quyền gọi*" có thay đổi một chút. Quyền đọc đối với thư mục thì hết sức dễ hiểu, nếu chúng ta nhớ rằng thư mục cũng chỉ là tập tin lưu danh sách các tập tin khác trong thư mục đó. Cho nên nếu người dùng có quyền đọc thư mục, thì tức là có thể xem nội dung của thư mục (có thể nói khác là xem danh sách tập tin trong thư mục). Quyền ghi cũng dễ hiểu. Khi có quyền này, người dùng có thể tạo và xóa các tập tin trong thư mục, tức là thêm vào hoặc xóa khỏi thư mục dòng lưu thông tin về một tập tin nào đó và các liên kết tương ứng. Quyền gọi đối với thư mục có hơi khó hiểu một chút. Trong trường hợp này quyền gọi chỉ quyền chuyển vào thư mục này. Nếu bạn đọc là chủ sở hữu thư mục và muốn cho những người dùng khác quyền xem một tập tin nào đó trong thư mục của mình thì cần phải cho họ quyền truy cập (chuyển) vào thư mục này, tức là cho những người dùng khác "quyền gọi" (thực thi) thư mục. Hơn nữa còn cần phải cho người dùng "quyền gọi" đối với tất cả các thư mục nằm trước thư mục này trong cây thư mục. Chính vì vậy mà theo mặc định tất cả các thư mục có đặt quyền gọi cho chủ sở hữu cũng như nhóm và những người dùng khác. Và tất nhiên nếu muốn ngăn chặn truy cập vào thư mục thì chỉ cần bỏ đi quyền chuyển vào thư mục (`r`) của

tất cả người dùng (kể cả nhóm sở hữu). Đừng tước bỏ quyền này của chính bản thân mình, nếu không sẽ phải phục hồi lại nó trước khi có thể đọc các tập tin.

Sau khi đọc đoạn trên có thể thấy "quyền đọc" thư mục là thừa thãi vì không cho ra tính năng gì mới so với "quyền gọi". Tuy nhiên vẫn có sự khác nhau giữa hai quyền này. Nếu chỉ đưa ra quyền gọi, thì người dùng có thể vào thư mục, nhưng sẽ không thấy ở đó bất kỳ một tập tin nào khi chạy lệnh `ls` (có thể thấy rõ hơn nếu bạn sử dụng chương trình Midnight Commander). Nếu có quyền truy cập tới một thư mục con nào đó của thư mục này, thì bạn có thể chuyển sang thư mục con bằng lệnh `cd`, nhưng cần phải nhớ tên của thư mục con này, vì sẽ không thấy bất kỳ danh sách và tập tin thư mục nào (trường hợp này giống như khi chúng ta đi trong màn đêm không thấy đường, chỉ nhớ hướng đi).

Cơ chế kiểm tra quyền người dùng khi sử dụng tập tin như sau. Đầu tiên hệ thống kiểm tra xem tên người dùng có trùng với tên chủ sở hữu tập tin hay không. Nếu hai tên này trùng nhau (tức là chủ sở hữu đang dùng tập tin của mình), thì kiểm tra xem chủ sở hữu có các quyền truy cập tương ứng (đọc, ghi và gọi) không. Đừng ngạc nhiên khi chủ sở hữu lại không có tất cả mọi quyền, người dùng root có thể tước bỏ một số quyền của chủ sở hữu tập tin. Nếu có quyền truy cập đó, thì sẽ được cho phép thực hiện thao tác tương ứng. Nếu chủ sở hữu không có quyền nào đó, thì thậm chí hệ thống không kiểm tra quyền có thể có ở nhóm sở hữu và những người dùng khác mà đưa ra luôn thông báo lỗi không thể thựchieejn được hành động yêu cầu (dạng "Permission denied").

Nếu tên người dùng không trùng với tên chủ sở hữu thì hệ thống kiểm tra xem người dùng này có nằm trong nhóm sở hữu hay không. Nếu có thì khả năng truy cập đến tập tin được xác định bằng quyền truy cập của nhóm, và không chú ý đến các quyền của chủ sở hữu và những người dùng còn lại. Nếu người dùng không phải là chủ sở hữu và cũng không nằm trong nhóm sở hữu, thì quyền của họ được xác định bằng nhóm tính chất thứ ba (nhóm dành cho những người dùng còn lại). Như vậy nhóm tính chất thứ ba trong quyền truy cập là dành cho tất mọi người dùng, trừ chủ sở hữu và những người dùng nằm trong nhóm sở hữu.

Để thay đổi quyền truy cập tới tập tin người ta sử dụng lệnh `chmod` (**ch**ange **mod**e). Có hai cách sử dụng lệnh này. Khi dùng cách thứ nhất bạn phải chỉ ra rõ ràng thêm quyền nào cho ai hoặc tước quyền nào và của ai như sau:

```
[user]$ chmod wXp tên_tập_tin
```

Trong đó, ở chỗ ký tự `w` phải đặt một trong các ký tự sau:

- **u** – chủ sở hữu
- **g** – nhóm sở hữu g
- **o** – những người dùng còn lại
- **a** – tất cả bao gồm chủ sở hữu, nhóm và những người dùng còn lại.

Ở chỗ `X` là một trong các ký tự sau:

- **+** – thêm quyền
- **–** – tước bỏ quyền

- **=** – dùng quyền chỉ ra thay cho quyền đã có.

Ở chỗ `p` là một trong những ký tự sau:

- **r** – quyền đọc
- **w** – quyền ghi
- **x** – quyền gọi (quyền thực hiện.

Sau đây là một số ví dụ sử dụng câu lệnh `chmod`:

```
[user]$ chmod a+x tên_tập_tin
```

thêm quyền gọi tập tin `tên_tập_tin` cho mọi người dùng của hệ thống.

```
[user]$ chmod go-rw tên_tập_tin
```

tước bỏ quyền đọc và ghi của mọi người dùng trừ chủ sở hữu tập tin.

```
[user]$ chmod ugo+rwx tên_tập_tin
[user]$ chmod a+rwx tên_tập_tin
```

cho mọi người dùng quyền đọc, ghi và gọi (thực hiện).

```
[user]$ chmod u=rwx,go=x tên_tập_tin
```

cho chủ sở hữu có tất cả mọi quyền (đọc, ghi, gọi), những người dùng còn lại chỉ có quyền gọi (thực hiện).

Nếu không chỉ ra ai được thêm quyền truy cập, thì sẽ áp dụng cho tất cả mọi người dùng, tức là có thể dùng lệnh:

```
[user]$ chmod +x tên_tập_tin
```

để thay cho

```
[user]$ chmod a+x tên_tập_tin
```

Phương án sử dụng thứ hai của câu lệnh `chmod` có khó hiểu hơn một chút trong thời gian đầu sử dụng Linux, nhưng lại thường xuyên được các nhà quản trị cũng như người dùng có kinh nghiệm dùng. Nó dựa trên mã hóa quyền truy cập ở dạng số. Ký tự `r` được mã hóa bằng số 4, `w` – số 2, `x` – số 1. Để xác định quyền của người dùng cần cộng các số tương ứng lại với nhau. Sau khi thu được ba giá trị số (cho chủ sở hữu, nhóm sở hữu và những người dùng còn lại), chúng ta đưa ba số này vào dùng làm tham số cho lệnh `chmod`. Chúng ta cần đặt ba số này phía sau tên lệnh và phía trước tham số thứ hai (tên tập tin). Ví dụ, nếu cần cho chủ sở hữu mọi quyền (4+2+1=7), cho nhóm sở hữu quyền đọc và ghi (4+2=6) và những người dùng còn lại quyền gọi (1=1), thì dùng lệnh sau:

```
[user]$ chmod 761 tên_tập_tin
```

Nếu bạn biết về mã đôi của hệ cơ số tám, thì hiểu rằng những số đứng sau tên lệnh không phải gì khác mà chính là bản ghi ở hệ cơ số tám của 9 bit xác định quyền truy cập cho chủ sở hữu, nhóm sở hữu và những người dùng còn lại.

Chỉ có chủ sở hữu tập tin hoặc người dùng cao cấp mới có khả năng thay đổi quyền truy cập bằng câu lệnh `chmod`. Để có thể thay đổi quyền của nhóm sở hữu, thì chủ sở hữu (không phải root) phải là thành viên của nhóm đó.

Để kết thúc bài học về quyền truy cập đến tập tin cần nói thêm về những tính chất khác có thể gặp của tập tin mà cũng xác định bằng lệnh `chmod`. Đó là những tính chất cho các tập tin thực thi. Trong mô tả inode, phần cấu trúc 2 byte xác định tính chất tập tin, chúng chiếm các vị trí 5 – 7 ngay sau mã cho biết dạng tập tin. Tính chất đầu tiên đó là "*bit thay đổi ID*[10] *người dùng*". Ý nghĩa của bit này như sau.

Thông thường, khi người dùng gọi thực hiện một chương trình nào đó, thì chương trình này nhận được những quyền truy cập đến tập tin và thư mục của người dùng đã chạy nó. Nếu như có đặt "bit thay đổi ID người dùng", thì chương trình nhận được quyền truy cập đến tập tin và thư mục của chủ sở hữu tập tin chương trình. Như vậy bit này còn có thể gọi là "*bit thay đổi ID chủ sở hữu tiến trình*". Điều này cho phép giải quyết một số vấn đề khó thực hiện. Ví dụ điển hình nhất là câu lệnh thay đổi mật khẩu `passwd`. Tất cả mật khẩu được lưu trong tập tin `/etc/passwd` (hoặc một tập tin mã hóa nào khác, trong đa số trường hợp là `/etc/shadow`) mà chủ sở hữu là người dùng cao cấp root. Vì thế chương trình nếu do người dùng chạy sẽ không thể thực hiện lệnh ghi vào tập tin này. Có nghĩa là người dùng không thể thay đổi mật khẩu của mình. Nhưng tập tin `/usr/bin/passwd` có "bit thay đổi ID người dùng", và root là chủ sở hữu tập tin chương trình này. Do đó chương trình thay đổi mật khẩu `passwd` được chạy với quyền root và nhận được quyền ghi vào tập tin `/etc/passwd`. Tất nhiên là trong chương trình `passwd` đã có mã để người dùng chỉ được phép thay đổi một dòng trong tập tin này – dòng tài khoản của người dùng đó.

Người dùng cao cấp root có thể đặt "bit thay đổi ID người dùng" bằng lệnh:

```
[root]# chmod +s tên_tập_tin
```

Tương tự như vậy chúng ta có "*bit thay đổi ID nhóm*". Ý nghĩa của bit này cũng giống như trên nhưng chỉ thay thế "người dùng" bằng "nhóm".

Một tính chất nữa của tập tin thực thi đó là "*bit dính*" (chính xác hơn là "*bit lưu chương trình*") hay thuật ngữ tiếng Anh là "sticky bit". Chúng ta sẽ dùng thuật ngữ "bit lưu chương trình" vì đúng với ngữ cảnh này hơn. Bit này chỉ hệ thống biết sau khi dừng chương trình cần lưu lại nó trong bộ nhớ. Rất thuận tiện khi đặt bit này cho những chương trình thường gọi, vì trong trường hợp này sẽ tiết kiệm được thời gian nạp chương trình vào bộ nhớ mỗi lần chạy. Bit này chỉ cần thiết trên những máy cũ. Trên những máy "top model" (high end) hiện đại thì rất hiếm khi sử dụng.

Nếu sử dụng phương án xác định tính chất ở dạng số của lệnh `chmod`, thì giá trị của ba tính chất vừa nói phải nằm trước những số xác định quyền truy cập (tức là số đầu tiên trong dãy 4 số xác định tất cả các tính chất của tập tin). Ví dụ:

[10] Identificator, mỗi người dùng có một ID dạng số như vậy. ID của người dùng là duy nhất, không trùng lặp. Ví dụ ID của người dùng cao cấp root là 0.

```
[root]# chmod 4775 tên_tập_tin
```

Khi này cũng vẫn sử dụng phép cộng như đối với trường hợp quyền truy cập và các tính chất có giá trị như sau:

- **4** – "bit thay đổi ID người dùng"
- **2** – "bit thay đổi ID nhóm"
- **1** – "bit lưu chương trình" (sticky bit).

Nếu có (những) bit nào đó trong số ba bit này được đặt (nhận giá trị 1), thì sẽ có thay đổi của kết quả của lệnh `ls -l` trong phần quyền truy cập (phần đầu tiên). Nếu "bit thay đổi ID người dùng" bằng 1, thì ký tự `x` trong phần xác định quyền truy cập của chủ sở hữu sẽ được thay thế bằng ký tự `s`. Lúc này nếu chủ sở hữu có *quyền gọi* tập tin thì ký tự `x` được thay thế bằng chữ cái `s` nhỏ, còn ngược lại (ví dụ tập tin không phải là chương trình), thì thay thế `x` bằng chữ cái `S` lớn. Sự thay thế như vậy cũng xảy ra nếu có đặt "bit thay đổi ID nhóm", tất nhiên là sẽ thay thế ký tự `x` trong phần xác định quyền truy cập của nhóm sở hữu. Nếu "bit lưu chương trình" (sticky bit) bằng 1, thì thay thế ký tự `x` trong phần xác định quyền truy cập của những người dùng còn lại bằng ký tự `t`, nếu những người dùng còn lại có quyền thực hiện tập tin, bằng ký tự `T`, nếu ngược lại.

Như vậy, mặc dù trong kết quả của lệnh `ls -l` không có những vị trí riêng để hiển thị kết quả của ba bit ("bit thay đổi ID người dùng", "bit thay đổi ID nhóm" và "bit lưu chương trình"), chúng ta vẫn có thể thấy được những thông tin này. Một vài ví dụ:

```
[user]$ ls -l /usr/bin/passwd /usr/bin/write
-rwsr-xr-x 1 root shadow 72836 2006-05-02 12:50 /usr/bin/passwd
-rwxr-sr-x 1 root tty 8936 2006-05-02 10:50 /usr/bin/write
```

(Ở đây chúng ta thấy có thể liệt kê nhiều tập tin trên dòng lệnh `ls -l`, tức là dùng nhiều tập tin làm tham số cho lệnh `ls`.)

```
[user]$ touch vidu
[user]$ chmod 7766 vidu
[user]$ ls -l vidu
-rwsrwSrwT 1 teppi82 users 0 2006-09-11 12:46 vidu
```

4.6 Các câu lệnh cơ bản để làm việc với tập tin và thư mục

Trong những phần trước chúng ta đã đề cập đến một vài câu lệnh để làm việc với tập tin và thư mục, đó là `pwd, cd, ls, ln, chmod`. Trong phần này chúng ta sẽ xem xét một cách ngắn gọn một vài câu lệnh thường dùng nữa.

4.6.1 Câu lệnh `chown` và `chgrp`

Những câu lệnh này dùng để thay đổi chủ sở hữu và nhóm sở hữu tập tin. Chỉ có người dùng cao cấp root mới có quyền thay đổi chủ sở hữu, còn thay đổi nhóm sở hữu tập tin có thể là root hoặc người dùng chủ sở hữu. Để có quyền thay đổi nhóm, thì chủ sở hữu còn phải là thành viên của nhóm **sẽ** sở hữu tập tin này. Cú pháp của hai câu lệnh này tương tự nhau:

```
[root]# chown tên_người_dùng tên_tập_tin
[root]# chgrp tên_nhóm tên_tập_tin
```

4.6.2 Câu lệnh `mkdir`

Câu lệnh `mkdir` cho phép tạo thư mục con trong thư mục hiện thời. Tham số của câu lệnh này là tên của thư mục muốn tạo ra. Trong thư mục vừa tạo sẽ tự động tạo ra hai mục: `.` (liên kết đến chính bản thân thư mục này) và `..` (liên kết đến thư mục mẹ). Để tạo ra thư mục con, bạn đọc cần phải có quyền ghi vào thư mục hiện thời. Có thể tạo ra thư mục con trong một thư mục khác thư mục hiện thời, nhưng khi này cần phải chỉ ra đường dẫn tới đó. Ví dụ:

```
[user]$ mkdir ~/projects/l4u/images
```

(hãy nhớ lại ký hiệu ~ dùng để chỉ của người dùng).

Có thể dùng các tùy chọn sau của câu lệnh `mkdir`:

- -m mode – xác định quyền (chế độ) truy cập cho thư mục mới (ví dụ: -m 700)
- -p – tạo ra các thư mục trung gian chỉ ra trong đường dẫn (nếu chưa có chúng). Ví dụ:

```
teppi82@ThinhQuyen:~> mkdir mot/hai
mkdir: cannot create directory 'mot/hai': No such file or directory
teppi82@ThinhQuyen:~> mkdir -p mot/hai
teppi82@ThinhQuyen:~> ls -l mot
tổng 4
drwxr-xr-x 2 teppi82 users 4096 2006-09-11 13:36 hai
```

4.6.3 Câu lệnh `cat`

Câu lệnh `cat` thường dùng để tạo các tập tin, mặc dù có thể sử dụng lệnh `touch`. Lệnh `cat` cũng đưa ra màn hình (đầu ra) nội dung của (các) tập tin dùng làm tham số của nó. Nếu chuyển kết quả làm việc của lệnh `cat` vào một tập tin nào đó thì có thể tạo ra bản sao của tập tin như sau:

```
[user]$ cat tập_tin1 > tập_tin2
```

Chính bản thân câu lệnh `cat` lúc đầu được phát triển để dùng cho việc chuyển hướng kết quả làm việc. Vì nó được tạo ra cho sự móc nối (concatenate, nếu dùng thuật ngữ của Hoá học là "sự cộng" các tập tin), tức là sự kết hợp các tập tin khác nhau vào một:

```
[user]$ cat tập_tin1 tập_tin2 ... tập_tinN > tập_tin_mới
```

Và cũng chính khả năng chuyển hướng kết quả của câu lệnh này được dùng để tạo các tập tin mới. Khi này đầu vào của lệnh `cat` đó là dòng dữ liệu nhập từ bàn phím (đầu vào tiêu chuẩn), còn đầu ra sẽ là tập tin mới:

```
[user]$ cat > tập_tin_mới
```

Sau khi nhập vào những gì muốn nhập, hãy nhấn tổ hợp phím <Ctrl>+<D> hoặc <Ctrl>+<C>, và tất cả những gì bạn đã gõ sẽ được lưu lại trong tập tin `tập_tin_mới`. Tất nhiên là như vậy chủ yếu dùng `cat` để tạo các tập tin văn bản ngắn.

4.6.4 Câu lệnh cp

Mặc dù đôi khi có thể dùng câu lệnh `cat` để sao chép các tập tin, nhưng trong Linux có một câu lệnh chuyên dùng cho việc này – lệnh `cp`. Có thể áp dụng một trong hai dạng của lệnh này:

```
[user]$ cp [tùy_chọn] nguồn đích
[user]$ cp [tùy_chọn] thư_mục_nguồn thư_mục_mới
```

Trong trường hợp thứ nhất sao chép tập tin (hoặc thư mục) `nguồn` vào tập tin (hoặc thư mục) `đích`. Còn trong trường hợp thứ hai thì tập tin có trong `thư_mục_nguồn` sẽ được sao chép vào thư mục `thư_mục_mới`. Để sao chép thì cần có quyền đọc tập tin muốn sao chép và quyền ghi vào thư mục sẽ sao chép đến (“*thư mục đích*”).

Nếu sử dụng một tập tin đã có vào chỗ của tập tin `đích` thì nội dung của nó sẽ bị xóa mất, do vậy khi sao chép cần phải cẩn thận. Vả lại có thể sử dụng câu lệnh `cp` với tùy chọn `-i`. Khi đó trước khi ghi chèn lên tập tin đã có hệ thống sẽ hỏi lại người dùng. Rất nên dùng tùy chọn này.

Câu lệnh `cp` còn có một vài tùy chọn có ích khác liệt kê trong bảng 4.3.

Bảng 4.3: Những tùy chọn chính của lệnh `cp`

Tùy chọn	**Ý nghĩa**
–p	Giữ lại thời gian sửa đổi tập tin và cố giữ lại những quyền truy cập có thể giữ lại. Nếu không đưa ra tùy chọn này thì quyền truy cập của tập tin sẽ được thiết lập theo quyền của người dùng đã chạy lệnh.
–R hoặc –r	Nếu `nguồn` là thư mục thì sao chép thư mục đó cùng với tất cả những gì (tập tin, thư mục con) nằm trong nó, tức là giữ lại được cấu trúc của thư mục bạn đầu (**r**ecursive).
–d	Nếu đưa ra tùy chọn này thì các liên kết mềm sẽ vẫn là các liên kết, nếu không thì sẽ sao chép tập tin (nội dung) mà liên kết này chỉ đến.
–f	Ghi chèn tập tin khi sao chép mà không hỏi lại hay cảnh báo.

4.6.5 Câu lệnh `mv`

Nếu bạn không cần sao chép, mà cần di chuyển tập tin từ một thư mục này vào một thư mục khác, thì có thể sử dụng câu lệnh `mv`. Cú pháp của lệnh này tương tự như cú pháp của `cp`. Hơn nữa, lệnh này đầu tiên sao chép tập tin (hay thư mục), và sau đó mới xóa tập tin (thư mục) ban đầu. Các tùy chọn của nó cũng giống như của `cp`.

Câu lệnh `mv` không chỉ dùng để di chuyển, mà còn dùng để thay đổi tên tập tin và thư mục, tức là di chuyển chúng trong phạm vi của một thư mục. Chỉ cần đặt vào chỗ hai tham số tên cũ và tên mới của tập tin như thế này:

```
[user]$ mv tên_cũ tên_mới
```

Nhưng hãy chú ý là câu lệnh `mv` không cho phép đổi tên một vài tập tin cùng lúc bằng cách sử dụng các mẫu tên. Do đó câu lệnh:

```
[user]$ mv *.doc *.odt
```

sẽ không làm việc. Khi sử dụng lệnh `mv` cũng giống như khi sử dụng `cp`, đừng quyền thêm vào tùy chọn `-i` để hiện ra cảnh báo khi có tập tin sẽ bị ghi chèn.

4.6.6 Câu lệnh `rm` và `rmdir`

Để xóa những tập tin và thư mục không cần thiết trên Linux có các câu lệnh `rm` (xóa tập tin) và `rmdir` (xóa thư mục rỗng). Để sử dụng những câu lệnh này, bạn đọc cần có quyền ghi vào thư mục lưu những tập tin hoặc thư mục muốn xóa. Khi này quyền thay đổi chính bản thân các tập tin và thư mục là không cần thiết. Nếu muốn câu hỏi xác nhận sự cho phép của người dùng xuất hiện trước khi xóa tập tin, thì hãy dùng tùy chọn `-i` (rất dễ nhớ, tùy chọn này có ở những câu lệnh `cp`, `mv` đã kể trên).

Nếu dùng câu lệnh `rm` (không có tùy chọn) để xóa thư mục thì sẽ xuất hiện thông báo dạng “cannot remove ‘l4u’: Is a directory” (không thể xóa bỏ, đây là thư mục). Để xóa thư mục thì cần xóa tất cả những tập tin có trong nó, sau đó xóa bản thận thư mục bằng lệnh `rmdir`. Tuy nhiên có thể xóa thư mục không rỗng cùng với tất cả những tập tin và thư mục có trong nó, nếu sử dụng câu lệnh `rm` với tùy chọn `-r`.

Nếu chạy lệnh `rm *`, thì sẽ xóa tất cả những tập tin có trong thư mục hiện thời. Các thư mục con không bị động tới. Để xóa cả tập tin và thư mục con của thư mục hiện thời cần dùng tùy chọn `-r` kể trên. Tuy nhiên cần luôn luôn nhớ rằng, trên Linux không có câu lệnh phục hồi tập tin sau khi xóa, thậm chí cả khi vừa xóa xong[11]. Theo tôi nghĩ bất kỳ người dùng Linux nào cũng có thể chia sẻ với bạn cảm giác bị mất tập tin “ngay trước mắt”. Vì thế hãy khi hai lần trước khi xóa gì đó và đừng quên tùy chọn `-i`.

[11] Có một số cách phụ hồi tập tin đã xóa trên hệ thống tập tin cũ ext3fs, nhưng chúng ta là những người dùng mới, do đó không xem xét chúng. Tất nhiên bạn có thể tham khảo tài liệu HOWTO có trên http://www.tldp.org này trong trường hợp khẩn cấp.

4.6.7 Câu lệnh `more` và `less`

Câu lệnh `cat` cho phép đưa ra màn hình (đầu ra tiêu chuẩn) nội dung của bất kỳ tập tin này, tuy nhiên rất ít khi lệnh `cat` được sử dụng cho mục đích này, và chỉ dùng để hiển thị những tập tin có dung lượng rất nhỏ. Nguyên nhân là nội dung của tập tin lớn sẽ ngay lập tức chạy qua màn hình và người dùng chỉ thấy những dòng cuối cùng của tập tin. Vì thế `cat` dùng chủ yếu theo chức năng chính của nó, tức là dùng để "cộng" các tập tin, còn để xem nội dung của các tập tin văn bản chúng ta dùng các lệnh `more` và `less` hoặc các trình soạn thảo khác.

Câu lệnh (bộ lọc) `more` đưa nội dung của tập tin ra màn hình theo từng trang có kích thước bằng kích thước màn hình (nói chính xác thì là gần bằng, vì có một dòng cuối cùng dành cho để hiển thị trạng thái (status)). Để xem trang tiếp theo cần nhấn vào phím trắng <Space> (phím dài nhất trên bàn phím hiện nay). Nhấn phím <Enter> để đọc một dòng tiếp theo. Ngoài <Space> và <Enter> còn có một vài phím điều khiển khác, ví dụ phím <B> để quay lại màn hình trước, nhưng chúng ta sẽ không liệt kê đầy đủ những phím này ở đây, và cũng không đưa ra danh sách các tùy chọn của lệnh `more`. Bây giờ bạn đọc chỉ cần nhớ phím <Q> dùng để thoát ra khỏi chế độ xem của `more`, nếu không thì bạn sẽ phải nhấn phím <Space> cho đến khi hết tập tin (chẳng may nếu nó quá dài thì bạn sẽ mất rất nhiều thời gian). Tất cả các tuỳ chọn của lệnh `more` bạn có thể đọc trong trang hướng dẫn man (more(1)) hoặc info của nó.

Tiện ích `less` là một trong những chương trình được dự án GNU phát triển. `less` có tất cả các chức năng và lệnh điều khiển của `more`, và có thêm một vài sự mở rộng khác. Ví dụ, cho phép sử dụng các phím điều khiển con trỏ (<↑>, <↓>, <→>, <←>, <Home>, <End>, <PgUp>, <PgDown>) để di chuyển trong văn bản. Hãy nhớ lại, chúng ta đã nói về điều này khi nói về hệ thống trợ giúp man.

Các lệnh `more` và `less` cho phép tìm kiếm từ khóa có trong tập tin đang xem, trong đó lệnh `less` cho phép tìm kiếm theo hai hướng: từ trên xuống dưới và ngược lại. Để tìm kiếm từ khóa "string" (một cụm ký tự nào đó) thì đầu tiên cần nhấn "/" để chuyển vào chế độ tìm kiếm, sau đó nhập vào "string" vào dòng "/" ở cuối màn hình. Nếu tìm thấy từ khóa trong tập tin, thì sẽ hiển thị đoạn văn bản tương ứng sao cho dòng tìm thấy nằm ở trên cùng. Nếu muốn tiếp tục tìm kiếm hãy nhấn phím <N>, trong `less` có thể dùng tổ hợp phím <Shift>+<N> để tìm kiếm theo hướng ngược lại.

4.6.8 Câu lệnh tìm kiếm `find` và mẫu tên tập tin

Còn có một câu lệnh thường dùng để làm việc với tập tin trong Linux đó là câu lệnh tìm kiếm tập tin `find`. Câu lệnh `find` có thể tìm kiếm tập tin theo tên, theo kích thước, thời gian tạo hoặc thời gian sửa đổi tập tin và theo các tiêu chí khác. Cú pháp chung của câu lệnh `find` có dạng sau:

```
find [danh_sách_thư_mục] tiêu_chí_tìm_kiếm
```

Trong đó tham số "danh sách thư mục" xác định nơi tìm kiếm tập tin mong muốn. Đơn giản nhất là dùng thư mục gốc "/" làm nơi khởi đầu tìm kiếm. Tuy nhiên trong trường hợp đó tìm kiếm có thể kéo dài rất lâu, vì sẽ "lục soát" tất các thư mục kể cả những hệ thống tập tin gắn vào, trong đó có thể có các thư mục mạng

(và chuyện gì xảy ra nếu tốc độ đường truyền thấp). Có thể làm số khối lượng công việc, nếu dùng một danh sách những thư mục, mà tập tin có thể nằm trong, để thay thế cho thư mục gốc. Ví dụ:

```
[user]$ find /usr/bin /sbin /bin /usr/local/bin -name cp
```

Phần đầu của `tiêu_chí_tìm_kiếm` xác định xem chương trình `find` phải tìm cái gì. Phần đầu này là tham số bắt đầu bằng "–", "(", ")", "," hoặc "!". Tất cả các tham số đứng trước `tiêu_chí_tìm_kiếm` được coi là tên thư mục cần "lục soát". Nếu không chỉ ra một thư mục nào, thì tìm kiếm sẽ bắt đầu từ thư mục hiện thời và đi sâu vào trong các thư mục con.

Người dùng thường thực hiện tìm kiếm theo tên tập tin như ở ví dụ trên, ở đây `tiêu_chí_tìm_kiếm` có dạng "–name tên_tập_tin". Ở chỗ tùy chọn `-name` có thể sử dụng tùy chọn `-path`, khi đó câu lệnh sẽ tìm kiếm sự tương ứng của tên tập tin đầy đủ bao gồm cả đường dẫn chỉ ra. Ví dụ, câu lệnh:

```
[user]$ find . -path './l*es'
```

sẽ tìm thấy trong thư mục hiện thời thư mục con `l4u/images`. Trong ví dụ này, ở chỗ tên của tập tin hoặc thư mục chúng ta sử dụng một "mẫu tên". Và bởi vì mẫu tên tập tin không chỉ sử dụng cho câu lệnh `find` mà còn sử dụng cùng với nhiều câu lệnh khác (bao gồm cả những câu lệnh đã nói đến: `chmod`, `chown`, `chgrp`, `cp`, `rm`, `cat`, `mv`), nên chúng ta cần chú ý và dành thời gian để nghiên cứu các quy định sử dụng và viết "mẫu tên".

Trong đa số trường hợp mẫu tên tập tin được tạo ra nhờ các ký tự đặc biệt "*" và "?". Ký tự "*" sử dụng để thay thế cho bất kỳ dòng ký tự nào. Trong Linux:

- "*" tương ứng với tất cả các tập tin trừ những tập tin ẩn.
- ".*" tương ứng với tất cả những tập tin ẩn (trong đó có thư mục hiện thời "." và thư mục mẹ "..").
- "*.*" chỉ tương ứng với những tập tin và thư mục có dấu chấm (.) ở giữa tên hoặc ở cuối cùng.
- "a*p" tương ứng với `anhchup` và `anhchep`.
- "*a*" tương ứng với `May` và `march`.

Ký tự "?" chỉ thay thế một ký tự bất kỳ, vì thế `taptin?.txt` sẽ tương ứng với các tên sau (`taptin1.txt, taptin2.txt, taptin9.txt`.

Ngoài "*" và "?" trong Linux còn sử dụng các dấu ngoặc vuông ([]) để tạo "mẫu tên". Trong hai dấu ngoặc này đặt danh sách các ký tự (có thể ở dạng khoảng[12]) có thể gặp. Ví dụ `[xyz]*` tương ứng với tất cả những tên tập tin bắt đầu bằng a, b, c. Còn `*[G-K4-7]` tương ứng với những tập tin có tên kết thúc bằng G, H, I, J, K, 4, 5, 6, 7. Hãy chú ý là không có khoảng trắng trong cả hai ví dụ kể trên.

Tất nhiên ở đây chỉ đưa ra những thông tin thật ngắn gọn về "mẫu tên" tập tin và các ký tự thay thế. Bạn đọc có thể tìm thấy thông tin chi tiết hơn về "mẫu

[12] interval

Bảng 4.4: Tiêu chí tìm kiếm của câu lệnh find.

Tùy chọn	Giá trị
–name mẫu_tên	Tìm tập tin có tên tương ứng với `mẫu_tên`.
–group tên	Tìm tập tin thuộc về nhóm chỉ ra.
–size số[c]	Tìm tập tin có cỡ bằng `số` khối 512 byte [13]. Nếu sau `số` có ký tự `c` thì có nghĩa là kích thước được tính theo byte (ký tự, **c**harater), chứ không phải theo khối.
–mtime số_ngày	Tìm tập tin được thay đổi lần cuối cùng trước `số_ngày` chỉ ra.
–newer mẫu	Tìm tập tin được thay đổi sau khi thay đổi tập tin có trong `mẫu`.
–type dạng_tập_tin	Tìm tập tin dạng chỉ ra. Dạng tập tin được xác định bằng một trong các ký tự sau: b (thiết bị khối), c (thiết bị ký tự), d (thư mục), f (tập tin thường), p (ống có tên pipe), hoặc l (liên kết mềm).

tên" tập tin trong tài liệu IBM LPI tutorial bản dịch tiếng Việt do nhóm cộng tác của vnoss.org (trong đó có tôi) thực hiện có trên `http://sourceforge.net/projects/vnoss`.

Còn bây giờ sau khi đã làm quen với "mẫu tên" tập tin, chúng ta quay trở lại với câu lệnh `find` và nói chi tiết hơn về những khả năng có thể của tiêu chí tìm kiếm. Một vài ví dụ đơn giản của tiêu chí tìm kiếm có trong bảng 4.4.

Những tiêu chí tìm kiếm đơn giản khác bạn có thể tìm thấy trong trang man của câu lệnh `find` hoặc trong tài liệu LPI tutorial nói trên. Cần nói rằng từ những *tiêu chí đơn giản* có thể tạo ra những *tiêu chí phức tạp* hơn nhờ các phép lôgíc `and, or` hoặc phép phủ định (ký hiệu là dấu chấm than "!"). Ví dụ, nếu bạn muốn tìm tất cả những tập tin có "phần mở rộng"[14] là `.odt` và `.tex`, thì có thể dùng tiêu chí tìm kiếm như sau: `(-name *.tex -or -name *.odt)`. Có thể sử dụng kết hợp như vậy một số lượng bất kỳ các tiêu chí kể cả tiêu chí phức tạp. Nếu không chỉ ra phép lôgíc cụ thể, thì coi như dùng `and`, tức là có thể dùng một trong hai cách ghi: `(-name *.tex -and -name *.odt)` hoặc `(-name *.tex -name *.odt)`. Nếu chỉ dùng một phép lôgíc `and` hoặc `!`, thì có thể bỏ đi dấu ngoặc, còn phép lôgíc `or` và các tiêu chí phức tạp hơn thì cần dấu ngoặc. Trước dấu ngoặc cần đặt một dấu gạch chéo ngược (\), còn sau dấu ngoặc cần đặt một khoảng trắng. Ví dụ, nếu bạn đọc muốn tìm thư mục theo tên của nó thì có thể dùng lệnh:

```
[user]$ find /usr/share -name man1 -type d
```

hoặc dùng tiêu chí phức tạp:

```
[user]$ find /usr/share \( -name man1 -and -type d \)
```

Trong ví dụ sau chúng ta tìm tập tin theo tiêu chí như sau: hoặc tên tập tin có "phần mở rộng" *.tex, hoặc kích thước tập tin nhỏ hơn 200KB.

```
[user]$ find ~/projects \( \( -name *.tex \) -or \( -size -200 \) \)
```

[14] để trong dấu ngoặc vì chúng ta biết rằng trong Linux không có khái niệm phần mở rộng tập tin.

Trong ví dụ cuối cùng này hãy chú ý rằng trước giá trị kích thước tập tin có dấu "–". Dấu này có thể sử dụng với bất kỳ tham số có giá trị số nào trong tiêu chí tìm kiếm của câu lệnh `find`. Có nghĩa rằng cần tìm tập tin có giá trị của tham số nhỏ hơn số đưa ra. Tương tự dấu "+" có nghĩa là cần tìm tập tin có giá trị của tham số lớn hơn số đưa ra. Nếu không có dấu "+" và dấu "–" thì tìm tập tin có giá trị của tham số bằng số đưa ra.

Để kết thúc phần về câu lệnh `find` này, cần nói thêm rằng sau tiêu chí tìm kiếm có thể đưa ra ngay thao tác xử lý tất cả những tập tin tìm thấy. Ví dụ đơn giản nhất là thao tác `-print`.

```
[user]$ find ~/projects -name *.tex -print
```

dùng để đưa ra màn hình danh sách tên của tất cả những tập tin tìm thấy cùng với đường dẫn đầy đủ đến tập tin. Thao tác này được dùng theo mặc định, tức là luôn luôn được dùng khi không chỉ ra thao tác nào như trong các ví dụ trước đây.

Một ví dụ khác là thao tác `-exec cmd {}`. Trong đó `cmd` là một câu lệnh bất kỳ nào đó của hệ vỏ shell. Trong trường hợp này sẽ thực hiện câu lệnh `cmd` đối với tất cả những tập tin tìm thấy. Sau `cmd {}` là dấu chấm phẩy (;) có dấu gạch chéo ngược "\" ở trước (chúng ta sẽ hiểu rõ hơn tác dụng của dấu gạch chéo ngược trong chương ngay sau).

Ví dụ, nếu bạn muốn xóa tất cả những tập tin trong thư mục hiện thời mà người dùng không "động" đến trong vòng 365 ngày hoặc lâu hơn, thì hãy dùng câu lệnh sau:

```
[user]$ find . -type f -atime +365 -exec rm {} \;
```

Ở chỗ `-exec` có thể dùng `-ok`, khi đó trước khi thực hiện câu lệnh `cmd` cho mỗi tập tin tìm thấy hệ thống sẽ hỏi lại xem bạn có muốn thực hiện thật không.

Nói chung câu lệnh `cmd` là một câu lệnh rất mạnh, có ích và là một công cụ tìm kiếm tốt trong hệ thống tập tin. Đương nhiên là chưa phải tất cả những khả năng của lệnh này được liệt kê ra ở đây. Hãy tìm hiểu trong trang trợ giúp `man` hoặc một cuốn sách dày cộp nào đó về Linux. Và hãy cẩn thận khi sử dụng những khả năng của câu lệnh này như gọi những câu lệnh khác để thực hiện trên tất cả những tập tin tìm thấy. Hãy nhớ rằng sự thay đổi thường là một chiều.

4.6.9 Câu lệnh `split`

Đôi khi chúng ta cần chia một tập tin lớn thành từng phần nhỏ. Lấy ví dụ bạn có một tập tin phim dạng mpg lớn, khoảng 1,2GB. Và bạn muốn sao chép tập tin này qua một máy khác nhưng lại không có một kết nối mạng. Và cũng không có ổ ghi DVD mà chỉ có ổ ghi CD. Nhưng tập tin này không thể nằm gọn trên một CD (dung lượng khoảng 700MB). Do đó có thể chia tập tin này thành hai phần mỗi phần nhỏ hơn 700MB sau đó sẽ gộp chúng lại. Để làm được việc này chúng ta có thể sử dụng lệnh `split`.

Câu lệnh `split` cho phép sao chép tập tin bằng cách chia chúng ra thành từng phần nhỏ theo kích thước đã định. Tham số của lệnh này là tên của tập tin ban đầu và phần đầu (prefix) tên của các tập tin sẽ tạo ra. Các tập tin thu được sẽ có tên gồm phần đầu (prefix) này và hai chữ thêm vào để chúng không trùng

nhau: ‘aa’, ‘ab’, ‘ac’, ‘ad’, v.v... (không có khoảng trắng và các dấu chấm giữa phần đầu và những chữ cái này). Nếu không đưa ra phần đầu, thì theo mặc định sử dụng ‘x’, tức là sẽ thu được các tập tin ‘xaa’, ‘xab’, ‘xac’, ‘xad’ v.v...

Ngoài các tham số có thể thêm vào tùy chọn `-b` để xác định kích thước của các tập tin tạo ra (tính theo byte). Sau `-b` là một số và sau đó là một chữ cái: `k` (kích thước tính theo KB) hoặc `m` (tính theo MB). Nếu không đưa ra tùy chọn này thì theo mặc định kích thước của tập tin thu được bằng 1MB. Để giải quyết bài toán đã đưa lúc đầu thì cần chạy lệnh:

```
[user]$ split -b 650m phim.mpg phim
```

Sau đó dùng chương trình ghi đĩa sao chép hai tập tin thu được (phimaa, phimab) lên hai đĩa CD-R(W), rồi đưa chúng (có nghĩa là dùng lệnh cp) lên máy thứ hai vào một thư mục nào đó. Cuối cùng phục hồi tập tin ban đầu bằng lệnh:

```
[user]$ cat phim* > phim.mpg
```

Sau đó có thể xóa đi các tập tin phimaa, phimab.

4.6.10 So sánh các tập tin và lệnh `patch`

Có thể bạn không để ý nhưng khi làm việc với máy tính công việc so sánh nội dung của hai tập tin khác nhau gặp một cách thường xuyên. Là vì có thể sao chép tập tin một cách dễ dàng, rồi sau đó quên luôn là tập tin nào mới hơn hoặc tốt hơn. Vì thế những công cụ dùng để so sánh tập tin là cần thiết và tất nhiên là có trong Linux. Công cụ đơn giản nhất trong số này là lệnh `cmp` (**co**mpare). Lệnh này chỉ so sánh nội dung của hai tập tin theo từng byte:

```
[user]$ cmp tập_tin1 tập_tin2
```

Nếu hai tập tin hoàn toàn trùng nhau, thì lệnh hoàn thành công việc mà không đưa ra thông báo gì, còn nếu chúng khác nhau thì đưa ra số thứ tự của dòng và byte ở chỗ có sự khác nhau.

Tất nhiên thông tin mà lệnh `cmp` đưa ra hơi ít để có thể quyết định chọn tập tin nào trong số hai tập tin này, tập tin nào quan trọng hơn. Vì thế còn có thể sử dụng câu lệnh `diff` để biết được sự khác nhau giữa hai tập tin ở đây là gì. Chỉ cần cho câu lệnh này biết tên hai tập tin mà bạn muốn so sánh. Ví dụ:

```
teppi82@ThinhQuyen:~> diff ChangeLog ChangeLog2
1c1
< 11 tháng 09 năm 2006, phiên bản 0.9.3
---
> 18 tháng 09 năm 2006, phiên bản 0.9.3
```

Thông báo về sự khác nhau sẽ hiển thị trên màn hình (đầu ra tiêu chuẩn). Chúng ta có thể chuyển hướng báo cáo này vào một tập tin:

```
teppi@ThinhQuyen:~> diff ChangeLog ChangeLog2 > ChangeLog.diff
teppi@ThinhQuyen:~> more ChangeLog.diff
1c1
< 11 tháng 09 năm 2006, phiên bản 0.9.3
---
> 18 tháng 09 năm 2006, phiên bản 0.9.3
```

Để đánh giá phiên bản của một tập tin thì thuận tiện hơn nếu sử dụng câu lệnh `sdiff`. Kết quả so sánh khi này sẽ hiển thị ở dạng hai cột, phân cách nhau bởi các khoảng trắng. Nếu hai cột có cùng số thứ tự khác nhau, thì chúng sẽ cách nhau bởi một dấu gạch thẳng đứng "|". Nếu một dòng nào đó chỉ có trong tập tin thứ nhất thì nó được đánh dấu bằng một ký tự "<", nếu không có trong tập tin thứ hai – một ký tự ">".

Còn có câu lệnh `diff3` cho phép so sánh 3 tập tin một lúc. Nhưng thường sử dụng nhất trên các hệ thống UNIX và Linux vẫn là câu lệnh `diff`. Có thể dễ dàng hiểu điều này nếu biết rằng kết quả báo cáo của `diff` về sự khác nhau giữa hai tập tin có thể sử dụng cho câu lệnh `patch`. Thông thường khả năng này được dùng khi phân phối bản cập nhật của chương trình ứng dụng. Lấy ví dụ đơn giản ("một cách ngu ngốc"), một chương trình ứng dụng nào đó được đưa cho người dùng ở dạng tập tin `xvnkb-0.2.8.c`, có chứa mã nguồn của chương trình này trên ngôn ngữ C. Sau đó nhà phát triển sửa các lỗi và cập nhật chương trình rồi lưu mã nguồn trong một tập tin khác `xvnkb-0.2.9.c`. Bây giờ cần đưa những thay đổi đã làm đến cho người dùng. Tất nhiên là chỉ cần gửi cho người dùng những thay đổi, tức là kết quả báo cáo tạo ra bằng lệnh:

```
[chuoi]$ diff xvnkb-0.2.8.c xvnkb-0.2.9.c > xvnkb.c.diff
```

Như vậy tất nhiên là kích thước của tập tin `xvnkb.diff` sẽ nhỏ hơn nhiều so với `xvnkb-0.2.9.c`, sẽ tiết kiệm được dung lượng truyền tải qua mạng Internet nếu chỉ gửi `xvnkb.c.diff`. Trong trường hợp chương trình ứng dụng lớn thì sự tiết kiệm này là đánh kể. Tuy nhiên tiết kiệm cho người dùng không phải là ứng dụng chính. Mà sự phát triển của ứng dụng mã mở mới là điểm chính. Chúng ta nhớ rằng một ứng dụng mã mở được phát triển bởi cả một nhóm các nhà phát triển và cộng đồng. Mỗi nhà phát triển sẽ đóng góp đoạn mã của mình bằng chính cách này.

Tuy nhiên sau khi nhận được tập tin *.diff thì cần phải đưa những sửa đổi đã làm vào phiên bản hiện thời. Bài toán này là do `patch` giải quyết. Đã có `xvnkb-0.2.8.c` và `xvnkb.c.diff`, chỉ cần chạy lệnh:

```
[user]$ patch xvnkb-0.2.8.c xvnkb.c.diff > xvnkb-0.2.9.c
```

sẽ thu được tập tin `xvnkb-0.2.9.c`.

4.7 Các câu lệnh lưu trữ và nén tập tin

Khi sử dụng Linux rất có thể bạn sẽ ít làm việc với phần lớn các lệnh hệ vỏ `shell`, vì đã có những chương trình tiện ích như Midnight Commander và các

môi trường giao diện đồ hoạ như KDE, GNOME. Và các tiện ích và môi trường đó giúp bạn làm việc dễ dàng với các tập tin nén sẽ nói tới. Nhưng là người dùng Linux thực sự bạn cũng nên biết những câu lệnh nén và giải nén làm việc trong hệ vỏ `shell`. Chúng sẽ giúp bạn làm việc nhanh hơn với những tập tin nén thường gặp trong Internet.

Phương tiện chính để làm việc với các tập tin nén trong UNIX và Linux là hai chương trình `tar` và `gzip`. Mặc dù không ai cấm bạn sử dụng các chương trình arj, pkzip, lha, rar v.v... Nhưng truyền thống của Unix đó là `tar` và `gzip` và phần lớn mã nguồn (và không chỉ mã nguồn) của các chương trình ứng dụng được phân phối ở dạng này. Vì thế biết cách làm việc với `tar` và `gzip` đó là danh dự của bất kỳ người dùng Linux nào, cũng giống như samurai phải biết sử dụng kiếm vậy.

4.7.1 Chương trình `tar`

Những người dùng đã quen với những chương trình nén dạng winzip (đưa tất cả tập tin vào một "kho" rồi sau đó nén chúng) thì có thể sẽ hỏi "Tại sao lại cần hai chương trình?". Chúng ta sẽ tìm thấy câu trả lời cho câu hỏi này sau khi đọc xong hai phần tới đây, và sẽ thấy đây không phải là một yếu điểm của Linux mà còn là điểm mạnh nếu biết cách kết hợp "nhịp nhàng" hai chương trình này.

Tên chương trình `tar` có nghĩa là `t`ape **ar**chiver, tức là chương trình này không nén các dữ liệu mà chỉ kết hợp chúng vào một tập tin chung sau đó ghi tập tin này lên cách băng nhớ (tape) thường dùng để lưu trữ thông tin. Nếu muốn tạo tập tin này trên đĩa cứng, thì cần sử dụng lệnh `tar` với tùy chọn `f`, sau đó chỉ ra tên tập tin. Chương trình `tar` có 8 tùy chọn khác với những tùy chọn còn lại ở chỗ khi chạy lệnh `tar` thì cần phải đưa ra một trong số 8 tùy chọn này. Tám tùy chọn này xác định các chức năng chính của chương trình (xem bảng 4.5).

Bảng 4.5: Những tùy chọn chính của `tar`

Tùy chọn	**Ý nghĩa**
`-A, --catenate, --concatenate`	Thêm tập tin vào kho đã có.
`-c, --create`	Tạo kho mới.
`-d, --diff, --compare`	Tìm sự khác nhau giữa các tập tin trong kho và trên hệ thống tập tin (so sánh).
`--delete`	Xóa tập tin khỏi kho (không dùng cho băng ghi).
`-r, --append`	Thêm tập tin vào cuối kho.
`-t, --list`	Đưa ra danh sách các tập tin trong kho.
`-u, --update`	Chỉ thêm những tập tin mới hơn bản sao trong kho (cập nhật kho).
`-x, --extract, --get`	Lấy tập tin ra khỏi kho ("giải phóng").

Nếu bạn làm việc với các tập tin kho trên đĩa chứ không phải với băng ghi (đây là đa số trường hợp sử dụng máy tính cá nhân), thì nhất định phải dùng tùy chọn `f`. Những tùy chọn khác (trừ 8 tùy chọn bắt buộc kể trên) là không cần thiết, chúng chỉ dùng để thêm vào các chức năng phụ cụ thể nào đó. Ví dụ, tùy chọn `v` bắt buộc chương trình phải đưa ra danh sách các tập tin đưa vào kho.

Có thể liệt kê các tuỳ chọn một chữ cái (c, f, . . .) liền nhau và dấu gạch ngang (–) ở phía trước có thể dùng nhưng không nhất thiết phải có. Chúng ta sẽ thấy ở ví dụ tới đây. Tôi sẽ không đưa ra mô tả tất cả các tùy chọn của lệnh `tar`, mà chỉ cho biết một số dòng lệnh cần thiết nhất để làm việc với các kho tập tin. Như vậy sẽ có ích hơn trong thời gian này.

Để tạo một kho tập tin tar từ vài tập tin cần sử dụng một trong hai lệnh sau:

```
[user]$ tar –cf tên_kho tập_tin1 tập_tin2
[user]$ tar cf tên_kho tập_tin1 tập_tin2
```

trong đó tùy chọn `c` (từ bây giờ tôi sẽ bỏ đi dấu gạch ngang "–" ở phía trước tuỳ chọn) cho biết chương trình cần tạo ra (**c**reate) kho tập tin, còn tuỳ chọn `f` cho biết là kho này phải được tạo ở dạng tập tin có tên `tên_kho` ở phía sau tùy chọn.

Trong phần tên tập tin muốn lưu vào kho có thể sử dụng các "mẫu tên", bao gồm cả các ký tự thay thế đơn giản như "*" và "?". Nhờ vào tính năng này có thể lưu vào kho ngay lập tức nhiều tập tin bằng một câu lệnh rất ngắn. Ví dụ, để tạo ra kho chứa tất cả các tập tin của một thư mục con của thư mục hiện thời, giả sử `projects`, thì chỉ cần gọi lệnh:

```
[user]$ tar cf projects.tar projects/*
```

hoặc thậm chí còn đơn giản hơn:

```
[user]$ tar cf projects.tar projects
```

Câu lệnh này sẽ tạo ra kho `projects.tar` trong thư mục hiện thời. Kho này không chỉ lưu tất cả những tập tin có trong thư mục `projects` mà còn lưu tất cả những thư mục con của nó cùng với tất cả những tập tin nằm trong chúng (chúng ta dùng thuật ngữ "lưu đệ quy", hoặc "lưu toàn bộ cấu trúc thư mục"). Trong tập tin kho cấu trúc thư mục của `projects` vẫn được giữ nguyên.

Cần chú ý là trong ví dụ trên nếu ở chỗ "*" đặt "*.*" thì sẽ chỉ lưu những tập tin nằm trực tiếp trong thư mục `projects` và những thư mục con có dấu chấm trong tên (ít gặp), những thư mục con còn lại của `projects` sẽ không được lưu. Cũng trong ví dụ này nếu không chỉ ra tên thư mục thì sẽ lưu tất cả các tập tin và thư mục con của thư mục hiện thời. Nếu chạy lệnh sau:

```
[user]$ tar cvf tên_kho ./.*
```

thì không chỉ lưu tất cả các tập tin và thư mục con của thư mục hiện thời, mà còn lưu những tập tin của thư mục mẹ.

Bây giờ bạn đọc đã biết cách tạo kho tập tin. Để lấy (giải phóng) tất cả tập tin ra khỏi kho, cần dùng lệnh:

```
[user]$ tar xvf tên_kho
```

Hiển thị danh sách các tập tin trong kho bằng lệnh:

```
[user]$ tar tvf tên_kho | less
```

Giải phóng một tập tin nào đó bằng:

```
[user]$ tar xvf tên_kho tên_tập_tin
```

Chương trình `tar` là phương tiện thuận lợi để tạo các bản sao lưu trữ (sao lưu) của tập tin. Tất nhiên còn có những tiện ích sao lưu chuyên dùng khác, nhưng cả khi bạn đọc không biết những tiện ích này, thì vẫn có thể sao lưu dữ liệu quý báu của mình lên đĩa mềm bằng:

```
[user]$ tar Mcvf /dev/fd0 thư_mục
```

rồi phục hồi thư mục bằng lệnh:

```
[user]$ tar Mxpvf /dev/fd0
```

Hoặc đơn giản hơn, tạo một tập tin kho chứa:

```
[user]$ tar cvf tên_kho  thư_mục
```

rồi sao chép tập tin `tên_kho` lên một thiết bị lưu tháo rời hoặc thư mục trên mạng nào đó (CD, DVD, flash, ftp,...). Phục hồi lại bằng lệnh:

```
[user]$ tar xpvf tên_kho
```

Nếu có khó khăn khi sử dụng lệnh `tar`, hãy đọc trợ giúp man của nó hoặc đọc những thông tin hiện ra khi chạy `tar` với tùy chọn `--help`

4.7.2 Chương trình `gzip`

Mặc dù chương trình `tar` tạo ra kho tập tin, nhưng như đã nói ở trên, nó không nén kho này lại mà chỉ kết hợp các tập tin riêng rẽ vào một tập tin chung. Để nén tập tin này lại thường sử dụng câu lệnh `gzip`. Trường hợp đơn giản nhất của lệnh này trông như sau:

```
[user]$ gzip tên_tập_tin
```

Trên dòng lệnh có thể đưa ra cùng lúc vài tên hoặc "mẫu tên" tập tin. Nhưng khi này mỗi tập tin sẽ được nén riêng rẽ, chứ không tạo một tập tin chung.

Để *giải nén* tập tin hãy dùng một trong hai câu lệnh sau:

```
[user]$ gzip -d tên_tập_tin
```

hoặc

```
[user]$ gunzip tên_tập_tin
```

Tập tin ban đầu sau khi nén sẽ bị xóa, chỉ còn lại tập tin đã nén. Còn khi giải nén thì tập tin nén sẽ bị xóa, chỉ còn lại tập tin bình thường. Chúng ta có cảm giác như tập tin được "đưa ra, đưa vào" một kho. Nhưng đó là những tập tin **hoàn toàn khác nhau**! Hãy sử dụng lệnh `ls -i` để kiểm tra chỉ số inode của chúng.

Bây giờ chúng ta sẽ liệt kê một vài tùy chọn có ích của chương trình `gzip` vào bảng 4.6.

Bởi vì chương trình `gzip` không có khả năng lưu nhiều tập tin vào trong một tập tin, nên thường dùng `gzip` để nén những kho tập tin do `tar` tạo ra. Hơn nữa còn có thể sử dụng "kết hợp" hai chương trình này. Chúng ta sẽ xem xét vấn đề này ngay sau.

Bảng 4.6: Những tùy chọn chính của chương trình `gzip`

Tùy chọn	Ý nghĩa
`-h, --help`	Hiển thị trợ giúp ngắn gọn về cách sử dụng chương trình.
`-l, --list`	Đưa ra tên tập tin nằm trong tập tin nén, kích thước của nó và mức độ nén (tính theo %).
`-L, --license`	Hiển thị số phiên bản và bản quyền của chương trình.
`-N, --name`	Lưu hoặc phục hồi tên ban đầu và thời gian tạo tập tin.
`-n, --no-name`	Không lưu hoặc không phục hồi tên ban đầu và thời gian tạo tập tin.
`-q, --quiet`	Bỏ đi những cảnh báo.
`-r, --recursive`	Nén toàn bộ (đệ quy) thư mục. Sử dụng trong trường hợp có đưa ra "mẫu tên" tập tin.
`-S .suf, --suffix .suf`	Thêm phần sau (suffix) vào tên tập tin nén. Theo mặc định sử dụng phần sau là `gz`. ***Chú ý***: nếu sử dụng phần sau khác "gz" thì khi giải nén chương trình sẽ đưa ra thông báo lỗi dạng "unknown suffix – ignored" (phần sau không biết – lờ đi).
`-t, --test`	Thử (kiểm tra) tính nguyên vẹn của tập tin nén.
`-v, --verbose`	Đưa ra các thông báo phụ trong khi làm việc.
`-V, --version`	Hiển thị phiên bản của chương trình.
`-1, --fast`	Nén nhanh (mức độ nén thấp).
`-9, --best`	Mức độ nén mạnh hơn. Kích thước tập tin thu được nhỏ hơn nhưng tất nhiên là sẽ lâu hơn.

4.7.3 Chương trình `bzip2`

Trong thời gian gần đây người dùng Linux thường sử dụng thêm một chương trình nén nữa để thay thế cho `gzip` – chương trình `bzip2`. Chương trình này nén mạnh hơn (có mức độ nén cao hơn, tạo ra các tập tin nhỏ hơn) và làm việc nhanh hơn. Các bản phân phối Linux mới đã có cài đặt sẵn chương trình này.

Chương trình `bzip2` làm việc tương tự như `gzip`, tức là thay thế mỗi tập tin đưa vào dòng lệnh bằng phiên bản đã nén của tập tin đó, nhưng thêm vào phần cuối là `.bz2`. Tập tin nén có thời gian sửa đổi, quyền truy cập và có thể cả chủ sở hữu như tập tin ban đầu. Do đó có khả năng phục hồi những tính chất này của tập tin khi giải nén.

Trong một số trường hợp tập tin nén có thể còn lớn hơn theo kích thước so với tập tin ban đầu. Điều này có thể xảy ra đối với những tập tin có kích thước nhỏ hơn 100 byte, vì cơ chế nén sử dụng phần đầu (head) có kích thước 50 byte. Những tập tin gồm một chuỗi ngẫu nhiên (random) của các ký tự, trong đó có các tập tin nén, thì kích thước của tập tin tăng lên khoảng 0,5%.

Câu lệnh `bunzip2` hoặc `bzip2 -d` giải nén tập tin chỉ ra. Nếu tập tin này không phải do `bzip2` tạo ra thì chương trình sẽ không giải nén mà đưa ra lời cảnh báo. Khi giải nén `bzip2` sẽ đoán tên của tập tin sẽ tạo ra theo quy luật sau:

- `tên_tập_tin.bz2` thay thế bằng `tên_tập_tin`
- `tên_tập_tin.bz` thay thế bằng `tên_tập_tin`

- `tên_tập_tin.tbz2` thay thế bằng `tên_tập_tin.tar`
- `tên_tập_tin.tbz` thay thế bằng `tên_tập_tin.tar`
- `tên_tập_tin` khác thay thế bằng `tên_tập_tin.out`.

Ví dụ:

```
teppi82@ThinhQuyen:~> bzip2 2
teppi82@ThinhQuyen:~> mv 2.bz2 2.sj
teppi82@ThinhQuyen:~> bunzip2 2.sj
bunzip2: Can't guess original name for 2.sj -- using 2.sj.out
```

Các tuỳ chọn của `bzip2` rất giống với tùy chọn của `gzip` nhưng không phải tất cả. Tôi sẽ đưa ra một danh sách ngắn những tùy chọn cần thiết nhất trong bảng 4.7.

Bảng 4.7: Những tùy chọn chính của chương trình `bzip2`

Tùy chọn	**Ý nghĩa**
`-d, --decompress`	Bắt buộc giải nén tập tin. Tùy chọn này cần thiết vì trên thực tế `bzip2`, `bunzip2` và `bzcat` chỉ là một chương trình. Mỗi chương trình tuỳ theo "phần mở rộng" tập tin mà quyết định xem sẽ làm gì với tập tin đó. Tùy chọn `-d` bỏ đi cơ chế này và bắt buộc chương trình phải giải nén tập tin đã chỉ ra.
`-z, --compress`	Bắt buộc nén tập tin (xem ở trên).
`-t, --test`	Thử (kiểm tra) tính nguyên vẹn của tập tin nén.
`-f, --force`	Ghi chèn lên tập tin đã có. Theo mặc định `bzip2` không ghi chèn lên tập tin đã có trên đĩa. Nếu muốn ghi chèn thì hãy dùng tùy chọn này.
`-k, --keep`	Giữ gìn (không xóa) tập tin ban đầu khi nén hoặc giải nén.
`-s, --small`	Giảm yêu cầu đối với dung lượng bộ nhớ cần sử dụng bằng cách giảm tốc độ nén. Chỉ nên dùng tùy chọn này trên những máy tính cũ có ít bộ nhớ (8MB hoặc ít hơn). Có lẽ chúng ta không bao giờ cần đến tùy chọn này hoặc ít nhất là hy vọng như vậy.
`-q, --quiet`	Bỏ đi những cảnh báo ít ý nghĩa.
`-v, --verbose`	Đưa ra các thông báo phụ trong khi làm việc (chỉ có ý nghĩa chuẩn đoán).
`-L, --license, -V, --version`	Hiển thị số phiên bản và bản quyền của chương trình.

Tham số đứng sau hai dấu gạch ngang (`--`) và một khoảng trắng được coi là tên tập tin, dù tham số có một dấu gạch ngang ở đầu. Ví dụ:

```
[user]$ bzip2 -- -tên_tập_tin
```

Bây giờ đã đến lúc chúng ta học cách kết hợp `tar` với hai câu lệnh `gzip` và `bzip2` để làm việc nhanh và có hiệu quả hơn.

4.7.4 Sử dụng kết hợp `tar` với `gzip` và `bzip2`

Tất nhiên không ai ngăn cản bạn sử dụng riêng rẽ các câu lệnh `tar` với `gzip` và `bzip2`, nhưng sẽ nhanh hơn nếu chúng ta chỉ cần sử dụng một câu lệnh để có thể tạo ra một tập tin nén, hay giải nén một tập tin. Hãy tưởng tượng bạn nhận được một tập tin, ví dụ `xvnkb-0.2.9.tar.gz`. Để giải nén tập tin này thông thường bạn cần dùng hai câu lệnh sau tiếp nối nhau:

```
[user]$ gzip -d xvnkb-0.2.9.tar.gz
[user]$ tar xvf xvnkb-0.2.9.tar
```

Nhưng trong số những tùy chọn của chương trình `tar` còn có một tuỳ chọn đặc biệt `z` cho phép giải nén tập tin bằng chương trình `gzip` (thực hiện vai trò của lệnh thứ nhất trong hai lệnh kể trên). Để giải nén tập tin kể trên chỉ cần dùng một câu lệnh như sau:

```
[user]$ tar xzvf xvnkb-0.2.9.tar.gz
```

Kết quả thu được là hoàn toàn như trên trừ một điểm: trong trường hợp dùng 2 câu lệnh thì trên đĩa sẽ giữ lại tập tin trung gian (`xvnkb-0.2.9.tar`). Như vậy cách thứ hai còn có một ưu điểm nữa là không giữ lại các tập tin trung gian, người dùng không cần phải gõ thêm một lệnh xoá tập tin.

Việc tạo tập tin bằng cách sử dụng kết hợp `tar` và `gz` cũng được thực hiện bằng tùy chọn `z`:

```
[user]$ tar czvf tên_tập_tin.tar.gz thư_mục
```

Cần chú ý rằng trong trường hợp này chương trình sẽ không tự động thêm phần đuôi `.gz` vào tên của tập tin thu được. Do đó cần đặt tên rõ ràng cho nó cùng với phần đuôi `.tar.gz`.

Sử dụng kết hợp `tar` và `bzip2` là hoàn toàn tương tự, nhưng cần dùng tùy chọn `j` của `tar` để thay cho tuỳ chọn `z`. Tức là, giải nén tập tin bằng lệnh:

```
[user]$ tar xjvf tên_tập_tin.tar.bz2
```

và tạo tập tin nén bằng:

```
[user]$ tar cjvf tên_tập_tin.tar.bz2 thư_mục
```

Tôi nghĩ rằng những thông tin kể trên đã đủ để làm việc một cách có hiệu quả với các chương trình nén `tar`, `gzip` và `bzip2`. Để có thêm thông tin hãy tìm hiểu trang trợ giúp man hoặc các tài liệu HOWTO tương ứng.

Để kết thúc phần về những chương trình làm việc với tập tin nén này, tôi muốn nói thêm rằng, nếu "không may" bạn nhận được một tập tin dạng *.zip hay *.rar thì cũng đừng vội chạy sang nhờ một máy sử dụng Windows để giải nén. Bạn có thể thử các chương trình unzip và unrar có trên hệ thống Linux của mình. Cách sử dụng những lệnh này hết sức đơn giản, chỉ cần đưa vào dòng lệnh tên của tập tin. Nếu có gì khó khăn hãy thử "unzip –help" hoặc "unrar –help". Tôi chắc chắn là bạn sẽ tự giải quyết được vấn đề. Và đừng quên **chỉ** tạo ra các tập tin nén bằng `tar`, `gzip` hoặc `bzip2` vì bạn là người dùng Linux!

4.8 Tạo và gắn các hệ thống tập tin

Trong những phần trước chúng ta đã đề cập ngắn gọn một số câu lệnh chính để làm việc với những hệ thống tập tin đã định dạng sẵn. Bây giờ chúng ta sẽ dừng lại ở vấn đề làm sao để tạo ra hệ thống tập tin và cách thay đổi nó.

Cây thư mục của Linux được tạo ra từ những "cành" riêng rẽ tương ứng với các ổ đĩa khác nhau. Thường nói là cây thư mục được tạo thành từ các hệ thống tập tin riêng. Nói như vậy vì trong UNIX (và Linux) không có khái niệm "định dạng đĩa" mà sử dụng khái niệm "tạo hệ thống tập tin". Khi chúng ta có một đĩa lưu mới, ví dụ đĩa cứng, chúng ta cần tạo trên đĩa này hệ thống tập tin. Tức là mỗi đĩa được đặt tương ứng với hệ thống tập tin riêng. Để có thể sử dụng hệ thống tập tin này để ghi các tập tin, thì đầu tiên cần kết nối nó và cây thư mục chung (chúng ta sử dụng thuật ngữ "gắn", mount). Như vậy là có thể nói gắn hệ thống tập tin hoặc gắn đĩa lưu cùng với các hệ thống tập tin có trên nó.

Còn cần phải nói thêm rằng thông thường đĩa cứng được chia thành các phân vùng, nhất là đối với những đĩa mới sản xuất gần đây có dung lượng lớn từ vài chục đến vài trăm GB. Việc tạo những phân vùng như vậy giúp thực hiện dễ dàng các thao tác như: sao lưu, xác định quyền truy cập, đồng thời tăng hiệu suất làm việc và làm giảm khả năng mất thông tin do chương trình gây ra. Vì thế tiếp theo chúng ta sẽ nói về tạo hệ thống tập tin trên một phân vùng, những đĩa không bị chia có thể coi là một phân vùng.

Còn một điểm nữa cũng cần nói đến là Linux có thể làm việc với nhiều dạng hệ thống tập tin khác nhau. Nhưng hệ thống tập tin gốc của nó là "hệ thống tập tin mở rộng" (extfs) phiên bản 2 và 3. Ngoài hai hệ thống tập tin này Linux còn có thể làm việc với các "phiên bản" khác nhau của hệ thống tập tin FAT (FAT16 và FAT32), hệ thống tập tin ISO9660 sử dụng để ghi thông tin trên CD-ROM và các hệ thống tập tin khác (kể cả NTFS[15]). Tức là khi tạo và gắn các hệ thống tập tin cần luôn luôn nhớ rằng dạng hệ thống tập tin trên các đĩa lưu khác nhau có thể không giống nhau.

Đầu tiên chúng ta sẽ xem xét trường hợp cần tạo hệ thống tập tin trên một phân vùng nào đó (đã có) của đĩa. Ví dụ hệ thống tập tin có dạng ext3fs. Tạo hệ thống tập tin dạng ext3fs có nghĩa là tạo trên phân vùng này của đĩa một *siêu khối* (superblock), một bảng các mô tả inode, và các khối dữ liệu. Thực hiện tất cả những việc này bằng lệnh `mkfs`[16]. Trong trường hợp đơn giản nhất chỉ cần chạy lệnh sau:

```
[root]# mkfs -t ext3 /dev/hda2
```

Tất nhiên là cần thay thế `/dev/hda2` bằng tên của phân vùng trên máy của bạn. Hãy cẩn thận khi viết tên phân vùng, nếu ghi nhầm bạn sẽ bị mất dữ liệu. Nếu bạn muốn tạo hệ thống tập tin trên đĩa mềm thì cần chạy:

```
[root]# mkfs -t ext3 /dev/fd0
```

[15] Sự hỗ trợ đọc đã tốt, tuy nhiên sự hỗ trợ ghi lên NTFS chưa thật hoàn hảo.

[16] Trên các bản phân phối Linux mới còn có thể sử dụng các câu lệnh mkfs.ext2, mkfs.ext3 và các câu lệnh tương tự. Nếu dùng chúng thì chỉ cần bỏ đi phần -t ext3 hoặc -t ext2.

Có thể nói rằng chúng ta đã "định dạng đĩa mềm", nhưng cần biết là với hệ thống tập tin ext3fs thì bạn không đọc được đĩa mềm này trên DOS hoặc Windows (nếu không dùng chương trình hoặc driver đặc biệt). Để tạo ra những đĩa mềm có thể đọc trên DOS và Windows cần dùng tùy chọn `-t` với giá trị `vfat` hoặc những tiện ích đặc biệt khác. Nếu không đưa ra tùy chọn `-t` thì sẽ dùng dạng hệ thống tập tin mặc định (hiện nay là phiên bản cũ của ext – ext2fs).

Sau khi thực hiện câu lệnh `mkfs`, sẽ tạo ra hệ thống tập tin dạng ext3fs trong phân vùng chỉ ra. Trong hệ thống tập tin mới sẽ tự động tạo ra một thư mục với tên `lost`found+. Thư mục này được chương trình `fsck` dùng trong những trường hợp khẩn cấp, vì vậy đừng xóa nó. Để bắt đầu làm việc với hệ thống tập tin mới, đầu tiên cần kết nối (gắn) nó vào cây thư mục chung bằng lệnh `mount`.

Phải có ít nhất hai tham số cho câu lệnh `mount`: thiết bị (device, tên phân vùng) và *điểm gắn* (mount point). Điểm gắn là một thư mục đã có trong cây thư mục, và dùng làm "thư mục gốc" **đối với** hệ thống tập tin gắn vào (giống như nút nối giữa thân cây và cành cây). Ví dụ, câu lệnh:

```
[root]# mount /dev/hda10 /mnt/diaC
```

sẽ gắn hệ thống tập tin của phân vùng `/dev/hda10` vào thư mục `/mnt/diaC`. Cần phải có thư mục `/mnt/diaC` trong cây thư mục. Nếu chưa có hãy tạo ra bằng lệnh `mkdir`.

Cần chú ý là sau khi gắn hệ thống tập tin vào thư mục `/mnt/diaC`, thì người dùng không còn truy cập được tới nội dung (bao gồm cả thông tin về chủ sở hữu cũ và quyền truy cập tới chính bản thân thư mục) của thư mục này nữa. Nội dung này sẽ chỉ "quay trở lại" khi người dùng bỏ gắn (unmount) hệ thống tập tin ra khỏi thư mục. Nội dung cũ của thư mục không bị huỷ, bị xóa, mà chỉ tạm thời bị giấu đi. Vì thế tốt nhất là dùng các thư mục rỗng đã chuẩn bị sẵn từ trước để làm "điểm gắn" (vì thế mà trong tiêu chuẩn FHS có đề cập đến thư mục `/mnt`, hãy xem bảng 4.1).

Dạng đơn giản nhất trong ví dụ ở trên của lệnh `mount` chỉ làm việc với điều kiện tất cả những tham số còn thiếu có thể tìm thấy trong tập tin `/etc/fstab`. Nếu không có tập tin đó (chỉ khi nào bạn cố tình hoặc vô tình xóa) hoặc trong tập tin không có những dữ liệu cần thiết, thì cần sử dụng dạng đầy đủ của lệnh `mount`, như sau:

```
[root]# mount -t dạng_httt thiết_bị đường_dẫn
```

trong đó `dạng_httt` xác định dạng hệ thống tập tin trên `thiết_bị` (phân vùng), còn `đường_dẫn` xác định điểm gắn.

Tập tin cấu hình `/etc/fstab` chủ yếu dùng để gắn tự động các hệ thống tập tin trong quá trình khởi động Linux. Mỗi dòng của tập tin này chứa thông tin về một hệ thống tập tin và gồm 6 vùng phân cách nhau bởi các khoảng trắng[17]:

- **Tên thiết bị** (phân vùng). Có thể sử dụng tên thiết bị có trên máy (ví dụ /dev/hda10), cũng như tên của hệ thống tập tin mạng NFS (ví dụ ThinhQuyen:/home/nhimlui – thư mục `/home/nhimlui` trên máy có tên ThinhQuyen).

[17] Để tiện đọc các vùng thường sắp cho thẳng hàng, nhưng điều đó là không nhất thiết.

- **Điểm gắn**. Tên đầy đủ bao gồm cả đường dẫn của thư mục sẽ gắn tập tin vào.
- **Dạng hệ thống tập tin**.
- **Các tuỳ chọn gắn**. Theo mặc định là rw (đọc và ghi).
- **Mức độ dump**. Vùng này được chương trình sao lưu `dump` sử dụng. Nếu hệ thống tập tin cần được sao lưu thì ở đây phải có số 1, nếu không – số 0. Có thể có các giá trị khác, hãy xem trang man của dump.
- **Thứ tự ưu tiên kiểm tra** hệ thống tập tin bằng câu lệnh fsck. Hệ thống tập tin với giá trị nhỏ hơn sẽ được kiểm tra trước. Nếu bằng nhau thì sẽ kiểm tra song song (tất nhiên nếu có thể).

Hiện nay Linux hỗ trợ các hệ thống tập tin sau: minix, ext, ext2, ext3, xia, msdos, umsdos, vfat, proc, nfs, iso9660, hpfs, sysv, smb, ncpfs[18]. Ở chỗ dạng hệ thống tập tin trong vùng "**dạng hệ thống tập tin**" và sau tùy chọn `-t` của lệnh `mount` có thể đặt giá trị `auto`. Trong trường hợp đó câu lệnh `mount` thử tự xác định dạng của hệ thống tập tin đang gắn. Tuy nhiên trong một số trường hợp có thể dẫn đến lỗi, nên tốt hơn hết là chỉ ra dạng một cách chính xác. Còn có thể liệt kê một số dạng phân cách nhau bởi dấu phẩy (,). Trong câu lệnh `mount` còn có thể đưa ra danh sách các dạng hệ thống tập tin không cần gắn bằng cờ (flag) `no`. Khả năng này có ích trong trường hợp sử dụng câu lệnh `mount` với tham số `-a` (câu lệnh `mount` với tham số `-a` sẽ gắn tất cả các hệ thống tập tin liệt kê trong tập tin `/etc/fstab`). Ví dụ, câu lệnh:

```
[root]# mount -a -t nosmb,ext
```

gắn tất cả các hệ thống tập tin trừ các dạng `smb` (Samba[19]) và `ext`

Khi gắn hệ thống tập tin có trong `/etc/fstab`, thì chỉ cần đưa ra một tham số: hoặc tên của thiết bị (phân vùng) hoặc điểm gắn. Tất cả các tham số khác câu lệnh `mount` sẽ lấy từ tập tin `/etc/fstab`.

Thông thường chỉ có người dùng cao cấp root mới có khả năng gắn các hệ thống tập tin, nhưng nếu trong vùng **các tùy chọn gắn** có chỉ ra tùy chọn `user`, thì tất cả mọi người dùng sẽ có khả năng gắn (bỏ gắn) hệ thống tập tin đó. Ví dụ, nếu trong tập tin `/etc/fstab` có dòng:

```
/dev/hdd        /media/dvd      auto    noauto,user,sync 0 0
```

thì bất kỳ người dùng nào cũng có quyền gắn hệ thống tập tin trên đĩa DVD của mình bằng câu lệnh:

```
[user]$ mount /dev/hdd
```

hoặc:

```
[user]$ mount /media/dvd
```

Bảng 4.8: Những tùy chọn chính của câu lệnh `mount`

Tùy chọn	Ý nghĩa
`async`	Vào/ra (ghi/đọc) của hệ thống tập tin thực hiện không đồng bộ (không tức thời).
`auto`	Có thể gắn hệ thống bằng câu lệnh `mount` với tùy chọn `-a`.
`defaults`	Sử dụng các tuỳ chọn theo mặc định: `rw, suid, dev, exec, auto, nouser, async`.
`dev`	Các thiết bị khối và thiết bị ký tự (byte) trong hệ thống tập tin là những tập tin đặc biệt.
`noauto`	Chỉ có thể tự gắn hệ thống tập tin. Tùy chọn `-a` không tự động gắn hệ thống tập tin này.
`exec`	Cho phép thực hiện các tập tin chương trình nằm trên hệ thống tập tin này.
`remount`	Cho phép gắn lại hệ thống tập tin đã gắn. Thường sử dụng để thay đổi các tùy chọn gắn, đặc biệt trong trường hợp mở rộng quyền truy cập (ví dụ thêm quyền ghi cho hệ thống tập tin đã gắn chỉ đọc).
`ro`	Gắn hệ thống tập tin chỉ để đọc.
`rw`	Gắn hệ thống tập tin để đọc và ghi.
`suid`	Cho phép dùng "bit thay đổi ID người dùng" và "bit thay đổi ID nhóm".
`sync`	Vào/ra (ghi/đọc) của hệ thống tập tin thực hiện đồng bộ (tức thời).
`user`	Cho phép người dùng bình thường gắn hệ thống tập tin. Đối với những người dùng này luôn luôn gắn với các tùy chọn `noexec, nosuid, nodev`.
`nodev`	Không coi các thiết bị khối và thiết bị ký tự (byte) trong hệ thống tập tin là những tập tin đặc biệt.
`nosuid`	Không cho phép dùng "bit thay đổi ID người dùng" và "bit thay đổi ID nhóm".
`nouser`	Cấm người dùng bình thường gắn hệ thống tập tin.

Trong bảng 4.8 có đưa ra thêm một vài tùy chọn có thể sử dụng trong câu lệnh `mount` và trong tập tin `/etc/fstab` (vùng **các tùy chọn gắn**).

Nếu muốn gắn một hệ thống tập tin nào đó và chỉ cho phép đọc thì cần chỉ ra tùy chọn `r` (**r**ead only) trên dòng tương ứng của tập tin `/etc/fstab` (theo mặc định sử dụng `rw`, tức là đọc và ghi), hoặc sử dụng câu lệnh `mount` với tham số `-r`

Câu lệnh `mount` và `umount` hỗ trợ bảng các hệ thống tập tin đã gắn. Bảng này nằm trên đĩa ở dạng tập tin `/etc/mtab`. Có thể xem trực tiếp tập tin này bằng các chương trình xem tập tin (`less, more` hoặc bằng câu lệnh `mount` (không có tham số).

Trước khi bỏ các đĩa tháo rời (đĩa mềm, flash) ra khỏi máy thì cần "tháo" (bỏ gắn) các hệ thống tập tin có trên các đĩa tháo rời này. Thao tác này được thực hiện bằng câu lệnh `umount` (không phải **un**mount!). Tham số của câu lệnh `umount` là tên thiết bị (phân vùng) hoặc điểm gắn.

Chỉ có người dùng đã gắn hệ thống tập tin và tất nhiên cả người dùng cao

[18] Hãy xem trang man fs để đọc mô tả ngắn gọn về những hệ thống tập tin này.

[19] các chia sẻ trong mạng của Windows

cấp root mới có quyền bỏ gắn nó. Để bất kỳ người dùng nào cũng có thể bỏ gắn hệ thống tập tin thì trong tập tin `/etc/fstab` cần thay thế tùy chọn `user` bằng tùy chọn `users` (trong vùng **các tùy chọn gắn**).

Chỉ có thể bỏ gắn hệ thống tập tin khi không có tập tin nào của nó mở ra, không tiến trình nào đang chạy từ tập tin chương trình nằm trên hệ thống tập tin này và trong hệ thống không có tiến trình nào sử dụng hệ thống tập tin này. Tức là hệ thống tập tin không được bận.

Cần nói rằng nếu so với Windows thì làm việc với các đĩa rời (đĩa mềm, CD, DVD, Zip, v.v...) trên Linux có một chút phức tạp hơn. Vì đầu tiên bạn cần gắn các đĩa này (nói chính xác là hệ thống tập tin có trên đĩa) vào cây thư mục chung. Để thay một đĩa rời khác thì đầu tiên cần bỏ gắn ("tháo") đĩa đã có ra rồi mới gắn tiếp đĩa thứ hai. Tuy trên các bản phân phối mới đã có các dịch vụ cho phép tự động gắn và tự động "tháo" các đĩa rời, nhưng bạn cũng cần biết cách làm việc với các đĩa rời nếu có vấn đề xảy ra với các dịch vụ đó. Tốt nhất hãy chuẩn bị sẵn cho mỗi đĩa rời một "điểm gắn" riêng. Ví dụ, nếu bạn có một đĩa mềm, một ổ dvd và một flash thì hãy tạo ba thư mục `floppy`, `dvd` và `flash` trong `/mnt` để làm điểm gắn cho ba thiết bị của mình. Một số bản phân phối (Debian, openSUSE) sẽ tạo sẵn cho bạn những điểm gắn này.

Đây là tất cả những gì mà người dùng Linux mới (và rất mới) cần biết về hệ thống tập tin ext3fs. Xin nhắc lại là những gì đã nói ở đây chỉ dành cho hệ thống tập tin ext3fs (một số thông tin vẫn còn đúng cho phiên bản ext2fs), và mới chỉ đề cập đến "mặt trước", mặt quay về phía người dùng của hệ thống này (chủ yếu là cấu trúc tập tin). Mặt còn lại, mặt sau (cấu trúc bên trong), chỉ được nói đến trong chương này khi cần thiết. Chúng ta sẽ xem xét kỹ hơn mặt sau này trong một vài chương sắp tới. Còn bây giờ bạn đọc sẽ chuyển sang nghiên cứu thành phần quan trọng thứ 2 trong 4 thành phần chính của Linux – hệ vỏ `bash`.

Chương 5

Bash

Tốt gỗ hơn tốt nước sơn
– ca dao tục ngữ Việt Nam

Trong phần này chúng ta sẽ đề cập đến vấn đề làm việc với Linux ở chế độ text, hay còn được gọi là console hoặc terminal. Những người dùng Linux mới (newbie) thường nghĩ sẽ chẳng bao giờ làm việc ở chế độ này, vì đã có giao diện đồ họa. Tuy nhiên đây là một ý kiến sai lầm, bởi vì rất nhiều công việc có thể thực hiện nhanh và thuận tiện trong chế độ này hơn là sử dụng giao diện đồ họa. Và dù sao thì chế độ text của HĐH Linux không phải là chế độ text một tiến trình của MS-DOS. Vì Linux là HĐH đa tiến trình, nên ngay trong chế độ text đã có khả năng làm việc trong vài cửa sổ. Và để soạn thảo một tập tin văn bản không nhất thiết phải chạy các trình soạn thảo lớn và chậm chạp (đặc biệt trên các máy có cấu hình phần cứng thấp) của môi trường đồ họa.

5.1 Hệ vỏ là gì?

Chúng ta thường nói "người dùng làm việc với hệ điều hành". Điều này không hoàn toàn đúng, vì trên thực tế "liên hệ" với người dùng được thực hiện bởi một chương trình đặc biệt. Có hai dạng của chương trình đã đề cập - hệ vỏ, hay shell, để làm việc trong chế độ text (giao diện dòng lệnh) và giao diện đồ họa GUI (Graphical User Interface), thực hiện "liên hệ" với người dùng trong môi trường đồ họa. Cần nói ngay rằng, bất kỳ chương trình nào trong Linux có thể khởi động từ dòng lệnh của hệ vỏ (nếu máy chủ X đã chạy), cũng như qua giao diện đồ họa. Chạy chương trình từ dòng lệnh của hệ vỏ tương đương với việc nháy (đúp) chuột lên biểu tượng của chương trình trong GUI. Đưa các tham số cho chương trình trên dòng lệnh tương đương với việc chúng ta kéo và thả cái gì đó lên biểu tượng chương trình trong môi trường đồ họa. Nhưng mặt khác, một số chương trình không thể chạy ở GUI và chỉ có thể thực hiện từ dòng lệnh. Nói ngoài lề một chút, tên gọi "hệ vỏ" bị phản đối rất nhiều. Theo ý kiến của một số chuyên gia ngôn ngữ cũng như chuyên gia Linux thì nên gọi chương trình này một cách đúng hơn là "trình xử lý lệnh" hay "trình biên dịch lệnh". Tuy nhiên, tên gọi "hệ vỏ" (shell) được dùng cho các chương trình dùng để biên dịch lệnh trong chế độ text trên mọi hệ thống UNIX. Trên các hệ thống UNIX đầu tiên có một chương trình, gọi là `sh`, viết tắt của `shell`. Sau đó, vài biến thể của sh được phát triển và làm tốt hơn, trong đó có `Bourne shell` - phiên bản mở rộng của `sh`, viết bởi Steve

Bourne. Dự án GNU (dự án phát triển chương trình ứng dụng của Stallman, xem http://www.gnu.org/) sau đó cho ra đời hệ vỏ `bash`, tên gọi của nó được giải mã ra là `Bourne-again shell`, tức là "lại là hệ vỏ của Bourne". Trên tiếng Anh đây là một cách chơi chữ, vì từ Bourne đọc giống với từ borne (sinh ra, đẻ ra), và như thế bash còn có thể giải mã là "shell được sinh ra lần hai". Tiếp theo chúng ta sẽ chỉ xem xét `bash`, vì thế ở dưới khi nói đến hệ vỏ, xin ngầm hiểu đó là `bash` Tự một mình `bash` không thực hiện một công việc ứng dụng nào. Nhưng nó hỗ trợ việc thực thi mọi chương trình khác, từ việc tìm kiếm chương trình được gọi, chạy chúng đến việc tổ chức dữ liệu đầu vào/đầu ra. Ngoài ra, hệ vỏ chịu trách nhiệm về công việc với các biến môi trường và thực hiện một vài biến đổi (thế, hoán đổi vị trí) các tham số lệnh. Nhưng tính chất chính của hệ vỏ, nhờ đó đưa hệ vỏ trở thành một công cụ mạnh của người dùng, đó là nó bao gồm một ngôn ngữ lập trình đơn giản. Trong toán học từ lâu đã được chứng minh rằng, bất kỳ một thuật toán nào cũng có thể được xây dựng từ hai (ba) thao tác cơ bản và một toán tử điều kiện. Hệ vỏ cung cấp các toán tử điều kiện và toán tử vòng lặp. Nó sử dụng các tiện ích và chương trình khác (có trong thành phần hệ điều hành, hay được cài đặt riêng) để làm các thao tác cơ bản cho ngôn ngữ lập trình mà nó hỗ trợ. Đồng thời cho phép đưa các tham số cũng như kết quả làm việc của một chương trình tới các chương trình khác hay tới người dùng. Kết quả thu được là một ngôn ngữ lập trình mạnh. Đây cũng là sức mạnh và là một trong các chức năng chính của hệ vỏ. Trước khi bắt đầu phần này, bạn đọc nên biết các tổ hợp phím chính, sử dụng để điều khiển việc nhập dữ liệu trên dòng lệnh. Nên nhớ ít nhất cách sử dụng của các (tổ hợp) phím <Ctrl>+<C>, <Ctrl>+<D>, <Tab> và các phím có mũi tên.

5.2 Các ký tự đặc biệt

Hệ vỏ bash sử dụng một vài ký tự từ bộ 256 ký tự ASCII cho các mục đính riêng, hoặc để biểu thị các thao tác nào đó, hoặc để biến đổi biểu thức. Các ký tự này bao gồm:

```
` ~ ! @ # $ % ^ & * ( ) _ -- [ ] { } : ; ' " / \ > <
```

và ký tự với mã 0, ký tự hàng mới (tạo ra khi nhấn phím <Enter>) và ký tự khoảng trắng. Phụ thuộc vào tình huống các ký tự đặc biệt này có thể sử dụng với ý nghĩa đặc biệt của nó hay sử dụng như một ký tự thông thường. Nhưng trong đa số các trường hợp không khuyên dùng các ký tự với giá trị thứ hai. Trước hết đó là việc sử dụng chúng trong tên tập tin và thư mục. Tuy nhiên các ký tự _, - và . (dấu gạch dưới, gạch ngang và dấu chấm) thường được sử dụng trong tên tập tin, và đây là một ví dụ cho thấy không phải lúc nào chúng cũng có giá trị đặc biệt. Trong tên tập tin chỉ dấu chấm (.) và gạch chéo (/) có giá trị đặc biệt. Ký hiệu gạch chéo dùng để phân chia tên các thư mục trong đường dẫn, còn dấu chấm có giá trị đặc biệt khi nó là ký tự đầu tiên trong tên tập tin (cho biết tập tin là "ẩn"). Việc đưa ngay tất cả ý nghĩa đặc biệt của những ký tự này và các tình huống sử dụng chúng tạm thời không có ích. Chúng ta sẽ xem xét chúng dần dần trong các phần sau, khi cần sử dụng đến. Tuy nhiên, 3 ký hiệu có ý nghĩa lớn và cần đề cập đến đầu tiên. Ký hiệu \ (gạch chéo ngược) có thể gọi

là “ký hiệu xóa bỏ ý nghĩa đặc biệt” cho bất kỳ ký tự đặc biệt nào, đứng ngay sau `\`. Ví dụ, nếu muốn sử dụng khoảng trắng trong tên tập tin, thì chúng ta cần đặt trước ký tự khoảng trắng đó một dấu Ví dụ, câu lệnh sau:

```
teppi82@teppi:~$ cp lennon_imagine lennon\ imagine
```

Các ký tự `'` và `"` (ngoặc đơn và ngoặc kép) có thể gọi là “các ký tự trích dẫn”. Mỗi ký tự này **luôn luôn** được sử dụng trong một cặp với bản sao của chính nó để đóng khung một biểu thức nào đó, giống như trong các văn bản, sách báo, ... thông thường. Nếu như một đoạn văn bản nào đó đặt trong ngoặc đơn, thì tất cả các ký tự nằm trong ngoặc đơn này có giá trị như các ký tự thông thường, khôn một ký tự nào có ý nghĩa đặc biệt. Trở lại với ví dụ sử dụng khoảng trắng trong tên tập tin ở trên, có thể nói, nếu muốn đặt tập tin cái tên “lennon imagine” cần đưa tên đó vào dấu ngoặc:

```
teppi82@teppi:~$ cp lennon_imagine 'lennon imagine'
```

Sự khác nhau trong cách sử dụng ký tự `'` và `"` đó là, trong ngoặc đơn mất ý nghĩa đặc biệt **tất cả** các ký tự, còn trong ngoặc kép – tất cả chúng **ngoại trừ** `$, '` và `\` (dấu đô la, ngoặc đơn và dấu gạch ngược).

5.3 Thực thi các câu lệnh

Như đã nói ở trên, một trong các chức năng chính của hệ vỏ là tổ chức việc thực hiện các câu lệnh mà người dùng đưa vào trên dòng lệnh. Hệ vỏ, nói riêng, cung cấp cho người dùng hai thao tác đặc biệt để tổ chức việc đưa các câu lệnh trên dòng lệnh: `;` và `&`.

5.3.1 Thao tác ;

Mặc dù người dùng thường chỉ nhập trên dòng lệnh từng câu lệnh một, nhưng còn có thể đưa vào dòng lệnh đó ngay lập tức vài câu lệnh, và chúng sẽ thực hiện lần lượt từ câu lệnh này đến câu lệnh khác. Để làm được điều này cần sử dụng ký tự đặc biệt - `;`. Nếu dùng ký tự này để phân chia các câu lệnh, thì câu lệnh tiếp theo sẽ được coi như tham số của lệnh phía trước. Như vậy, nếu nhập vào dòng lệnh cái gì đó giống như sau:

```
teppi82@teppi:~$ command1 ; command2
```

thì hệ vỏ đầu tiên sẽ thực hiện câu lệnh command1, chờ cho lệnh đó hoàn thành, sau đó chạy command2, chờ lệnh hoàn thành, sau đó lại đưa ra dòng nhập lệnh và chờ các hành động tiếp theo của người dùng.

5.3.2 Thao tác &

Thao tác & được dùng để tổ chức việc thực hiện các câu lệnh trong chế độ nền sau. Nếu đặt dấu & ngay sau câu lệnh, thì hệ vỏ sẽ trả lại quyền điều khiển cho người dùng ngay sau khi chạy câu lệnh, mà không đợi cho câu lệnh đó hoàn thành. Ví dụ, nếu nhập vào dòng lệnh "`command1 & command2 &`", thì hệ vỏ chạy câu lệnh command1, ngay lập tức chạy lệnh command2, và sau đó không chậm trễ trả lại dòng nhập lệnh cho người dùng.

5.3.3 Thao tác && và ||

Các thao tác && và || lá những thao tác điều khiển. Nếu trên dòng lệnh là `command1 && command2`, thì command2 sẽ thực hiện và chỉ thực hiện trong trường hợp trạng thái thoát ra của lệnh command1 bằng không (0), tức là lệnh đó thực hiện thành công. Một cách tương tự, nếu dòng lệnh có dạng `command1 || command2`, thì command2 sẽ thực hiện và chỉ thực hiện khi trạng thái thoát của lệnh command1 khác không. Chúng ta sẽ không xem xét mặt kỹ thuật của việc thực hiện một câu lệnh nào đó. Chỉ có thể nói ngắn gọn rằng, hệ vỏ phải tìm mã (code) chương trình, nạp mã đó vào bộ nhớ, chuyển các tham số đã nhập trên dòng lệnh vào cho câu lệnh, và sau khi thực hiện xong thì theo một cách nào đó trả lại kết quả thực hiện lệnh này cho người dùng hay tiến trình khác. Chúng ta sẽ xem xét qua các bước này. Bước đầu tiên - tìm kiếm câu lệnh. Các câu lệnh chia thành hai loại: nội trú (mã của chúng có trong mã của chính hệ vỏ) và ngoại trú (mã của chúng nằm trong một tập tin riêng lẻ trên đĩa). Hệ vỏ luôn luôn tìm thấy lệnh nội trú, còn để tìm các lệnh ngoại trú người dùng, theo nguyên tắc, phải chỉ cho hệ vỏ đường dẫn đầy đủ tới tập tin tương ứng. Tuy nhiên để gỡ "gánh nặng" cho người dùng hệ vỏ biết cách tìm lệnh ngoại trú trong các thư mục, mà được liệt kê trong *đường dẫn tìm kiếm*. Chỉ khi (hệ vỏ) không thể tìm thấy tập tin cần thiết trong các thư mục đó, nó mới quyết định rằng người dùng đã nhầm khi nhập tên lệnh. Về cách thêm thư mục vào đường dẫn tìm kiếm chúng ta sẽ nói đến ở dưới, còn bây giờ chúng ta sẽ xem xét cách hệ vỏ tổ chức việc đưa dữ liệu vào cho câu lệnh đang thực hiện và việc đưa kết quả tới cho người dùng.

5.4 Đầu vào/đầu ra tiêu chuẩn

5.4.1 Dòng dữ liệu vào – ra

Khi một chương trình được thực hiện, nó được cung cấp ba dòng dữ liệu (hay còn gọi là kênh):

- *đầu vào tiêu chuẩn* (standard input hay stdin). Qua kênh này dữ liệu được đưa vào cho chương trình;
- *đầu ra tiêu chuẩn* (standard output hay stdout). Qua kênh này chương trình đưa ra kết quả làm việc của mình;

- *kênh thông báo lỗi tiêu chuẩn* (standard error hay stderr). Qua kênh này chương trình đưa ra thông tin về lỗi.

Từ đầu vào tiêu chuẩn chương trình chỉ có thể đọc, còn hai đầu ra và kênh thông báo lỗi được chương trình sử dụng chỉ để ghi. Theo mặc định đầu vào có liên kết[1] với bàn phím, còn đầu ra và kênh báo lỗi hướng đến terminal của người dùng. Nói cách khác, toàn bộ thông tin của lệnh hay chương trình mà người dùng đã chạy, và tất cả những thông báo lỗi, được đưa ra cửa sổ terminal. Tuy nhiên, chúng ta sẽ thấy ở dưới, có thể chuyển hướng thông báo đầu ra (ví dụ, vào tập tin). Để cho thấy kênh thông báo lỗi tiêu chuẩn làm việc như thế nào, hãy thực hiện câu lệnh `ls` với một tham số không đúng, ví dụ dùng tham số là một tên tập tin không tồn tại. Trong trường hợp này, `ls` đưa một tin nhắn báo lỗi ra kênh thông báo lỗi tiêu chuẩn. Tuy nhiên, đối với người dùng thì trong trường hợp này kênh thông báo lỗi tiêu chuẩn không khác gì với đầu ra tiêu chuẩn, bởi vì chúng ta cũng thấy thông báo lỗi đó trên cửa sổ terminal. Làm việc với đầu vào và đầu ra tiêu chuẩn được minh họa tốt nhất qua ví dụ các lệnh `echo` và `cat`.

5.4.2 Lệnh `echo`

Câu lệnh `echo` dùng để chuyển tới đầu ra tiêu chuẩn dòng ký tự, mà được đưa vào làm tham số cho nó. Sau đó lệnh này đưa ra tín hiệu chuyển dòng và hoàn tất công việc. Hãy thử thực hiện câu lệnh sau:

```
[user]$ echo 'xin chao cac ban!'
```

Tôi nghĩ rằng lời giải thích sẽ là thừa thãi (chỉ xin hãy sử dụng dấu ngoặc đơn, nếu không kết quả có thể sẽ khác. Nếu bạn đọc chú ý thì có thể giải thích tại sao lại khác).

5.4.3 Lệnh `cat`

Chúng ta sẽ xem xét lệnh `cat` ở đây vì lệnh này thường làm việc với đầu vào và đầu ra tiêu chuẩn. Theo mặc định kết quả làm việc của lệnh `cat` hướng tới đầu ra tiêu chuẩn. Để chứng minh là lệnh này theo mặc định tiếp nhận dòng dữ liệu nhập vào, hãy chạy lệnh `cat` không có tham số. Kết quả là con trỏ chuyển tới một dòng mới, và hơn nữa có vẻ như không có gì xảy ra. Lúc này câu lệnh chờ các ký tự đến từ đầu vào tiêu chuẩn. Hãy nhập bất kỳ ký tự nào, và nó sẽ xuất hiện ngay lập tức trên màn hình, tức là chương trình ngay lập tức đưa chúng tới đầu ra tiêu chuẩn. Có thể tiếp tục nhập các ký tự, và chúng cũng sẽ xuất hiện trên màn hình. Thông thường bàn phím được cấu hình để nhập vào theo từng dòng, vì thế nếu bạn nhấn phím <Enter>, dòng ký tự bạn vừa nhập sẽ được đưa tới lệnh `cat`, và lệnh này sẽ lại đưa dữ liệu ra màn hình **thông qua** đầu ra tiêu chuẩn. Như vậy, mỗi dòng ký tự nhập vào sẽ được hiện ra hay lần: một lần khi gõ và lần thứ hai bởi câu lệnh `cat`. Nếu nhấn tổ hợp phím <Ctrl>+<D>, mà dùng để ngừng việc nhập dữ liệu, chúng ta sẽ qua lại dòng nhập lệnh. Cũng có thể sử dụng tổ hợp phím <Ctrl>+<C>, mà là câu lệnh trong hệ vỏ để dừng chương trình

[1] giống liên kết hóa học

đang chạy. Nếu đưa tên một tập tin vào làm tham số cho lệnh `cat`, thì nội dung của lệnh này sẽ được đưa tới đầu vào tiêu chuẩn, từ đó lệnh `cat` sẽ đọc nội dung này và đưa tới đầu ra tiêu chuẩn (xem sơ đồ).

```
 Nội dung tập tin --> Đầu vào tiêu chuẩn (stdin) --cat--> đầu
ra tiêu chuẩn (stdout)
```

Đây chỉ là một trường hợp riêng của việc chuyển hướng dữ liệu đầu vào, một cơ chế rất có ích của hệ vỏ. Và tất nhiên chúng ta cần xem xét kỹ hơn cơ chế này.

5.5 Chuyển hướng đầu vào/đầu ra, đường ống và bộ lọc

Mặc dù, như đã nói ở trên, thông thường đầu vào/đầu ra của một chương trình liên kết với các đầu vào/đầu ra tiêu chuẩn, trong hệ vỏ còn có các môi trường đặc biệt cho phép chuyển hướng đầu vào/đầu ra.

5.5.1 Sử dụng >, < và »

Để chuyển hướng đầu vào/ra, sử dụng các ký hiệu “>”, “<” và “»”. Thường sử dụng việc chuyển hướng đữ liệu ra của câu lệnh vào tập tin. Dưới đây là một ví dụ tương ứng:

```
maikhai@fpt:/some/where$ ls -l > /home/maikhai/ls.txt
```

Theo lệnh này danh sách tập tin và thư mục con của thư mục, mà từ đó người dùng thực hiện lệnh `ls`[2] sẽ được ghi vào tập tin `/home/maikhai/ls.txt`; khi này nếu tập tin `ls.txt` không tồn tại, thì nó sẽ được tạo ra; nếu tập tin đã có, thì nội dung của nó sẽ bị xóa và ghi đè bởi danh sách nói trên. Nếu bạn không muốn xóa nội dung cũ mà ghi thêm dữ liệu đầu ra vào cuối tập tin, thì cần sử dụng ký hiệu >> thay cho >. Khi này khoảng trắng trước và sau các ký hiệu > hay >> không có ý nghĩa và chỉ dùng với mục đích thuận tiện, dễ nhìn. Bạn có thể chuyển hướng không chỉ vào tập tin, mà còn tới đầu vào của một câu lệnh khác hay tới một thiết bị nào đó (ví dụ, máy in). Ví dụ, để đưa nội dung tập tin `/home/maikhai/ls.txt` vừa tạo ở trên tới cửa sổ terminal thứ hai[3] có thể sử dụng lệnh sau:

```
maikhai@fpt:/sw$ cat /home/maikhai/ls.txt > /dev/tty2
```

Như bạn thấy, > dùng để chuyển hướng dữ liệu của đầu ra. Chức năng tương tự đối với đầu vào được thực hiện bởi <. Ví dụ, có thể đếm số từ trong tập tin `ls.txt` như sau (chú ý, đây chỉ là một ví dụ minh họa, trên thực tế thường sử dụng câu lệnh đơn giản hơn):

```
maikhai@fpt:/sw$ wc -w < /home/maikhai/ls.txt
```

[2] thư mục hiện thời

[3] bạn cần dùng tổ hợp phím <Ctrl>+<Alt>+<F2> để chuyển tới cửa sổ terminal này và đăng nhập trước

Cách chuyển hướng này thường được sử dụng trong các script, cho các câu lệnh mà thường tiếp nhận (hay chờ) dữ liệu vào từ bàn phím. Trong script dùng để tự động hóa một thao tác nào đó, có thể đưa các thông tin cần thiết cho câu lệnh từ tập tin: trong tập tin này ghi sẵn những gì cần để thực hiện lệnh đó. Bởi vì các ký hiệu <, > và >> làm việc với các kênh tiêu chuẩn (đầu vào hoặc đầu ra), chúng không chỉ được dùng theo các cách quen thuộc, thường dùng, mà còn có thể theo cách khác, "lạ mắt" hơn. Ví dụ, các câu lệnh sau là tương đương:

```
[user]$ cat > file
[user]$ cat>file
[user]$ >file cat
[user]$ > file cat
```

Tuy nhiên, tự chúng (không có một lệnh nào, tức là không có kênh tiêu chuẩn nào cho lệnh) các ký tự chuyển hướng này không thể được sử dụng, như thế không thể, ví dụ, nhập vào dòng lệnh sau:

```
[user]$ file1 > file2
```

mà thu được bản sao của một tập tin nào đó. Nhưng điều này không làm giảm giá trị của cơ chế này, bởi vì các kênh tiêu chuẩn có cho **mọi** câu lệnh. Khi này, có thể chuyển hướng không chỉ đầu vào và đầu ra tiêu chuẩn, mà còn các kênh khác. Để làm được điều này, cần đặt trước ký hiệu chuyển hướng **số** của kênh muốn chuyển. Đầu vào tiêu chuẩn stdin có số 0, đầu ra tiêu chuẩn stdout - số 1, kênh thông báo lỗi stderr - số 2. Tức là lệnh chuyển hướng có dạng đầy đủ như sau (xin được nhắc lại, khoảng trắng cạnh > là không nhất thiết):

```
command N > M
```

Trong đó, N và M - số của kênh tiêu chuẩn (0, 1, và 2) hoặc tên tập tin. Trong một vài trường hợp có sử dụng các ký hiệu <, > và >> mà không chỉ ra số kênh hay tên tập tin, vì vào chỗ còn thiếu sẽ đặt, theo mặc định, 1 nếu dùng >, tức là đầu ra tiêu chuẩn, hoặc 0 nếu dùng <, tức là đầu vào tiêu chuẩn. Như thế, khi không có số nào chỉ ra, > sẽ được biên dịch là `1 >`, còn < sẽ được biên dịch là `0 <`. Ngoài việc chuyển hướng các kênh tiêu chuẩn đơn giản như vậy, còn có khả năng không những chuyển hướng dữ liệu vào kênh này hay kênh khác, mà còn sao chép nội dung của các kênh tiêu chuẩn đó. Ký hiệu & dùng để thực hiện điều này, khi đặt nó (&) trước số của kênh sẽ chuyển dữ liệu **đến**:

```
command N > &M
```

Lệnh này có nghĩa là, đầu ra của kênh với số N được gửi đến cả đầu ra tiêu chuẩn, và sao chép tới kênh có số M. Ví dụ, để sao chép thông báo lỗi vào đầu ra tiêu chuẩn, cần dùng lệnh 2>&1, còn 1>&2 sao chép stdout vào stderr. Khả năng này đặc biệt có ích khi muốn ghi đầu ra vào tập tin, vì khi đó chúng ta vừa có thể nhìn thấy thông báo trên màn hình, vừa ghi chúng vào tập tin. Ví dụ, trường hợp sau thường được ứng dụng trong các script chạy khi khởi động Linux:

```
teppi82@teppi:~$ cat hiho > /dev/null
cat: hiho: No such file or directory
teppi82@teppi:~$ cat hiho > /dev/null 2>&1
```

5.5.2 Sử dụng |

Một trường hợp đặc biệt của chuyển hướng đầu ra là sự tổ chức các đường ống (ha y còn có thể gọi là kênh giữa các chương trình, hoặc băng chuyền). Hai hay vài câu lệnh, mà đầu ra của lệnh trước dùng làm đầu vào cho lệnh sau, liên kết với nhau (có thể nói phân cách nhau, nếu muốn) bởi ký hiệu gạch thẳng đứng - "|". Khi này đầu ra tiêu chuẩn của lệnh đứng bên trái so với | dược chuyển đến đầu vào tiêu chuẩn của chương trình, dứng bên phải so với |. Ví dụ:

```
maikhai@fpt:/sw$ cat ls.txt | grep knoppix | wc -l
```

Dòng này có nghĩa là kết quả của lệnh `cat`, tức là nội dung tập tin ls.txt, sẽ được chuyển đến đầu vào của lệnh `grep`, lệnh này sẽ phân chia nội dung nói trên và chỉ lấy ra những dòng nào có chứa từ knoppix. Đến lượt mình, kết qủa của lệnh `grep` được chuyển tới đầu vào của lệnh `wc -l`, mà tính số những dòng thu được. Đường ống sử dụng để kết hợp vài chương trình nhỏ lại với nhau (mỗi chương trình thực hiện một biến đổi xác định nào đó trên đầu vào) tạo thành một lệnh tổng quát, mà kết quả của nó sẽ là một biến đổi phức tạp. Cần chú ý rằng, hệ vỏ gọi và thực hiện tất cả các câu lệnh có trong đường ống cùng một lúc, chạy mỗi lệnh đó trong một bản sao hệ vỏ riêng. Vì thế ngay khi chương trình thứ nhất bắt đầu đưa kết quả ở đầu ra, chương trình tiếp theo bắt đầu xử lý kết quả này. Cũng y như vậy, các lệnh tiếp theo thực hiện các công việc của mình: chờ dữ liệu từ lệnh trước và đưa kết quả cho lệnh tiếp theo, giống như một dây chuyền sản xuất. Nếu như muốn một lệnh nào đó kết thúc **hoàn toàn**, trước khi thực hiện lệnh tiếp theo, bạn có thể sưẻ dụng trên một dòng cả ký hiệu dây chuyền |, cũng như dấu chấp phẩy ;. Trước mỗi dấu chấm phẩy, hệ vỏ sẽ dừng lại và chờ cho đến khi thực hiện xong tất cả các câu lệnh trước của đường ống. Trạng thái thoát ra (giá trị lôgíc, mà được trả lại sau khi thực hiện xong chương trình) của một đường ống sẽ trùng với trạng thái thoát ra của câu lệnh sau cùng trong đường ống. Ở trước câu lệnh đầu tiên của đường ống có thể đặt ký hiệu "!", khi đó trạng thái thoát ra của đường ống sẽ là phủ định lôgíc của trạng thát thoát ra của lệnh cuối cùng trong đường ống. Tức là nếu trạng thái thoát ra của lệnh cuối cùng bằng 0 thì trạng thái thoát ra của đường ống sẽ bằng 1 và ngược lại. Hệ vỏ chờ cho tất cả các câu lệnh kết thúc rồi mới xác định và đưa ra giá trị này.

5.5.3 Bộ lọc

Ví dụ cuối cùng ở trên (ví dụ với câu lệnh `grep`) có thể dùng để minh họa cho một khái niệm qua trọng khác, đó là, **bộ lọc chương trình**. Bộ lọc – đó là lệnh (hay chương trình), mà tiếp nhận dữ liệu vào, thực hiện một vài biến đổi trên dữ liệu này và đưa ra kết quả ở đầu ra tiêu chuẩn (từ đây còn có thể chuyển đến nơi nào đó theo ý muốn của người dùng). Các câu lệnh - bộ lọc bao gồm các lệnh đã nói đến ở trên `cat, more, less, wc, cmp, diff`, và cả những câu lệnh có trong bảng 5.1.

Một bộ lọc đặc biệt, câu lệnh `tee`, nhân đôi dữ liệu đầu vào, một mặt gửi dữ liệu này đến đầu ra tiêu chuẩn, mặt khác ghi nó (dữ liệu) vào tập tin (người dùng cần đặt tên). Dễ thấy rằng theo chức năng của mình lệnh `tee` tương tự như nhóm ký tự chuyển hướng 1>&file. Khả năng của bộ lọc có thể mở rộng với việc sử dụng

Bảng 5.1: Các câu lệnh bộ lọc

Lệnh	Mô tả ngắn gọn
grep, fgrep, egrep	Tìm trong tập tin hay dữ liệu đầu vào các dòng có chứa mẫu văn bản được chỉ ra và đưa các dòng này tới đầu ra tiêu chuẩn
tr	Trong dữ liệu đầu vào thay thế các ký tự ở ô thứ nhất bởi các ký tự tương ứng ở ô thứ hai. Hãy thử gõ lệnh `tr abc ABC` rồi gõ vài dòng chứa các ký tự abc!
comm	So sánh hai tập tin theo từng dòng một và đưa vào đầu ra tiêu chuẩn 3 cột : một - những dòng chỉ gặp ở tập tin thứ nhất, hai - những dòng chỉ gặp ở tập tin thứ hai, và ba - những dòng có trong cả hai tập tin.
pr	Định dạng tập tin hay nội dung của đầu tiêu chuẩn để in ấn.
sed	Trình soạn thảo tập tin theo dòng, sử dụng để thực hiện một vài biến đổi trên dữ liệu đầu vào (lấy từ tập tin hay đầu vào tiêu chuẩn)

các biểu thức chính quy (điều khiển), cho phép, ví dụ, tổ chức tìm kiếm theo các mẫu tìm kiếm từ đơn giản đến phức tạp và rất phức tạp. Nếu muốn, chúng ta có thể nói rất nhiều về chuyển hướng và bộ lọc. Nhưng nội dung này có trong phần lớn các cuốn sách về UNIX và Linux (xem phần lời kết). Vì vậy, chúng ta sẽ dừng ở đây và chuyển sang một phần khác, được gọi là "môi trường và các biến môi trường" tạo bởi hệ vỏ.

5.6 Tham biến và các biến số. Môi trường của hệ vỏ

Khái niệm tham biến trong hệ vỏ bash tương ứng với khái niệm biến số trong các ngôn ngữ lập trình thông thường. Tên gọi (hay ID) của tham biến có thể là một từ bao gồm các ký tự bảng chữ cái, chữ số, dấu gạch dưới (chỉ ký tự đầu tiên của từ này không được là chữ số), và cả những ký tự sau: , , #, , - (gạch ngang), $,, 0, _ (gạch dưới). Chúng ta nói rằng, tham biến được xác định hay được đặt ra, nếu người dùng gắn cho nó một giá trị. Giá trị có thể là một dòng trống rỗng. Để nhìn thấy giá trị của tham biến, người ta sử dụng ký tự $ ở trước tên của nó. Như vậy, lệnh:

```
maikhai@fpt:/sm$ echo parameter
```

hiển thị từ parameter, còn lệnh

```
maikhai@fpt:/sm$ echo $parameter
```

hiển thị giá trị của tham biến parameter (tất nhiên nếu như tham biến đó dược xác định).

5.6.1 Các dạng tham biến khác nhau

Tham biến chia thành ba dạng: tham biến vị trí, tham biến đặc biệt (các ký tự đặc biệt đã nói ở trên chính là tên của những tham biến này) và các biến số của hệ vỏ. Tên (ID) của tham biến vị trí gồm một hay vài chữ số (nhưng không có

tham biến vị trí 0). Giá trị của tham biến vị trí là các tham số cho lệnh, được đưa ra khi chạy hệ vỏ (tham số đầu tiên là giá trị của tham biến 1, tham số thứ hai - tham biến 2, v.v...). Có thể dùng câu lệnh `set` để thay đổi giá trị của tham biến vị trí. Giá trị của các tham biến này cũng thay đổi trong khi hệ vỏ thực hiện một trong các hàm số (chúng ta sẽ xem xét vấn đề này ở dưới). Các tham biến đặc biệt không gì khác hơn là các mẫu, mà sự thay thế (phép thế, phép hoán đổi) chúng được thực hiện như trong bảng 5.2:

Bảng 5.2: Thay thế các tham biến đặc biệt

Tham biến	Quy luật thay thế
*	Thay thế bởi các tham biến vị trí, bắt đầu từ tham biến thứ nhất. Nếu sự thay thế thực hiện trong dấu ngoặc kép, thì tham biến này sẽ được thay bởi **một từ duy nhất**, mà tạo ra từ **tất cả** các tham biến vị trí, phân cách nhau bởi ký tự đầu tiên của biến số IFS (sẽ nói đến ở sau). Tức là "$" tương đương với "$1c$2c..."", trong đó `c` - ký tự đầu tiên trong giá trị của biến số IFS. Nếu giá trị của IFS trống, hoặc không được xác định giá trị, thì tham biến phân cách nhau bởi các khoảng trắng.
@	Thay thế bởi tham biến vị trí, bắt đầu từ tham biến thứ nhất. Nếu thay thế thực hiện trong ngoặc kép, thì mỗi tham biến sẽ được thay thế bởi một từ riêng biệt. Tức là, "$@" tương đương với "$1" "$2" ... Nếu không có tham biến vị trí, thì giá trị sẽ không được thiết lập (tham biến sẽ bị x
#	Thay thế bởi giá trị thập phân của các tham biến vị trí.
?	Thay thế bởi trạng thái thoát ra của câu lệnh cuối cùng trong đường ống, mà được thực hiện trong chế độ nền trước.
- (gạch ngang)	Thay thế bởi giá trị các cờ, flag, được đặt bởi lệnh nội trú `set` hay trong khi chạy hệ vỏ.
$	Thay thế bởi số của tiến trình (PID - process identificator)
	Thay thế bởi số của tiến trình (PID) cuối cùng trong số các câu lệnh thực hiện trong nền sau.
0	Thay thế bởi tên hệ vỏ hay tên của script đang chạy. Nếu bash chạy một tập tin lệnh nào đó, thì $0 có giá trị bằng tên của tập tin này. Trong trường hợp ngược lại giá trị này bằng đường dẫn đầu đủ đến hệ vỏ (ví dụ, `/bin/bash`
_ (gạch dưới)	Thay thế bởi tham số cuối cùng của câu lệnh trước trong số các câu lệnh đã được thực hiện (nếu đó lại là một tham biến hay biến số, thì sẽ sử dụng giá trị của n

Các tham biến đặc biệt, được liệt kê ở bảng trên, có một điểm khác biệt đó là chỉ có thể "nhắc" đến chúng, không thể gán các giá trị cho các tham biến này. **Biến môi trường**, nhìn từ phía hệ vỏ, đó là các tham biến được đặt tên. Giá trị của biến môi trường được gán nhờ thao tác có dạng sau:

```
[user]$ name=value
```

Trong đó, `name` - tên của biến, còn `value` - giá trị muốn gán cho biến (có thể là một dòng trống). Tên của biến môi trường chỉ có thể bao gồm các chữ số, chữ cái và không được bắt đầu bởi một chữ số. (Tin rằng sau khi đọc đoạn viết về

tham biến vị trí thì các bạn đã hiểu tại sao một biến môi trường không thể bắt đầu bởi một chữ số.) Giá trị có thể là bất kỳ một dòng văn bản nào. Nếu giá trị có chứa những ký tự đặc biệt, thì cần đặt nó (giá trị) vào dấu ngoặc. Giá trị tất nhiên sẽ không chứa các dấu ngoặc này. Nếu một biến môi trường được xác định, thì cũng có thể bị xóa bỏ bằng lệnh nội trú `unset`. **Tập hợp** tất cả các biến này cùng với các giá trị đã gán cho chúng gọi là *môi trường* (enviroment) của hệ vỏ. Có thể xem nó (môi trường) nhờ lệnh `set` khi không có tham số (có thể cần dùng đường ống "`set | less`", nếu môi trường lớn, có nhiều biến). Để xem giá trị của một biến môi trường cụ thể, thay vì dùng lệnh `set` (khi này cần tìm trong kết quả của nó biến muốn xem), có thể sử dụng lệnh `echo`:

```
[user]$ echo $name
```

Trong đó, cần thay name bởi tên biến (như vậy, trong trường hợp này, bạn lại cần phải biết tên của biến muốn xem). Trong số các biến, mà bạn sẽ thấy trong kết quả của lệnh `set`, có những biến rất thú vị. Xin hãy chú ý đến, ví dụ, biến RANDOM. Nếu thử chạy vài lần liên tiếp câu lệnh sau:

```
maikhai@fpt:/sw$ echo $RANDOM
```

thì mỗi lần bạn sẽ nhận được một giá trị mới. Nguyên nhân là vì biến này trả lại một giá trị ngẫu nhiên[4] trong khoảng 0 - 32 768.

5.6.2 Dấu nhắc của hệ vỏ

Một trong các biến rất quan trọng có tên PS1. Biến này cho biết dạng của dấu nhắc, mà `bash` đưa ra trong khi chờ người dùng nhập câu lệnh tiếp theo. Theo mặc định thì biến này được gán giá trị "`\s-\v\$ `", tuy nhiên trên các bản phân phối khác nhau thường có các script khởi động (hay script đăng nhập) xác định lại biến này. Nói chung thì trong `bash` có tất cả bốn dấu nhắc, được sử dụng trong các trường hợp khác nhau. Biến PS1 đưa ra dạng của dấu nhắc khi hệ vỏ chờ nhập lệnh. Dấu nhắc thứ hai, xác định bởi biến PS2, xuất hiện khi hệ vỏ chờ người dùng nhập thêm một vài dữ liệu cần thiết nào đó để có thể tiếp tục chạy câu lệnh (chương trình) đã gọi. Theo mặc định biến PS2 có giá trị ">". Rất có thể bạn đã nhìn thấy dấu nhắc này, khi chạy lệnh `cat` để đưa dữ liệu vào từ bàn phím vào tập tin. Một ví dụ khác - lệnh `ftp`, sau khi chạy lệnh này dấu nhắc sẽ có dạng như đã nói. Dấu nhắc, xác định bởi biến PS3, sử dụng trong lệnh `select`. Dấu nhắc, xác định bởi biến PS4, được đưa ra trước mỗi câu lệnh, trong lúc `bash` theo dõi quá trình thực hiện. Giá trị theo mặc định - "+". Nếu có mong muốn, bạn có thể thay đổi các biến PS1 và PS2. Khi này có thể sử dụng bất kỳ ký tự nào nhập từ bàn phím, cũng như một vài ký tự chuyên dùng để xác định dạng dấu nhắc như trong bảng 5.3 (chỉ đưa ra một vài trong số chúng làm ví dụ, danh sách đầy đủ xem trong trang man của `bash` - gõ lệnh "`man bash`").

Số của lệnh (số thứ tự của lệnh đang thực hiện trong buổi làm việc hiện thời) có thể khác với số của chính nó trong danh sách "lịch sử các câu lệnh", bởi vì danh sách còn chứa cả những câu lệnh đã được ghi lại trong tập tin lịch sử. Sau

[4] random là từ tiếng Anh có nghĩa ngẫu nhiên

Bảng 5.3: Ký tự xác định dạng dấu nhắc

Cụm ký tự	**Giá trị** (kết quả thu được)
\a	Tín hiệu âm thanh (mã ASCII 07)
\d	Thời gian ở dạng "Thứ, tháng, ngày", ví dụ, Sun, Dec, 26.
\h	Tên máy (hostname) đến dấu chấm đầu tiên.
\H	Tên máy đầy đủ, ví dụ teppi.phanthinh.com
\t	Thời gian hiện thời ở dạng 24 giờ: HH:MM:SS (giờ:phút:giây)
\T	Thời gian hiện thời ở dạng 12 giờ: HH:MM:SS
\@	Thời gian hiện thời ở dạng 12 giờ am/pm (sáng/chiều)
\u	Tên người dùng đã chạy hệ vỏ, ví dụ teppi
\w	Tên đầy đủ của thư mục làm việc hiện thời (bắt đầu từ gốc), ví dụ /home/teppi82/project/l4u
\W	Thư mục hiện thời (không có đường dẫn)
\$	Ký tự #, nếu hệ vỏ được chạy bởi người dùng root, và ký tự $, nếu hệ vỏ được chạy bởi người dùng thường.
\nnn	Ký tự có mã hệ tám nnn
\n	Dòng mới (chuyển dòng)
\s	Tên hệ vỏ
\#	Số hiện thời của câu lệnh
\\	Dấu gạch ngược (backslash)
\[	Sau ký tự này tất cả các ký tự sẽ không được in ra.
\]	Kết thúc chuỗi các ký tự không được in ra.
\!	Số thứ tự của lệnh hiện thời trong lịch sử các câu lệnh đã dùng.

khi giá trị của biến được hệ vỏ đọc xong, sẽ xảy ra sự thay thế theo các quy luật mở rộng trong bảng trên, đồng thời còn xảy ra sự thay thế trong tên các câu lệnh, trong các biểu thức số học, và sự chia từ (word splitting). Chúng ta sẽ nói đến những sự thay thế này ở dưới.Ví dụ, sau khi thực hiện lệnh (vì trong dòng văn bản có khoảng trống, nên nhất thiết phải có dấu ngoặc):

```
[user/root]$ PS1="[\u@\h \W]\$"
```

thì trong dấu nhắc sẽ có dấu mở ngoặc vuông, tên người dùng, ký hiệu , tên máy, khoảng trắng, tên của thư mục hiện thời (không có đường dẫn), dấu đóng ngoặc vuông, và ký hiệu $ (nếu trên hệ vỏ đang làm việc người dùng bình thường) hay #(nếu hệ vỏ chạy dưới người dùng root).

5.6.3 Biến môi trường `PATH`

Còn một biến cũng quan trọng nữa có tên `PATH`. Biến này đưa ra danh sách đường dẫn đến các thư mục, mà `bash` sẽ tìm kiếm tập tin (trường hợp riêng là các tập tin lệnh) trong trường hợp, đường dẫn đầy đủ đến tập tin không được đưa ra. Các thư mục trong danh sách này phân cách nhau bởi dấu hai chấm (:). Theo mặc định biến môi trường `PATH` bao gồm các thư mục "bin" sau: `/usr/local/bin, /bin, /usr/bin, /usr/X11R6/bin`, tức là biến PATH trông như thế này: `/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin`. Để thêm thư mục vào danh sách này, cần thực hiện câu lệnh sau:

```
[user]$ PATH=$PATH:new_path
```

Khi tìm kiếm, hệ vỏ "lục lọi" các thư mục theo đúng thứ tự đã liệt kê trong biến `PATH`. Một chú ý nhỏ, có thể đưa vào danh sách này thư mục hiện thời, khi thêm vào biến `PATH` một dấu chấm (.). Tuy nhiên, đây là điều không khuyên làm với lý do bảo mật: người có ác ý có thể đặt vào thư mục dùng chung một chương trình nào đó, có cùng tên với một trong số những câu lệnh thường dùng bởi root, nhưng thực hiện những chức năng khác hoàn toàn (đặc biệt nguy hiểm nếu thư mục hiện thời đứng ở đầu danh sách tìm kiếm).

5.6.4 Biến môi trường `IFS`

Biến này xác định ký tự (cụm ký tự) phân cách (**I**nternal **F**ield **S**eparator), sử dụng trong thao tác phân chia từ ngữ khi biến đổi dòng lệnh, mà hệ vỏ thực hiện trước khi chạy một câu lệnh nào đó (xem dưới). Giá trị theo mặc định của biến này – "<Khoảng_trắng><Tab><Ký_tự_hàng_mới>". Nếu thử gõ lệnh "`echo $IFS`", bạn sẽ nhận được một ngạc nhiên nhỏ.

5.6.5 Thư mục hiện thời và thư mục cá nhân

Tên của thư mục hiện thời ghi trong biến môi trường với tên `PWD`, và giá trị của biến này thay đổi sau mỗi lần chạy chương trình `cd` (cũng như mỗi lần thay đổi thư mục hiện thời theo bất kỳ cách nào, ví dụ, qua Midnight Commander). Tương tự như vậy tên đầy đủ (gồm cả đường dẫn) của thư mục cá nhân của người dùng, chạy tiến trình đã cho, ghi trong biến `HOME`.

5.6.6 Câu lệnh `export`

Khi hệ vỏ chạy một chương trình hay câu lệnh nào đó, nó (hệ vỏ) cung cấp cho chúng một phần biến môi trường. Để có thể cung cấp biến môi trường cho tiến trình chạy từ hệ vỏ, cần gán giá trị cho biến này với lệnh `export`, tức là thay vì

```
[user]$ name=value
```

cần gõ:

```
[user]$ export name=value
```

Trong trường hợp này, tất cả các chương trình chạy từ hệ vỏ (kể cả bản sao thứ hai của chính hệ vỏ) sẽ có quyền truy cập tới các biến được gán như vậy, tức là sử dụng giá trị của chúng qua tên.

5.7 Khai triển biểu thức

Hay hệ vỏ đọc các câu lệnh như thế nào?.

Khi hệ vỏ nhận được một dòng lệnh này đó cần thực hiện, nó (hệ vỏ) trước khi chạy câu lệnh thực hiện việc "phân tích ngữ pháp" dòng lệnh này (giống trong

ngôn ngữ, phân tích chủ ngữ, vị ngữ). Một trong những bước của sự phân tích này là **phép mở** hay **khai triển** biểu thức (expansion). Trong `bash` có bảy loại khai triển biểu thức:

- Khai triển dấu ngoặc (brace expansion);
- Thay thế dấu ngã (tilde expansion);
- Phép thế các tham biến và biến số;
- Phép thế các câu lệnh;
- Phép thế số học (thực hiện từ trái sang phải);
- Phép chia từ (word splitting);
- Khai triển các mẫu tên tập tin và thư mục (pathname expansion).

Các thao tác này được thực hiện theo đúng thứ tự liệt kê trên. Chúng ta sẽ xem xét chúng theo tứ tự này.

5.7.1 Khai triển dấu ngoặc

Khai triển dấu ngoặc tốt nhất minh họa trên ví dụ. Giả thiết, chúng ta cần tạo thư mục con trong một thư mục nào đó, hoặc thay đổi người dùng sở hữu của vài tập tin **cùng một lúc**. Có thể thực hiện điều này nhờ các câu lệnh sau:

```
[user]$ mkdir /usr/src/unikey/{old,new,dist,bugs}
[root]# chown root /usr/{ucb/{ex,edit},lib/{ex?.?*,how_ex}
```

Trong trường hợp đầu, trong thư mục `/usr/src/unikey/` sẽ tạo ra các thư mục con `old`, `new`, `dist`, và `bugs`. Trong trường hợp thứ hai, người dùng sở hữu của các tập tin sau sẽ thay đổi (thành root):

- /usr/ucb/ex
- /usr/lib/ex?.?
- /usr/ucb/edit
- /usr/lib/ex?.?
- /usr/ucb/ex
- /usr/lib/how_ex
- /usr/ucb/edit
- /usr/lib/how_ex

Tức là với **mỗi cặp** dấu ngoặc sẽ tạo ra vài dòng **riêng rẽ** (số những dòng này bằng số từ nằng trong dấu ngoặc) bằng cách ghi thêm vào trước mỗi từ trong ngoặc những gì đứng trước dấu ngoặc, và ghi thêm vào sau mỗi từ này những gì đứng sau dấu ngoặc. Một ví dụ khác: dòng `a{d,c,b}e` khi khai triển sẽ thu được ba từ "`ade ace abe`". Khai triển dấu ngoặc được thực hiện trước các dạng khai triển khác trong dòng lệnh, hơn nữa tất cả các ký tự đặc biệt có trong dòng lệnh, kể cả những ký tự nằm trong dấu ngoặc, sẽ được giữ không thay đổi (chúng sẽ được biên dịch ở các bước phía sau).

5.7.2 Thay thế dấu ngã (Tilde Expansion)

Nếu như từ bắt đầu với ký tự dấu ngã ('∼'), tất cả các ký tự đứng trước dấu gạch chéo đầu tiên (hay tất cả các ký tự nếu như không có dấu gạch chéo) sẽ được hiểu là tên người dùng (login name). Nếu như tên này là một dòng rỗng (tức là dấu gạch chéo đứng ngay phía sau dấu ngã), thì dấu ngã sẽ được thay thế bởi giá trị của biến `HOME`. Và nếu giá trị của biến `HOME` không được gán thì dấu ngã sẽ được thay thế bởi đường dẫn đầu đủ đến thư mục cá nhân của người dùng, mà đã chạy hệ vỏ. Nếu như ngay sau dấu ngã (và trước dấu gạch chéo) là một từ trùng với tên của một người dùng hợp pháp, thì dấu ngã cộng với tên người dùng được thay thế bởi đường dẫn đầy đủ đến thư mục cá nhân của người dùng này. Nếu như từ đứng sau dấu ngã không phải là tên của một người dùng (và không rỗng), thì từ không bị thay đổi. Nếu như sau dấu ngã là '+', hay ký hiệu này sẽ được thay thế bởi tên đầu đủ của thư mục hiện thời (tức là giá trị của biến `PWD`). Nếu dứng sau dấu ngã là '-', thì thay thế giá trị của biến `OLDPWD` (thư mục "cũ").

5.7.3 Phép thế các tham biến và biến số

Ký tự $ được sử dụng cho các thao tác thế tham biến, thế các câu lệnh và thế các biểu thức số học. Biểu thức hay tên đứng sau $ có thể được đưa vào ngoặc, không nhất thiết, nhưng rất tiện, vì dấu ngoặc phân cách biểu thức với các từ hay ký tự đứng sau. Như vậy, để gọi giá trị của tham biến nói chung cũng như biến môi trường nói riêng trong dòng lệnh, cần đặt biểu thức dạng $parameter. Dấu ngoặc chỉ cần thiết, nếu tên của tham biến có chứa vài chữ số, hoặc khi theo sau tên còn có các ký tự khác, mà chúng ta không muốn hệ vỏ "hiểu lầm" chúng là một phần của tên tham biến. Trong tất cả các giá trị của biến số xảy ra phép thế dấu ngã (∼), sự khai triển tham biến và biến số, phép thế các câu lệnh, phép thế các biểu thức số học, cũng như xóa các ký tự trích dẫn (xem dưới). Sự phân chia từ không xảy ra, trừ trường hợp "$" (lời giải thích xem ở bảng số 3). Sự khai triển các mẫu tên tập tin và thư mục cũng không được thực hiện.

5.7.4 Phép thế các câu lệnh

Phép thế các câu lệnh là một công cụ rất mạng của `bash`. Ý nghĩa của nó nằm ở chỗ thay thế tên các câu lệnh bởi kết quả thực hiện của chúng. Có hai dạng phép thế lệnh: `$(command)` và `` `command` `` Nếu ứng dụng dạng thứ hai (chú ý ở đây sử dụng dấu "ngoặc đơn ngược", phím cho nó thường nằm trên phím Tab), thì dấu gạch ngược (\) ở trong dấu ngoặc sẽ có chức năng như một ký tự thông

thường, trừ trường hợp, khi đứng sau nó (dấu gạch ngược) là một $, `, hay một \. Nếu như sử dụng dạng $(command), thì tất cả các ký tự đứng trong ngoặc tạo thành một câu lệnh, không có ký tự nào có ý nghĩa đặc biệt. Nếu phép thế câu lệnh xảy ra phía trong ngoặc kép, thì trong kết quả của phép thế sẽ không thực hiện phép phân chia từ và sự khai triển mẫu tên tập tin và thư mục.

5.7.5 Phép thế số học (Arithmetic Expansion)

Phép thế số học cho phép tính giá trị của một biểu thức số học và thay thế nó (biểu thức) bởi kết quả thu được. Có hai dạng phép thế số học: $[expression] ((expression)) Trong đó expression được hiểu (được bash đọc) như khi đứng trong ngoặc kép, nhưng những dấu ngoặc kép ở trong expression lại được đọc như một ký tự thường. Phía trong expression có thực hiện các phép thế tham biến và thế câu lệnh. Cú pháp của biểu thức expression tương tự như cú pháp của biểu thức số học của ngôn ngữ C, cụ thể hơn về vấn đề này có thể đọc trong phần ARITHMETIC EVALUATION của trang man của bash. Ví dụ, câu lệnh

```
[user]$ echo $((2 + 3 * 5))
```

cho kết quả bằng “17”. Nếu biểu thức không chính xác, bash sẽ đưa ra thông báo lỗi.

5.7.6 Phân chia từ (word splitting)

Sau khi thực hiện xong các phép thế tham biến, thế lệnh, và thế các biểu thức số học, hệ vỏ lại phân tích dòng lệnh một lần nữa (nhưng ở dạng thu được sau các phép thế nói trên) và thực hiện việc phân chia từ (word splitting). Thao tác này nằm ở chỗ, hệ vỏ tìm trong dòng lệnh tất cả các ký tự phân chia, xác định bởi biến IFS (xem trên), và nhờ đó chia nhỏ dòng lệnh thành các từ riêng rẽ trong các chỗ tương ứng. Nếu giá trị của IFS bằng một dòng trống, thì việc phân chia từ sẽ không xảy ra. Nếu trong dòng lệnh không thực hiện phép thế nào trong các phép thế kể trên, thì phân chia từ cũng không xảy ra.

5.7.7 Khai triển các mẫu tên tập tin và thư mục (Pathname Expansion)

Phép thế tên đường dẫn và tập tin (Pathname expansion) sử dụng để chỉ nhờ một mẫu nhỏ gọn mà có thể chỉ ra vài tập tin (hay thư mục), tương ứng với mẫu này. Sau khi phân chia từ, và nếu như không đưa ra tùy chọn -f, thì bash sẽ tìm kiếm trong từng từ của dòng lệnh các ký tự *, ?, và [. Nếu tìm thấy từ với một hay vài ký tự như vậy, thì từ này sẽ được xem như một mẫu, và cần thay thế bởi các từ trong danh sách đường dẫn, tương ứng với mẫu này. Nếu như không tìm thấy tên tương ứng với mẫu, và biến nullglob không được đưa ra, thì từ sẽ không thay đổi, tức là các ký tự đặc biệt bị mất giá trị và hiểu như các ký tự thường. Nếu như biến này được xác định, mà đường dẫn tương ứng với mẫu không tìm thấy, thì từ sẽ bị xóa khỏi dòng lệnh. Các ký tự dùng để tạo mẫu có các giá trị trong bảng 5.4.

Bảng 5.4: Các ký tự tạo mẫu

Ký tự	Quy luật thay thế
*	Tương ứng với bất kỳ dòng ký tự nào, kể cả dòng rỗng. Ví dụ, `v*.txt` sẽ được thay thế bởi `vnoss.txt, vnlinux.txt` và `vntex.txt` (nếu các tập tin này tồn tại), và *.png sẽ tương ứng tất cả các tập tin có phần mở rộng png (tập tin đồ họa hai chiều).
?	Tương ứng bất ký một ký tự đơn nào. Ví dụ, mẫu `file?.txt` sẽ được thaythế bởi các tên tệp sau `file1.txt, file2.txt, file3.txt,` và `filea.txt` (nếu chúng tồn tại), nhưng `file23.txt` thì không.
[...]	Tương ứng bất kỳ ký tự nào trong số các ký tự nằm trong dấu ngoặc vuông này. Cặp ký tự, phân cách nhau bởi **dấu trừ** (-), ví dụ `c-f`, biểu thị một dãy; bất ký ký tự nào, theo từ điển, nằm giữa hai ký tự này, kể cả hai ký tự tạo ra dãy (`c` và `f` trong ví dụ) cũng tương ứng với mẫu. Nếu ký tự đầu tiên trong ngoặc vuông là hay ¸ thì mẫu (ở vị trí này) sẽ tương ứng tất cả các ký tự, **không** được chỉ ra trong ngo

Mẫu tên tập tin rất thường xuyên sử dụng trong dòng lệnh có chứa `ls`. Hãy tưởng tượng là bạn muốn xem thông tin của một thư mục, trong đó có chứa một số lượng lớn các tập tin đủ các dạng, ví dụ, tập tin hình ảnh, phim với dạng gif, jpeg, avi, v.v.... Để thu được thông tin **chỉ** của tập tin dạng jpeg, có thể dùng câu lệnh

```
[user]$ ls *.jpg
```

Nếu trong thư mục có nhiều tập tin, mà tên của chúng là các số gồm bốn chữ số (thư mục `/proc` là một ví dụ+), thì lệnh sau chỉ đưa ra danh sách các tập tin có số từ 0500 đến 0999:

```
[user]$ ls -l 0[5-9]??
```

5.7.8 Xóa các ký tự đặc biệt

Sau khi làm xong tất cả các phép thế, các ký tự `\, ` và " còn lại trong dòng lệnh (chúng được sử dụng để huỷ bỏ giá trị đặc biệt của các ký tự khác) sẽ bị xóa hết.

5.8 Shell - một ngôn ngữ lập trình

Như đã nói ở trên, để có thể xây dựng bất kỳ giải thuật nào, cũng cần có các toán tử kiểm tra điều kiện. Hệ vỏ `bash` hỗ trợ các toán tử lựa chọn `if...then...else` và `case`, cũng như các toán tử vòng lặp `for, while, until`, nhờ đó nó (`bash`) trở thành một ngôn ngữ lập trình mạnh.

5.8.1 Toán tử `if` và `test` (hoặc [ ])

Cấu trúc của toán tử điều kiện có dạng **thu gọn** như sau:

`if` list1 `then` list2 `else` list3 `fi`

trong đó, `list1, list2`, và `list3` là các chuỗi câu lệnh, phân cách nhau bởi dấu phảy và kết thúc bởi một dấu chấm phẩy hay ký tự dòng mới. Ngoài ra, các chuỗi này có thể được đưa vào dấu ngoặc nhọn: list. Toán tử if kiểm tra giá trị được trả lại bởi các câu lệnh từ `list1`. Nếu trong danh sách có vài câu lệnh, thì kiểm tra giá trị được trả lại bởi câu lệnh **cuối cùng** của danh sách. Nếu giá trị này bằng 0, thì sẽ thực hiện các lệnh từ `list2`; còn nếu giá trị này khác không, thì sẽ thực hiện những lệnh từ `list3`. Giá trị được trả lại bởi toán tử `if` như vậy, trùng với giá trị mà chuỗi lệnh thực hiện (`list2` hoặc `list3`) đưa ra. Dạng **đầy đủ** của lệnh `if` như sau:

`if` list `then` list [`elif` list `then` list] ... [`else` list] `fi`

(ở đây dấu ngoặc vuông chỉ có nghĩa là, những gì nằm trong nó, ngoặc vuông, không nhất thiết phải có). Biểu thức đứng sau `if` hay `elif` thường là câu lệnh `test`, mà có thể được biểu thị bởi dấu ngoặc vuông [ ]. Lệnh `test` thực hiện phép tính một biểu thức nào đó, và trả lại giá trị 0, nếu biểu thức là đúng, và 1 trong trường hợp ngược lại. Biểu thức được đưa tới chương trình `test` như một tham số của chương trình. Thay vì gõ `test expression` có thể đưa biểu thức `expression` vào ngoặc vuông: `[ expression ]`

Cần chú ý rằng, `test` và `[` đó là hai tên của của cùng một chương trình, chứ không phải là một phép biến hóa thần thông nào đó của hệ vỏ `bash` (chỉ là cú pháp của `[` đòi hỏi phải có dấu đóng ngoặc). Và cũng cần chú ý rằng ở chỗ của `test` trong cấu trúc `if` có thể sử dụng bất kỳ chương trình nào. Để kết thúc mục này, chúng ta đưa ra ví dụ sử dụng `if`:

```
if [ -x /usr/bin/unicode_start ] ; then
unicode_start
else
echo "hello world"
fi
```

Về toán tử `test` (hay `[...]`) cần đi sâu hơn.

5.8.2 Toán tử test và điều kiện của biểu thức

Biểu thức điều kiện, sử dụng trong toán tử `test`, được xây dựng trên cơ sở *kiểm tra thuộc tính tập tin, so sánh các dòng* và *các so sánh số học thông thường*. Biểu thức phức tạp hơn được tạo ra từ các thao tác đơn và kép sau ("những viên gạch cơ sở"):

- `-a file`
  Đúng nếu tập tin có tên `file` tồn tại.

- `-b file`
  Đúng nếu `file` tồn tại, và là một tập tin thiết bị khối (block device) đặc biệt.

- `-c file`
  Đúng nếu `file` tồn tại, và là một tập tin thiết bị ký tự (charater device) đặc biệt.

- `-d file`
  Đúng nếu `file` tồn tại và là một thư mục.

- `-e file`
  Đúng nếu tập tin có tên `file` tồn tại.

- `-f file`
  Đúng nếu tập tin có tên `file` tồn tại và là một tập tin thông thường.

- `-g file`
  Đúng nếu tập tin có tên `file` tồn tại và được đặt *bit thay đổi nhóm*.

- `-h file` hay `-L file`
  Đúng nếu tập tin có tên `file` tồn tại và là liên kết mềm (liên kết tượng trưng).

- `-k file`
  Đúng nếu tập tin có tên `file` tồn tại và được đặt *bit sticky*.

- `-p file`
  Đúng nếu tập tin có tên `file` tồn tại và là tên của một ống (kênh FIFO).

- `-p file`
  Đúng nếu tập tin có tên `file` tồn tại và là tên của một ống (kênh FIFO).

- `-r file`
  Đúng nếu tập tin có tên `file` tồn tại và có quyền đọc.

- `-s file`
  Đúng nếu tập tin có tên `file` tồn tại và kích thước lớn hơn không.

- `-t fd`
  Đúng nếu bộ mô tả của tập tin (fd) mở và chỉ lên terminal.

- `-u file`
  Đúng nếu tập tin có tên `file` tồn tại và được đặt *bit thay đổi người dùng*.

- `-w file`
  Đúng nếu tập tin có tên `file` tồn tại và có quyền ghi.

- `-x file`
  Đúng nếu tập tin có tên `file` tồn tại và có quyền thực thi.

- `-O file`
  Đúng, nếu tập tin có tên `file` và chủ sở hữu của nó là người dùng mà ID có hiệu lực chỉ đến.

- `-G file`
  Đúng, nếu tập tin có tên `file` tồn tại và thuộc về nhóm, xác định bởi ID nhóm có hiệu lực.

- `-S file`
  Đúng, nếu tập tin có tên `file` tồn tại và là socket.

- `-N file`
  Đúng, nếu tập tin có tên `file` tồn tại và thay đổi từ lần được đọc cuối cùng.

- `file1 -nt file2`
  Đúng, nếu tập tin `file1` có *thời gian sửa đổi* muộn hơn `file2`.

- `file1 -ot file2`
  Đúng, nếu tập tin `file1` "già" hơn `file1` (trường hợp ngược lại của trường hợp trên).

- `file1 -ef file2`
  Đúng, nếu tập tin `file1` và `file2` có cùng một *số thiết bị* và chỉ số mô tả inode.

- `-o optname`
  Đúng, nếu tùy chọn `optname` của hệ vỏ được kích hoạt. Chi tiết xin xem trên trang `man bash`.

- `-z string`
  Đúng, nếu độ dài của chuỗi `string` bằng không.

- `-n string`
  Đúng, nếu độ dài của chuỗi khác không.

- `string1 == string2`
  Đúng, nếu hai chuỗi trùng nhau. Có thể thay hai == bằng một =.

- `string1 !== string2`
  Đúng, nếu hai chuỗi không trùng nhau.

- `string1 < string2`
  Đúng, nếu chuỗi `string1`, theo từ điển, đứng trước chuỗi `string2` (đối với ngôn ngữ hiện thời).

- `string1 > string2`
  Đúng, nếu chuỗi `string1`, theo từ điển, đứng sau chuỗi `string2` (đối với ngôn ngữ hiện thời).

- `arg1 OP arg2`
  Ở đây OP là một trong các phép so sánh số học: `-eq` (bằng), `-ne` (khác, không bằng), `-lt` (nhỏ hơn), `-le` (nhỏ hơn hoặc bằng), `-gt` (lớn hơn), `-ge`

(lớn hơn hoặc bằng). Ở chỗ các tham số arg1 và arg2 có thể sử dụng các số nguyên (âm hoặc dương).

Từ các biểu thức điều kiện cơ bản này có thể xây dựng các biểu thức phức tạp theo ý muốn nhờ các phép lôgíc thông thường PHỦ ĐỊNH, VÀ (cộng) và HOẶC:

- `!(expression)`
  Phép phủ định. Đúng, nếu biểu thức **sai**.
- `expression1 -a expression2`
  Phép cộng lôgic `AND`. Đúng nếu **cả hai** biểu thức đều đúng.
- `expression1 -o expression2`
  Phép lôgíc hoặc `OR`. Đúng nếu **một trong hai** biểu thức đúng.

5.8.3 Toán tử `case`

Dạng của toán tử `case` như sau:

`case` word `in` [[(] pattern [| pattern] ...) list ;;] ... `esac`

Câu lệnh `case` đầu tiên khai triển từ `word`, và so sánh nó (word) với mỗi từ trong mẫu `pattern` theo thứ tự. Sau khi tìm thấy sự trùng nhau đầu tiên thì dừng việc so sánh lại, và thực hiện danh sách `list` các câu lệnh đứng sau mẫu đã tìm thấy. Giá trị trả lại bởi toán tử này, bằng 0, nếu không tìm thấy sự trùng nhau nào. Trong trường hợp ngược lại, trả lại giá trị mà câu lệnh cuối cùng trong danh sách `list` đưa ra. Ví dụ sử dụng toán tử `case` sau lấy từ script `/etc/rc.d/rc.sysinit` (FreeBSD-style):

```
case "$UTC" in
yes|true)
CLOCKFLAGS="$CLOCKFLAGS -u";
CLOCKDEF="$CLOCKDEF (utc)";
;;
no|false)
CLOCKFLAGS="$CLOCKFLAGS --localtime";
CLOCKDEF="$CLOCKDEF (localtime)";
;;
esac
```

Nếu biến số (`UTC`) nhận giá trị `yes` hoặc `true`, thì sẽ thực hiện cặp lệnh thứ nhất, nhận giá trị `no` hoặc `false` - cặp thứ hai.

5.8.4 Toán tử `select`

Toán tử `select` cho phép tổ chức hội thoại với người dùng. Nó có dạng sau:

`select` name [`in` word;] `do` list; `done`

Lúc đầu từ mẫu `word` hình thành một danh sách những từ tương ứng với mẫu này. Tập hợp những từ này được đưa vào *kênh thông báo lỗi tiêu chuẩn*, hơn nữa mỗi từ được đi kèm với một số thứ tự. Nếu mẫu `word` bị bỏ qua (không có trong toán tử `select`), thì sẽ đưa vào các tham biến vị trí (xem trên) theo một cách tương tự. Sau đó, dấu nhắc PS3 được đưa ra, và hệ vỏ chờ chuỗi nhập vào trên *đầu vào tiêu chuẩn*. Nếu chuỗi nhập vào có chứa số, tương ứng với một trong các số đã hiện ra, thì biến `name` sẽ được gán giá trị bằng từ đi kèm với số này. Nếu nhập vào một dòng rỗng, thì số và từ sẽ được hiện ra them một lần nữa. Nếu nhập vào bất kỳ một giá trị nào khác, thì biến `name` sẽ nhận giá trị bằng không. Chuỗi mà người dùng nhập vào, được ghi lại trong biến `REPLY`. Danh sách lệnh `list` được thực hiện với giá trị biến `name` đã chọn. Sau đây là một script nhỏ (xin hãy gõ không dấu nếu console của bạn chưa hỗ trợ việc hiển thị Tiếng Việt):

```
#!/bin/sh
echo "Bạn thích dùng OS nào?"
select var in "Linux" "Gnu Hurd" "Free BSD" "MacOSX" "Solaris"
"QNX" "Other"; do
break
done
echo "Bạn đã chọn $var"
```

Ghi đoạn trên vào một tập tin (ví dụ, `select.sh`), thay đổi để tập tin thành khả thi (ví dụ, `chmod 755 select.sh`), và chạy (`./select.sh`). Trên màn hình sẽ hiện ra câu hỏi sau:

```
Bạn thích dùng OS nào?
1) Linux     3) Free BSD  5) Solaris   7) Other
2) Gnu Hurd  4) MacOSX    6) QNX
#?
```

Hãy nhấn một trong 7 số đưa ra (từ 1 đến 7). Nếu bạn nhập 4 (nhấn cả <Enter>), thì sẽ thấy thông báo sau:

```
Bạn đã chọn MacOSX
```

5.8.5 Toán tử `for`

Toán tử `for` làm việc có khác một chút so với `for` trong các ngôn ngữ lập trình thông thường. Thay vì tăng hoặc giảm giá trị của một biến số nào đó (lên hoặc xuống) một đơn vị sau mỗi vòng lặp, thì nó gán giá trị tiếp theo trong `danh sách từ đưa sẵn` cho biến đó trong mỗi vòng lặp. Nói chung cấu trúc có dạng sau:

`for` **name** `in` **words** `do` **list** `done`

Quy luật xây dựng danh sách lệnh (`list`) giống trong toán tử `if`. Ví dụ. Script sau tạo các tập tin `fu1, fu2`, và `fu3`:

```
for a in 1 2 3 ; do
touch fu$a
done
```

Có thể gõ ba dòng này trên một dòng lệnh, kết quả thu được tương tự với script. Dạng tổng quát của toán tử `for` như sau:

`for` name [`in` word;] `do` list ; `done`

Đầu tiên cũng xảy ra sự khai triển từ `word` theo quy luật khai triển biểu thức (xem trên). Sau đó biến `name` lần lượt được gán các giá trị thu được từ sự khai triển này, và thực hiện danh sách lệnh `list` trong mỗi lần như vậy. Nếu không có "`in word`", thì danh sách lệnh `list` được thực hiện một lần cho mỗi `tham biến vị trí` đã đưa ra. Trên Linux có chương trình `seq`, tiếp nhận hai số nguyên làm tham số, và đưa ra chuỗi tất cả các số nằm giữa hai số này (cộng thêm cả chúng). Nhờ câu lệnh này có thể sử dụng `for` của `bash` làm việc như toán tử `for` trong các ngôn ngữ lập trình thông thường. Để làm được điều này chỉ cần viết vòng lặp `for` như sau:

```
for a in $( seq 1 6 ) ; do
cat fu$a
done
```

Câu lệnh (script) này đưa ra màn hình nội dung của 10 tập tin (nếu có): "`fu1`", ..., "`fu10`".

5.8.6 Toán tử `while` và `until`

Toán tử `while` làm việc tương tự như `if`, nhưng vòng lặp các câu lệnh trong `list2` chỉ thực hiện khi điều kiện còn đúng, và sẽ ngừng khi điều kiện không thỏa mãn. Cấu trúc có dạng như sau:

`while` list1 `do` list2 `done`

Ví dụ:

```
while [ -d directory ] ; do
ls -l directory >> logfile
echo -- SEPARATOR -- >> logfile
sleep 60
done
```

Chương trình (script) trên sẽ theo dõi và ghi lại nội dung của thư mục `directory` theo từng phút nếu thư mục còn tồn tại. Toán tử `until` tương tự như toán tử `while`:

`until` list1 `do` list2 `done`

Điểm khác biệt nằm ở chỗ, sử dụng giá trị phủ định của điều kiện `list1`, tức là `list2` thực hiện, nếu câu lệnh cuối cùng trong danh sách `list1` trả lại trạng thái thoát ra khác không.

5.8.7 Các hàm số

Cú pháp Hệ vỏ `bash` cho phép người dùng tạo các hàm số cho mình. Hàm số làm việc và được sử dụng giống như các câu lệnh thông thường của hệ vỏ, tức là chúng ta có thể tự tạo các câu lệnh mới. Hàm số có cấu trúc như sau:

`function` name `()` `{` list `}`

Hơn nữa từ `function` không nhất thiết phải có, `name` xác định tên của hàm (dùng để gọi hàm), còn phần thâm của hàm số tạo bởi danh sách các câu lệnh `list`, nằm giữa `{` và `}`. Các câu lệnh này sẽ được thực hiện mỗi khi tên `name` được gọi (giống như một lệnh thông thường). Cần chú ý rằng hàm có thể là đệ qui, tức là gọi hàm số ở **ngay trong** phần thân của nó. Hàm số thực hiện trong phạm vi hệ vỏ hiện thời: **không** có tiến trình mới nào được chạy khi biên dịch hàm số (khác với việc chạy script).

5.8.8 Tham số

Khi hàm số được gọi để thực hiện, các tham số của hàm sẽ trở thành các tham biến vị trí (positional parameters, xem trên) **trong thời gian** thực hiện hàm này. Chúng được đặt các tên như `$n`, trong đó `n` là số của tham số mà chúng ta muốn sử dụng. Việc đánh số bắt đầu từ 1, như vậy `$1` là tham số đầu tiên. Cũng có thể sử dụng tất cả các tham số một lúc nhờ `$*`, và đưa ra số thứ tự của tham số nhờ `$#` Tham số vị trí số 0 không thay đổi. Trong khi thực hiện nếu gặp câu lệnh nội trú `return` (trong phần thân của hàm), thì hàm số sẽ bị dừng lại và quyền điều khiển được trao cho câu lệnh đứng sau hàm. Khi thực hiện xong hàm số, các tham biến vị trí và tham biến đặc biệt `#` sẽ được trả lại các giá trị mà chúng có trước khi chạy hàm.

5.8.9 Biến nội bộ (local)

Nếu muốn tạo một tham biến địa phương, có thể sử dụng từ khóa `local`. Cú pháp đưa ra biến địa phương giống hệt các tham biến khác, chỉ có điều cần đứng sau từ khóa `local`: `local name=value`. Dưới đây là một ví dụ hàm số, thực hiện công việc của lệnh `seq` đã nhắc đến ở trên:

```
seq()
{
local I=$1;
while [ $2 != $I ]; do
{
echo -n "$I ";
I=$(( $I + 1 ))
```

```
};
done;
echo $2
}
```

Cần chú ý đến tùy chọn `-n` của `echo`, nó (tùy chọn) hủy bỏ việc tạo dòng mới. Mặc dù tùy chọn này không có nhiều ý nghĩa với mục đích chúng ta muốn ở đây, nhưng sẽ rất có ích trong các hàm số với mục đích khác. **Hàm số tính giai thừa fact** Một ví dụ khác:

```
fact()
{
if [ $1 = 0 ]; then
echo 1;
else
{
echo $(( $1 * $( fact $(( $1 -- 1 )) ) ))
};
fi
}
```

Đây là hàm số giai thừa, một ví dụ của hàm đệ qui. Hãy chú ý đến sự khai triển số học, và phép thế các câu lệnh.

5.9 Script của hệ vỏ và lệnh source

Script của hệ vỏ chỉ là các tập tin có chứa chuỗi lệnh. Tương tự hàm số script có thể được thực hiện như một câu lệnh. Cú pháp truy cập đến các tham số cũng như hàm số. Trong các trường hợp thông thường khi chạy script sẽ có một tiến trình mới được chạy. Để có thể thực hiện script ở trong bản `bash` hiện thời, cần sử dụng câu lệnh `source`, hay một dấu chấm "." (đồng nghĩa của source). Trong trường hợp này script chỉ đơn giản là một tham số của lệnh nói trên. Câu lệnh sẽ có dạng:

```
source filename [arguments]
```

Câu lệnh này đọc và thực hiện các câu lệnh có trong tập tin `filename` trong *môi trường* hiện thời, và trả lại giá trị, xác định bởi câu lệnh cuối cùng của `filename`. Nếu `filename` không chứa dấu gạch chéo, thì đường dẫn, liệt kê trong biến số PATH, sẽ được sử dụng để tìm tập tin có tên `filename`. Tập tin này không nhất thiết phải khả thi (không nhất thiết phải có bit x). Nếu trong thư mục, liệt kê trong PATH, không tìm thấy tập tin cần, thì sẽ tìm nó (tập tin) trong thư mục hiện thời. Nếu có các tham số (đưa ra arguments, xem định dạng câu lệnh ở trên), thì trong thời gian thực hiện script chúng sẽ thành các tham biến vị trí. Nếu không có tham số, thì tham biến vị trí không thay đổi. Giá trị (trạng thái), mà lệnh `source` trả lại, trùng với giá trị, trả lại bởi câu lệnh cuối cùng trong script. Nếu không câu lệnh nào được thực hiện, hoặc không tìm thấy tập tin `filename`, thì trạng thái thoát bằng 0.

5.10 Câu lệnh sh

Bạn luôn luôn có thể chạy một bản sao của hệ vỏ `bash` nhờ câu lệnh `bash` hay `sh`. Khi này có thể bắt bản sao này chạy một script nào đó, nếu đưa tên của script như một tham số cho lệnh `bash`. Như vậy, để thực hiện script `myscript` cần đưa câu lệnh "`sh myscript`". Nếu xem nội dung của một tập tin script nào đó (những tập tin như vậy có rất nhiều trên hệ thống), bạn sẽ thấy dòng đầu tiên có dạng sau: `#!/bin/sh`. Điều này có nghĩa là, khi chúng ta gọi script để thực hiện như một lệnh thông thường, thì `/bin/sh` sẽ giúp chúng ta "thu xếp" mọi thứ. Có thể thay thế dòng này bởi liên kết đến bất kỳ một chương trình nào, mà sẽ đọc tập tin và thực hiện các câu lệnh tương ứng. Ví dụ, script trên ngôn ngữ Perl bắt đầu bởi dòng có dạng `!/bin/perl`. Một chú ý khác là ký tự `#` dùng để viết lời chú thích trong script. Tất cả những gì đứng sau ký tự này đến cuối dòng sẽ được coi là chú thích và bị `bash` bỏ qua (tức là hệ vỏ sẽ không xem dòng này là câu lệnh). Nếu bạn muốn kiểm chứng lại tác dụng của ký tự này, thì hãy nhập vào dòng lệnh một câu lệnh bất kỳ, và đặt trước nó (câu lệnh) ký tự `#`, ví dụ "`# ls`", bạn sẽ thấy rằng hệ vỏ bỏ qua câu lệnh này.

Chúng ta sẽ dừng bài học ngắn gọn về `bash` tại đây. Tất nhiên, còn rất nhiều vấn đề quan trọng cần xem xét nhưng nằm ngoài phạm vi của bài học, ví dụ, quản lý tiến trình, lịch sử câu lệnh, mô tả về thư viện readline, tín hiệu, v.v... Các bạn sẽ tìm thấy thông tin cần thiết trong các cuốn hướng dẫn khác hoặc trên trang `man bash`.

Chương 6

Sử dụng Midnight Commander

Mọi thứ đều đã được viết ra – các nhà lập trình Nga.

Trong chương này chúng ta sẽ học cách sử dụng chương trình Midnight Commander, một trình quản lý tập tin mạnh. Sau khi đọc xong chương này bạn đọc sẽ có thể sử dụng các tổ hợp phím cũng như giao diện của Midnight Commander để thực hiện rất nhiều công việc có liên quan đến hệ thống tập tin từ nhỏ bé đến nặng nhọc. Đầu tiên chúng ta hãy xem xét cách cài đặt chương trình này...

6.1 Cài đặt chương trình Midnight Commander

Mặc dù để điều khiển hệ thống tập tin nói chung và để làm việc với các tập tin nói riêng có thể sử dụng các câu lệnh của hệ điều hành, như `pwd`, `ls`, `cd`, `mv`, `mkdir`, `rmdir`, `cp`, `rm`, `cat`, `more` v.v... nhưng sẽ thuận tiện hơn khi sử dụng chương trình Midnight Commander.

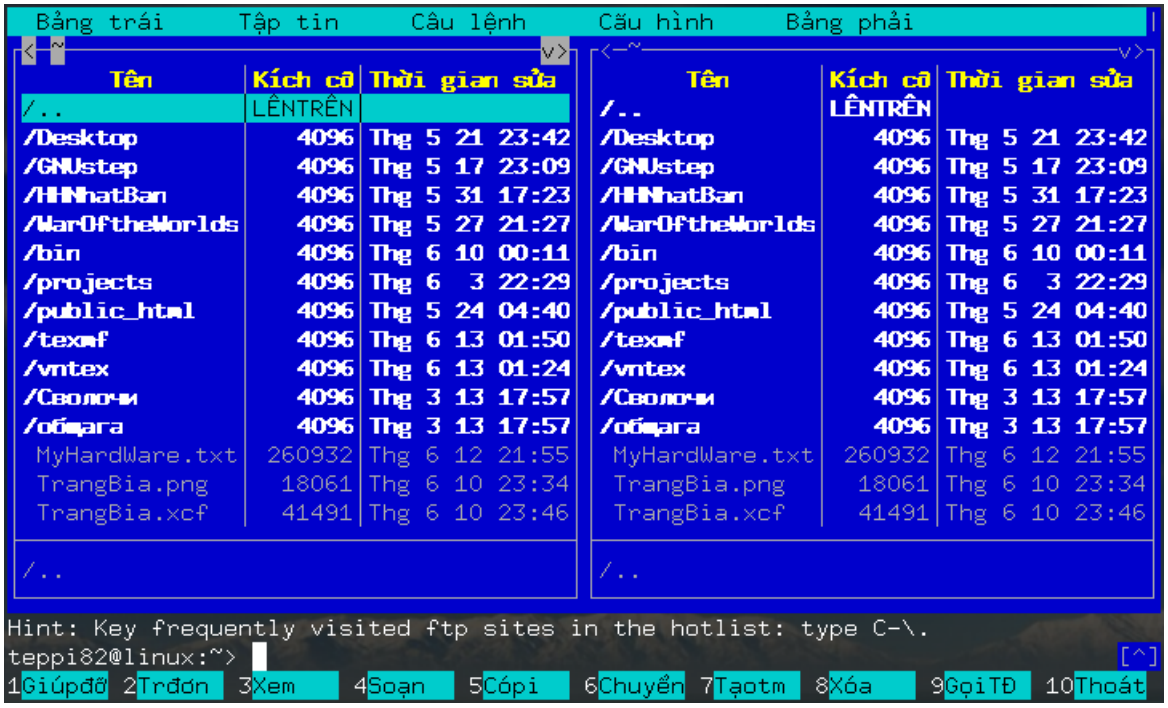
Midnight Commander (hay thường rút gọn thành `mc`) là chương trình cho phép xem cấu trúc cây thư mục và thực hiện những thao tác điều khiển hệ thống tập tin. Nói cách khác, đây là trình quản lý tập tin (File Manager). Nếu như bạn đọc có kinh nghiệm làm việc với Norton Commander (nc) trong MS-DOS hay với FAR trong Windows, thì sẽ làm việc với `mc` một cách dễ dàng. Bởi vì thậm chí những tổ hợp "phím nóng" chính của chúng cũng trùng nhau. Trong trường hợp này, để có thể làm việc với Midnight Commander bạn đọc chỉ cần xem nhanh những nội dung phía dưới. Tác giả xin có lời khuyên đối với những ai còn xa lạ với NC hay FAR (nếu như có?): hãy chú ý đọc và thực hành chăm chỉ, vì Midnight Commander sẽ giúp đỡ rất nhiều trong khi làm việc với hệ điều hành.

Ghi chú:

1. Kiến thức trong chương này được viết để sử dụng cho phiên bản 4.6.1-pre3[1], mặc dùng có thể áp dụng cho những phiên bản khác.

2. Kiến thức đưa ra chỉ áp dụng được hoàn toàn trong trường hợp chương trình chạy từ kênh giao tác (console), hay còn gọi là giao diện text. Khi làm việc với chương trình qua trình giả lập (emulator) của terminal trong giao diện đồ họa, ví dụ `xterm`, `rxvt`, v.v... thì một số mô tả hoạt động của chương trình sẽ không còn

[1] trong bản gốc tác giả Kostromin dùng phiên bản 4.5.30, theo ý kiến người dịch là khá cũ và không còn gặp trong những bản phân phối Linux mới.



Hình 6.1: Giao diện tiếng Việt của Midnight Commander

chính xác nữa, vì việc nhấn phím đã bị vỏ đồ họa chiếm lấy. Sự không tương ứng như vậy thường gặp ở những chỗ nói về phím "nóng".

Trong phần lớn các bản phân phối chương trình Midnight Commander không được tự động cài đặt cùng với hệ thống. Nhưng các gói (rpm, deb, tgz ...) của Midnight Commander sẽ có trên đĩa, và việc cài đặt từ các gói này là không khó khăn gì. Và bởi vì chương trình này sẽ làm cho bạn đọc "dễ thở" hơn, tác giả rất muốn rằng chương trình sẽ được cài đặt ngay sau khi cài xong hệ điều hành.[2]

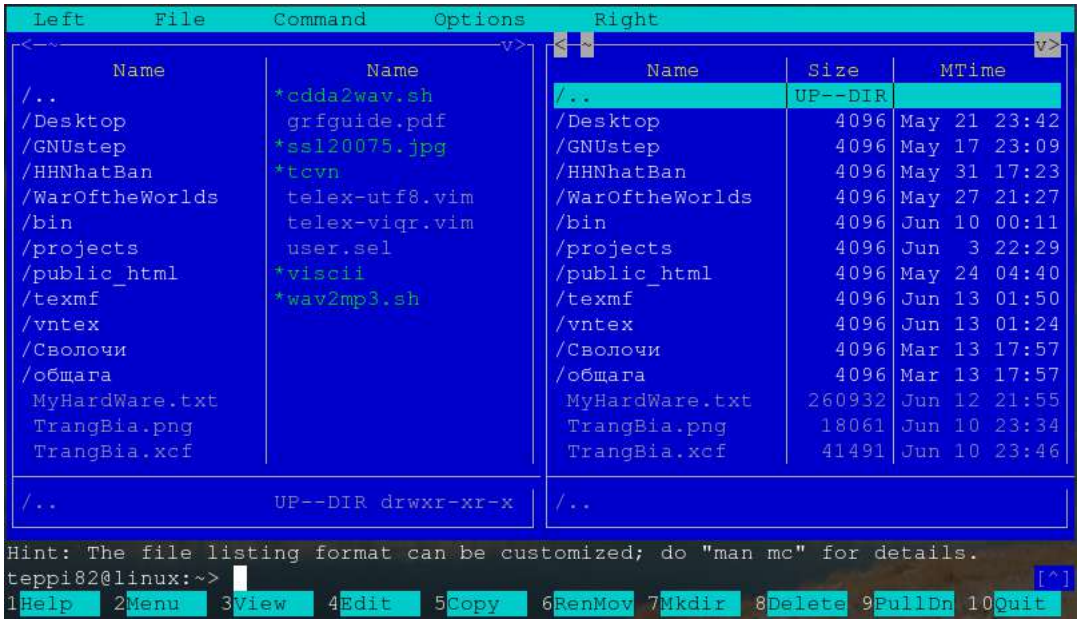
6.2 Vẻ ngoài của màn hình Midnight Commander

Để khởi động Midnight Commander, cần gõ vào dòng lệnh shell câu lệnh `mc` và nhấn <Enter>. Nếu ứng dụng không chạy, thì cần tìm xem tập tin chương trình `mc` nằm ở đâu (có thể dùng câu lệnh `find / -name mc -type f`), sau đó gõ vào dòng lệnh đường dẫn đầy đủ tới tập tin đó, ví dụ, trên máy tác giả là `/usr/bin/mc`. Sau khi chạy chương trình, bạn đọc sẽ thấy màn hình màu da trời làm chúng ta nhớ đến màn hình chương trình Norton Commander cho MS-DOS hay chương trình FAR cho Windows như trong hình 6.2.

Gần như toàn bộ không gian màn hình khi làm việc với Midnight Commander bị chiếm bởi hai "bảng"[3] hiển thị danh sách tập tin của hai thư mục. Ở phía trên hai bảng này là trình đơn (thực đơn). Có thể chuyển đến trình đơn để chọn các lệnh có trong đó bằng phím <F9> hoặc nhờ phím chuột (nếu như sau khi khởi

[2] Ngoài ra người dịch cuốn sách này cũng đã dịch giao diện của Midnight Commander ra tiếng Việt. Vì thế nếu muốn bạn có thể sử dụng giao diện tiếng mẹ đẻ của Midnight Commander.

[3] panel



Hình 6.2: Vẻ ngoài của màn hình Midnight Commander

động `mc` bạn đọc không nhìn thấy dòng trình đơn đâu, thì cũng đừng buồn, vì có hiển thị trình đơn hay không được xác định bởi cấu hình chương trình).

Dòng dưới cùng là dãy các nút, mỗi nút tương ứng với một phím chức năng <F1> – <F10>. Có thể coi dòng này là lời mách nước về cách sử dụng những phím chức năng đã nói, và còn có thể chạy trực tiếp các câu lệnh tương ứng bằng cách nhấn chuột vào các nút này. Việc hiển thị các nút có thể tắt đi, nếu như bạn đọc muốn tiết kiệm không gian màn hình (cách làm sẽ có sau này khi chúng ta nói về cấu hình chương trình). Sự tiết kiệm có hai lý do. Thứ nhất, bạn đọc sẽ nhanh chóng nhớ được công dụng của 10 phím này, và lời mách nước sẽ trở thành không cần thiết (và việc nhấn chuột lên các nút không phải lúc nào cũng thuận tiện). Thứ hai, thậm chí nếu bạn đọc không nhớ phải dùng phím nào để thực hiện công việc mong muốn, thì vẫn có thể sử dụng trình đơn **File** (**Tập tin**) trong trình đơn chính của chương trình (chỉ cần nhớ rằng, phím để chuyển vào trình đơn chính là <F9>). Qua trình đơn **File** (**Tập tin**) có thể thực hiện bất kỳ thao tác nào mà thông thường phải nhờ các phím chức năng, ngoại trừ <F1> và <F9>.

Dòng thứ hai từ dưới lên là dòng lệnh của chương trình Midnight Commander (hay chính xác hơn là dòng lệnh của shell hiện thời). Ở đây có thể nhập và thực hiện bất kỳ câu lệnh nào của hệ thống. Ở phía trên dòng này (nhưng phía dưới các bảng) có thể hiển thị "những lời khuyên có ích" (hint4s). Cũng có thể bỏ đi dòng lời khuyên này khi điều chỉnh cấu hình của chương trình.

Mỗi bảng gồm phần đầu, danh sách tập tin của một thư mục nào đó và dòng trạng thái nhỏ (mini-status, có thể không hiển thị nếu đặt trong cấu hình chương trình). Trong phần đầu của mỗi bảng là đường dẫn đầy đủ đến thư mục có nội dung được hiển thị, và đồng thời còn có ba nút – "<", "v" và ">" sử dụng để điều khiển chương trình bằng chuột (những nút này không làm việc nếu như bạn đọc chạy `mc` trong trình giả tạo (emulator) terminal). Trên dòng trạng thái nhỏ có hiển thị một vài dữ liệu về tập tin hay thư mục đang được thanh chiếu sáng chỉ đến (ví dụ, kích thước tập tin và quyền truy cập).

Chỉ một trong hai bảng là *hiện thời* (*hoạt động*). Bảng hiện thời có thanh chiếu sáng tên thư mục ở phần đầu và thanh chiếu sáng một trong những dòng của bảng đó. Tương tự, trong shell đã chạy chương trình Midnight Commander, thư mục hiện thời là thư mục được hiển thị trong bảng hoạt động. Hầu hết các thao tác được thực hiện trong thư mục này. Các thao tác như sao chép (<F5>) hay di chuyển (<F6>) tập tin sử dụng thư mục được hiển thị trong bảng thứ hai làm thư mục đích đến (sẽ sao chép hay di chuyển đến thư mục này).

Trong bảng hoạt động một dòng được chiếu sáng. Thanh chiếu sáng có thể di chuyển nhờ các phím điều khiển việc di chuyển. Chương trình xem tập tin nội trú, chương trình xem lời mách nước và chương trình xem thư mục sử dụng cùng một mã chương trình để điều khiển việc di chuyển. Vì thế việc di chuyển sử dụng một bộ các tổ hợp phím (nhưng trong mỗi chương trình con có các tổ hợp phím chỉ áp dụng trong nội bộ mà thôi). Xin đưa ra một bảng ngắn gọn liệt kê các tổ hợp phím dùng chung để điều khiển việc di chuyển.

Bảng 6.1: Các tổ hợp phím di chuyển dùng chung

Phím	**Di chuyển thực hiện**
<↑> hoặc <Ctrl>+<P>	Di chuyển trở lại (lên trên) một dòng
<↓> hoặc <Ctrl>+<N>	Di chuyển về phía trước (xuống dưới) một dòng
<Page Up> hoặc <Alt>+<V>	Quay lại một trang
<Page Down> hoặc <Ctrl>+<V>	Tiến về trước một trang
<Home>	Quay về dòng đầu
<End>	Chuyển đến dòng cuối cùng

6.3 Trợ giúp

Khi làm việc với chương trình Midnight Commander, có thể xem trợ giúp vào bất kỳ lúc nào nhờ phím <F1>. Trợ giúp được tổ chức dưới dạng siêu văn bản, tức là trong văn bản có cả những liên kết đến những phần khác. Những liên kết đó được đánh dấu bởi nền màu xanh nhạt. Liên kết được chọn hiện thời sẽ có nền màu xanh đậm.

Để di chuyển trong cửa sổ xem trợ giúp có thể sử dụng những phím mũi tên hoặc chuột. Ngoài những tổ hợp phím di chuyển nói chung trong bảng 6.1, chương trình xem trợ giúp còn chấp nhận những tổ hợp phím sử dụng trong chương trình con dùng để xem tập tin:

Bảng 6.2: Di chuyển trong trình xem tập tin

Phím	**Di chuyển thực hiện**
<B> hoặc <Ctrl>+<B> hoặc <Ctrl>+<H> hoặc <Backspace> hoặc <Delete>	Lùi lại một trang
<Dấu cách>	Tiến tới một trang
<U> (<D>)	Lùi lại (tiến tới) nửa trang
<G> (<Shift>+<G>)	Đi tới đầu (cuối) danh sách

Ngoài những tổ hợp phím đã chỉ ra còn có thể sử dụng những tổ hợp chỉ làm việc khi xem trợ giúp (chúng được liệt kê trong bảng 6.3).

Bảng 6.3: Di chuyển khi xem trợ giúp

Phím	**Di chuyển thực hiện**
<Tab>	Đi tới liên kết tiếp theo
<Alt>+<Tab>	Quay lại liên kết trước
<↓>	Đi tới liên kết tiếp theo hoặc kéo lên một dòng
<↑>	Quay lại liên kết trước hoặc kéo xuống một dòng
<→> hoặc <Enter>	Mở trang mà liên kết hiện thời chỉ tới
<←> hoặc <L>	Mở trang trợ giúp vừa xem trước trang hiện thời
<F1>	Trợ giúp sử dụng của bản thân trợ giúp
<N>	Chuyển tới phần tiếp theo của trợ giúp
<P>	Chuyển tới phần nằm trước của trợ giúp
<C>	Chuyển tới mục lục của trợ giúp
<F10>, <Esc>	Thoát khỏi trợ giúp

Bạn có thể sử dụng phím trắng (space) để chuyển tới trang trợ giúp tiếp theo và phím <B> để quay lại một trang. Chương trình ghi nhớ thứ tự di chuyển theo liên kết và cho phép trở lại phần đã xem trước đó bằng phím <L>.

Nếu như có hỗ trợ chuột (xem phần 6.4), thì có thể sử dụng chuột để di chuyển. Nhấn chuột trái lên liên kết để chuyển tới văn bản mà liên kết này chỉ tới. Chuột phải sử dụng để quay lại phần đã xem trước đó.

6.4 Sử dụng chuột

Chương trình Midnight Commander có hỗ trợ chuột. Tính năng này được thực hiện nếu có chạy driver gpm không phụ thuộc vào nơi người dùng làm việc là trên kênh giao tác Linux hay chạy Midnight Commander trên xterm (hoặc thậm chí sử dụng kết nối từ xa qua telnet, rlogin hay ssh).

Bằng cách nhấn nút chuột trái sẽ có thể di chuyển dòng chiếu sáng lên bất kỳ tập tin nào trong các bảng. Để đánh dấu (chọn) tập tin nào đó, chỉ cần nhấn nút chuột phải lên tên của tập tin đó, khi này tên tập tin sẽ có màu khác (theo mặc định là màu vàng). Để bỏ đánh dấu thì cũng chỉ cần sử dụng nút chuột phải đó.

Nhấn kép chuột trái lên tên tập tin để thực hiện tập tin (nếu đây là một chương trình), hoặc chạy chương trình có khả năng và đã được gán để đọc tập tin này. Ví dụ chương trình `xv` được gán để mở các tập tin hình ảnh `*.jpg`, thì khi nhấn kép chuột trái lên tập tin `screenshot.jpg`, chương trình `xv` sẽ cho chúng ta thấy tập tin `screenshot.jpg` trông ra sao.

Nhấn chuột (bất kỳ nút nào) lên các nút chức năng (các nút **F1-F10** ở dưới cùng) cũng đồng thời chạy chương trình tương ứng với nút đó. Nhấn chuột (bất kỳ nút nào) lên trình đơn ở trên cùng sẽ mở ra (nói đúng hơn là mở xuống dưới) trình đơn con của nó.

Nếu nhấn chuột lên khung trên cùng của bảng có một danh sách các tập tin rất dài, thì sẽ thực hiện di chuyển dài 1 cột tập tin về phía đầu danh sách. Nhấn

chuột lên khung nằm dưới của bảng, thì tương ứng sẽ thực hiện di chuyển dài 1 cột tập tin về phía cuối danh sách. Phương pháp di chuyển này cũng làm việc khi xem trợ giúp và xem danh sách **Cây thư mục**

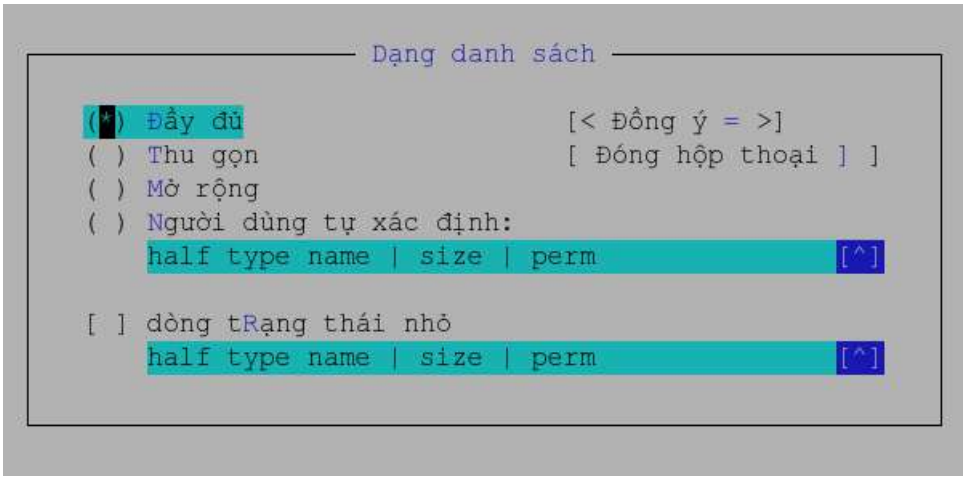
Nếu chạy `mc` với hỗ trợ chuột thì người dùng có thể thực hiện các thao tác sao chép và dán văn bản khi giữ phím <Shift>. Để làm được điều này, bạn cần nhấn và giữ phím <Shift>, chọn đoạn văn bản cần thiết bằng cách kéo chuột, sau đó thả phím <Shift> ra, đưa con trỏ đến nơi cần dán, rồi lại nhấn và giữ phím <Shift> trong khi nhấn chuột phải. Cần chú ý rằng tính năng này không làm việc trong các trình giả lập terminal như `xterm`.

6.5 Điều khiển các bảng

Các bảng của Midnight Commander thông thường hiển thị những gì có trong thư mục của hệ thống tập tin (vì thế thường được gọi là bảng thư mục). Tuy nhiên có thể hiển thị những thông tin khác trên bảng. Trong phần này sẽ nói đến cách thay đổi dạng của bảng hay cách hiển thị thông tin trên bảng.

6.5.1 Dạng danh sách tập tin

Dạng bảng trên đó hiển thị danh sách tập tin và thư mục con có thể thay đổi qua các câu lệnh của thực đơn mở ra khi chọn Bảng trái (Left) và Bảng phải (Right). Nếu bạn đọc muốn thay đổi dạng hiển thị danh sách tập tin, thì có thể sử dụng câu lệnh **Dạng danh sách...** (**Listing mode...**) của bảng (trái hoặc phải) tương ứng. Có 4 khả năng để chọn: **Đầy đủ** (Full), **Thu gọn** (Brief), **Mở rộng** (Long) và **Người dùng tự xác định** (User).



Hình 6.3: Hộp thoại chọn định dạng hiển thị

- Định dạng **Đầy đủ** (Full) hiển thị tên tập tin, kích thước của nó và thời gian sửa đổi gần nhất.
- Định dạng **Thu gọn** (Brief) chỉ hiện thị tên tập tin, do đó trên mỗi bảng có hai cột và hiển thị được số tập tin nhiều gấp đôi.

- Định dạng **Mở rộng** (Long) hiển thị tập tin như khi thực hiện câu lệnh `ls -l`. Với định dạng này một bảng chiếm hết màn hình.
- Nếu chọn định dạng **Người dùng tự xác định** (User), thì người dùng cần đưa ra dạng hiển thị tự chọn của mình.

Khi tự đưa ra định dạng, thì đầu tiên cần chỉ ra kích thước của bảng: "half" (một nửa màn hình) hoặc "full" (toàn màn hình). Sau kích thước bảng có thể chỉ ra là trên bảng phải có hai cột bằng cách thêm số 2 vào dòng định dạng. Tiếp theo cần liệt kê tên những vùng hiển thị cùng với chiều rộng của vùng. Có thể sử dụng những tên vùng sau:

- name – tên tập tin.
- size – kích thước tập tin.
- bsize – kích thước ở dạng khác, khi đó chỉ đưa ra kích thước tập tin, còn đối với thư mục con thì chỉ đưa ra dòng chữ "SUB-DIR" hoặc "UP–DIR".
- type – hiển thị dạng tập tin (một ký tự). Ký tự này có thể là một trong những giá trị mà câu lệnh `ls -F` đưa ra:

  * (asterisk) – cho tập tin chương trình.

  / (slash) – cho thư mục.

  @ (at-sign) – cho liên kết (links).

  = (giấu bằng) – cho các sockets.

  - (gạch ngang) – cho các thiết bị trao đổi theo byte.

  + (dấu cộng) – cho các thiết bị trao đổi theo block.

  | (pipe, ống) – cho các tập tin dạng FIFO.

  ~ (dấu sóng) – cho các liên kết tượng trưng đến thư mục.

  ! (dấu chấm than) – cho các liên kết tượng trưng đã hỏng (stalled) (liên kết chỉ đến tập tin không còn nữa).
- mtime – thời gian sửa đổi tập tin cuối cùng.
- atime – thời gian truy cập đến tập tin lần cuối.
- ctime – thời gian tạo tập tin.
- perm – dòng chỉ ra quyền truy cập đến tập tin.
- mode – quyền truy cập ở dạng số 8bit.
- nlink – số liên kết đến tập tin.
- ngid – chỉ số xác định của nhóm (GID).
- nuid – chỉ số xác định của người dùng (UID).
- owner – chủ sở hữu tập tin.

- group – nhóm sở hữu tập tin.
- inode – chỉ mục inode của tập tin.

Đồng thời còn có thể sử dụng những tên vùng sau để tổ chức việc hiển thị thông tin ra màn hình:

- space – chèn khoảng trắng.
- mark – chèn dấu sao * (asterisk) nếu tập tin được chọn, hoặc khoảng trắng – nếu ngược lại.
- | – chèn đường thẳng đứng.

Để có thể xác định chính xác chiều rộng của một vùng, cần thêm dấu hai chấm ':', sau đó chỉ ra số vị trí (ký tự) cần giữ cho vùng này. Nếu sau số vị trị có đặt dấu '+', thì số đó sẽ được hiểu là chiều rộng nhỏ nhất của vùng, và nếu màn hình cho phép thì vùng sẽ được mở rộng.

Ví dụ, định dạng **Đầy đủ** (Full) thực chất được xác định bởi dòng:

```
half type,name,|,size,|,mtime
```

còn định dạng **Mở rộng** (Long) thì xác định bởi:

```
full perm, space, nlink, space, owner, space, group, space, size, space, mtime, space, name
```

Dưới đây là ví dụ dạng hiển thị do người dùng đưa ra:

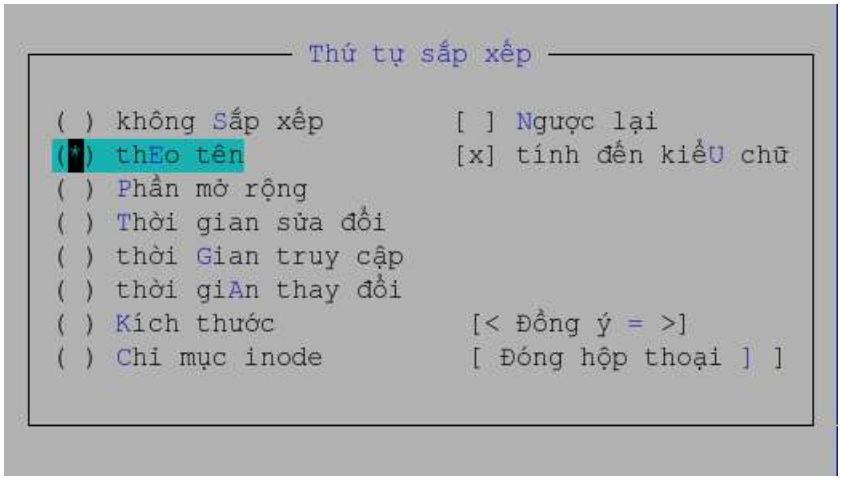
```
half name,|,size:7,|,type,mode:3
```

Hiển thị danh sách tập tin trên mỗi bảng còn có thể sắp xếp theo một trong 8 cách:

- Theo tên
- Theo phần mở rộng
- Theo kích thước tập tin
- Theo thời gian sửa đổi
- Theo thời gian truy cập lần cuối
- Theo chỉ mục inode
- Không sắp xếp.

Có thể chọn cách sắp xếp bằng cách chọn câu lệnh **Thứ tự sắp xếp** (Sort order...) trong trình đơn tương ứng của mỗi bảng. Khi đó sẽ hiện ra một hộp thoại (hình 6.4) ngoài việc cho phép chọn cách sắp xếp còn cho phép chọn sắp xếp theo thứ tự ngược lại (đánh dấu tuỳ chọn **Ngược lại** (Reverse)), hay sắp xếp có tính đến kiểu chữ thường chữ hoa hay không (tùy chọn **Tính đến kiểu chữ** (case sensitive)).

Theo mặc định các thư mục con được hiển thị ở đầu danh sách, nhưng cũng có thể thay đổi nếu đánh dấu tùy chọn **Trộn lẫn tất cả tập tin** ("Mix all files") của câu lệnh **Cấu hình...** (Configuration...) thực đơn **Cấu hình** (Option). Người dùng cũng có thể chọn chỉ hiển thị trên bảng những tập tin tương ứng với một



Hình 6.4: Hộp thoại sắp xếp

mẫu nào đó. Câu lệnh **Lọc tập tin...** (Filter...) trong thực đơn của mỗi bảng cho phép đưa ra những mẫu mà tên tập tin sẽ hiển thị tương ứng với nó (ví dụ dùng mẫu "*.tar.gz" để **chỉ** hiển thị những tập tin nén `tar.gz`). Tên của thư mục con và đường dẫn đến thư mục con luôn luôn được hiển thị không phụ thuộc vào mẫu đưa ra. Trong thực đơn của mỗi bảng còn có câu lệnh **Quét lại** (Rescan) (tương đương với câu lệnh Cập nhật (Refresh) trong các chương trình khác). Câu lệnh **Quét lại** (phím nóng <Ctrl>+<R>) cập nhật lại danh sách tập tin hiển thị trên bảng. Điều này có ích khi những tiến trình khác tạo hay xóa các tập tin.

6.5.2 Những chế độ hiển thị khác

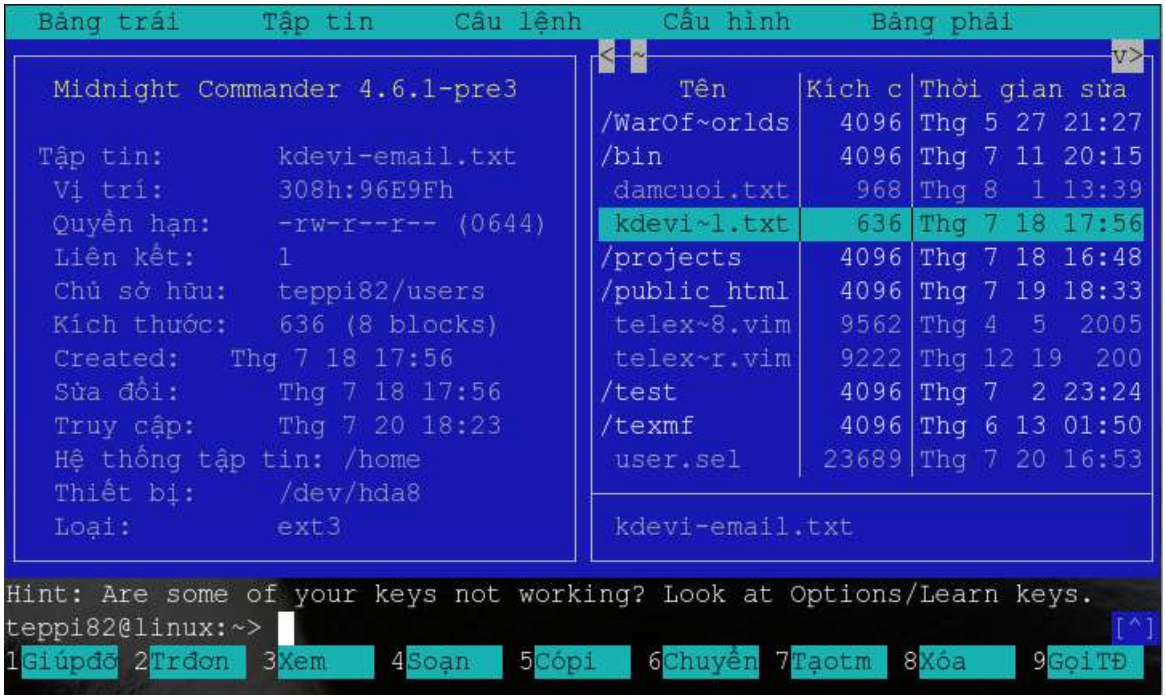
Ngoài việc đưa ra định dạng hiển thị danh sách tập tin trên bảng, còn có thể đưa bất kỳ bảng nào vào một trong những chế độ sau:

- Chế độ **Thông tin** (Info). Trong chế độ này (hình 6.5) trên bảng đưa ra thông tin về tập tin được chiếu sáng (được chọn) trên bảng bên cạnh, về hệ thống tập tin hiện thời (dạng, kích thước chỗ trống và tổng số chỉ mục inode còn trống).

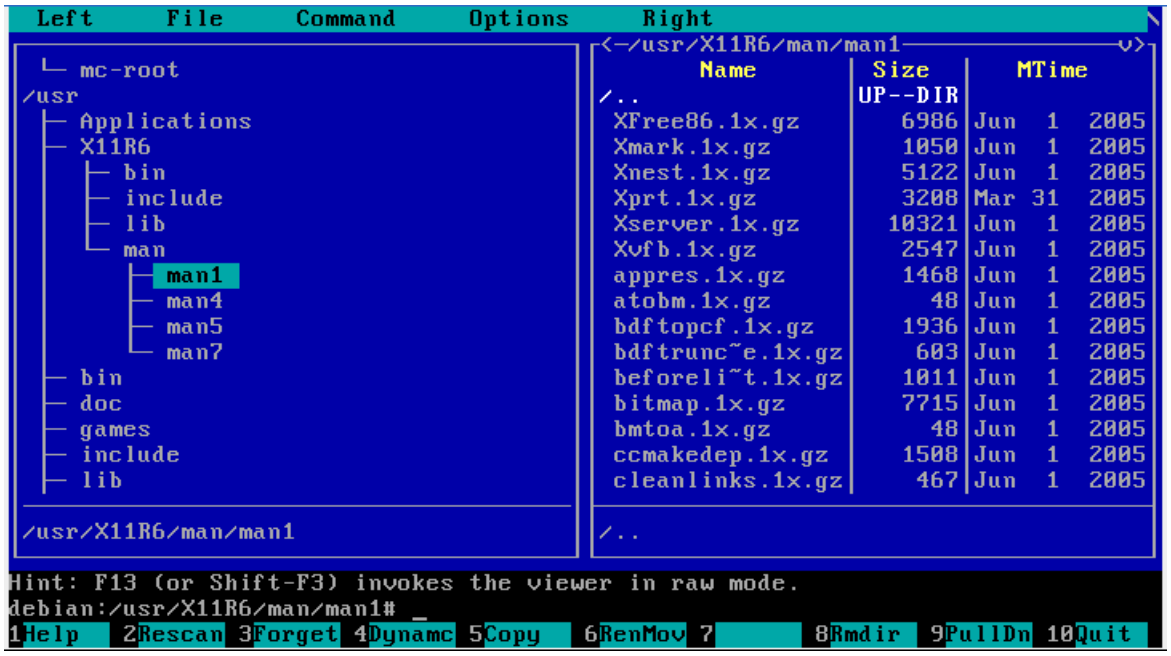
- Chế độ **Cây thư mục** (Tree). Trong chế độ này trên một bảng hiển thị cây thư mục ở dạng đồ hoạ (hình 6.6). Chế độ này tương tự như khi người dùng chọn câu lệnh **Cây thư mục** (Directory Tree) từ thực đơn **Câu lệnh** (Command), như câu lệnh sau hiển thị cây thư mục ở một cửa sổ riêng.

- Chế độ **Xem nhanh** (Quick View). Trong chế độ này bảng sẽ hiển thị nội dung của tập tin được chiếu sáng (được chọn) trên bảng bên cạnh. Ví dụ trên hình 6.7 là khi dùng chế độ này để xem nhanh nội dung tập tin `HISTORY` của gói chương trình mediawiki.

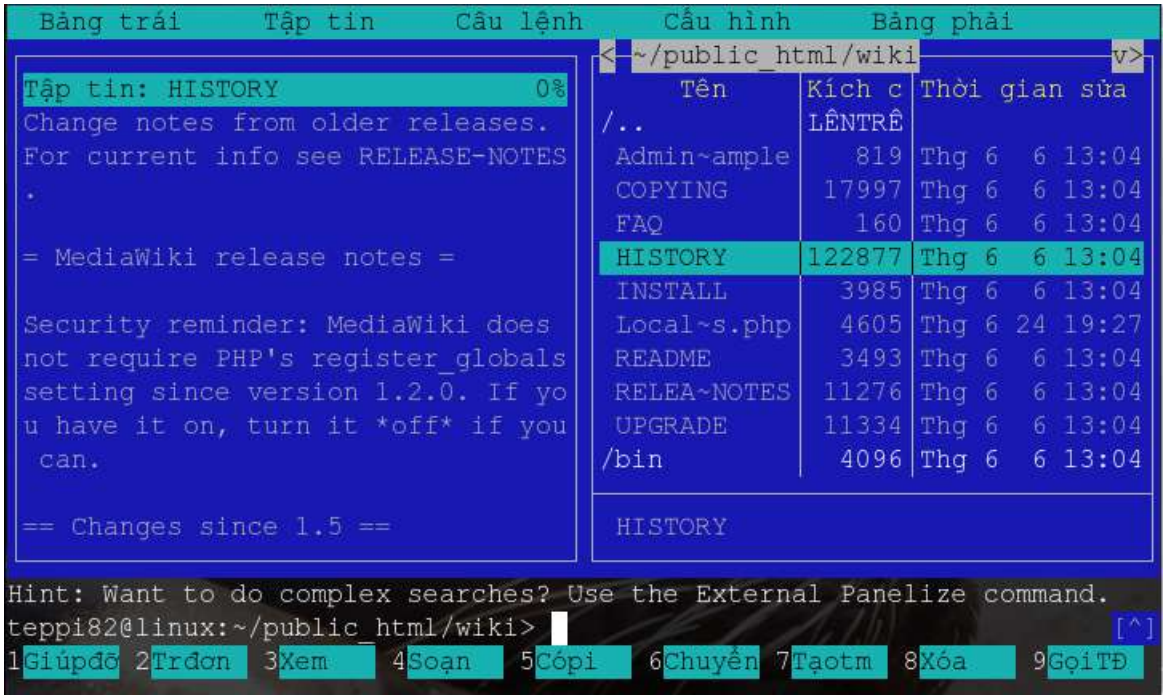
  Để ra bảng xem nhanh nội dung tập tin sử dụng chương trình xem tập tin có sẵn trong `mc`, vì thế nếu dùng phím <Tab> để chuyển sang bảng xem nhanh, thì người dùng có thể sử dụng mọi câu lệnh điều khiển việc xem, ví dụ, những phím liệt kê trong bảng 6.1 và bảng 6.2.



Hình 6.5: Chế độ thông tin



Hình 6.6: Chế độ cây thư mục



Hình 6.7: Chế độ xem nhanh

- Chế độ **Kết nối FTP...** (FTP link...) và **Kết nối Shell...** (Shell link...). Hai chế độ này chỉ khác ở chỗ sử dụng để hiển thị danh sách thư mục nằm trên máy ở xa. Còn lại mọi thứ kể cả định dạng hiển thị thông tin đều tương tự như những gì sử dụng cho các thư mục nội bộ. Nếu người dùng muốn biết thêm về cách sử dụng những chế độ này, xin hãy xem trợ giúp của `mc`.

6.5.3 Các tổ hợp phím điều khiển bảng

Để điều khiển chế độ làm việc của bảng có thể sử dụng các câu lệnh của trình đơn đã nói tới ở trên, nhưng sẽ thuận tiện hơn nếu sử dụng các tổ hợp phím điều khiển.

- <Tab> hoặc <Ctrl>+<i>. Thay đổi bảng hiện thời (hoạt động). Dòng chiếu sáng sẽ chuyển từ bảng đang là hiện thời sang bảng khác và như vậy bảng sau sẽ trở thành hiên thời.

- <Alt>+<G> hoặc <Alt>+<R> hoặc <Alt>+<J>. Sử dụng để di chuyển dòng chiếu sáng tương ứng lên tập tin trên đầu, nằm giữa hoặc dưới cùng trong số những tập tin đang hiển thị (đang thấy) trên bảng.

- <Alt>+<T>. Chuyển đổi vòng quanh giữa các định dạng hiển thị danh sách tập tin của thư mục hiện thời. Nhờ tổ hợp phím này có thể chuyển đổi nhanh chóng từ chế độ hiển thị **Mở rộng** (long) sang **Thu gọn** hay chế độ do người dùng xác định.

- <Ctrl>+<\>. Hiển thị danh sách thư mục thường dùng và chuyển tới thư mục lựa chọn.

- <Home> hoặc <Alt>+<‘<’>. Chuyển dòng chiếu sáng tới vị trí đầu tiên của danh sách tập tin.
- <End> hoặc <Alt>+<‘>’>. Chuyển dòng chiếu sáng tới vị trí cuối cùng của danh sách tập tin.
- <Alt>+<O>. Nếu trong bảng hiện thời tên thư mục được chiếu sáng, còn trên bảng còn lại hiển thị danh sách tập tin, thì bảng thứ hai sẽ chuyển vào chế độ hiển thị các tập tin của thư mục được chiếu sáng. Nếu trong bảng hiện thời dòng chiếu sáng là tên tập tin, thì trên bảng thứ hai sẽ hiển thị nội dung của thư mục mẹ của tập tin đó.
- <Ctrl>+<PageUp>, <Ctrl>+<PageDown>. Chỉ khi `mc` được chạy dưới kênh giao tác (console) của Linux: thực hiện tương ứng việc chuyển (chdir) tới thư mục mẹ (`..`) hoặc tới thư mục đang được chiếu sáng.
- <Alt>+<Y>. Chuyển tới thư mục ngay trước trong số những thư mục đã xem. Tương đương với việc nhấn chuột lên ký tự < ở góc trên của bảng.
- <Alt>+<U>. Chuyển tới thư mục ngay sau trong số những thư mục đã xem. Tương đương với việc nhấn chuột lên ký tự >.