TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRUNG TÂM TÍNH TOÁN HIỆU NĂNG CAO

KHOA CÔNG NGHỆ THÔNG TIN

NGUYỄN THANH THUỶ (CHỦ BIÊN)
LÊ QUÂN
NGÔ HỒNG SƠN
TRƯƠNG DIỆU LINH

Quản trị hệ thống Linux

NHÀ XUẤT BẢN KHOA HỌC VÀ KỸ THUẬT

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRUNG TÂM TÍNH TOÁN HIỆU NĂNG CAO

KHOA CÔNG NGHỆ THÔNG TIN

NGUYỄN THANH THUỶ (Chủ biên)

LÊ QUÂN, NGUYỄN HỒNG SƠN, TRƯƠNG DIỆU LINH

QUẢN TRỊ HỆ THỐNG LINUX

(In lần thứ 2, có bổ sung và sửa chữa)

NHÀ XUẤT BẢN KHOA HỌC VÀ KỸ THUẬT

HÀ NỘI - 2005

Lời nói đầu

Trong chương trình đào tạo kỹ sư, cử nhân chuyên ngành về Công nghệ Thông tin (CNTT) và Tin học các kiến thức cơ sở về hệ điều hành có vai trò rất quan trọng. Để có những hiểu biết kỹ lưỡng các đặc trưng cơ bản và các nguyên lý xây dựng hệ điều hành cho máy tính điện tử, các sinh viên CNTT phải tìm tòi sáng tạo những bí ẩn bên trong các hệ điều hành đã có trên thị trường. Bên cạnh các hệ điều hành truyền thống như Windows/WindowNT và UNIX, Linux được đánh giá là một hệ điều hành có nhiều triển vọng.

Linux là một hệ điều hành họ UNIX miễn phí dùng cho máy tính cá nhân đang được sử dụng rộng rãi hiện nay. Được viết vào năm 1991 bởi Linus Tovard, hệ điều hành Linux đã thu được những thành công nhất định. Hiện nay, Linux ngày càng phát triển, được đánh giá cao và thu hút nhiều sự quan tâm của các nhà tin học.

Là một hệ điều hành đa nhiệm, đa người dùng, Linux có thể chạy được trên nhiều nền phần cứng khác nhau. Linux hỗ trợ chuẩn POSIX, tức là hoàn toàn tương thích với các hệ UNIX khác. Với tính năng ổn định và mềm dẻo, Linux đang dần được sử dụng nhiều trên các máy chủ cũng như máy trạm trong các mạng máy tính. Linux còn cho phép dễ dàng thực hiện việc tích hợp nó với các hệ điều hành khác trong một mạng máy tính như Windows, Novell, Apple... Ngoài ra, với tính năng mã nguồn mở, hệ điều hành này còn cho phép khả năng tuỳ biến cao, thích hợp cho các nhu cầu sử dụng cụ thể.

Cũng như các hệ Unix khác, việc quản trị hệ thống Linux nói chung là một việc tốn khá nhiều công sức. Công việc này không những đòi hỏi sự hiểu biết về hệ thống, về mạng máy tính mà còn đòi hỏi đức tính cẩn thận, kiên trì, đầu óc tổ chức của người quản trị.

*Trong những năm gần đây, hệ điều hành Linux từng bước đã được đưa vào sử dụng tại Việt Nam. Nhiều tổ chức, công ty và các dự án tin học đã chọn Linux là môi trường để phát triển các ứng dụng của mình. Chính vì thế nhu cầu tìm hiểu hệ điều hành này đang trở nên rất lớn. Để đáp ứng phần nào yêu cầu nêu trên, chúng tôi đã cho ra đời bộ sách gồm 3 cuốn : "**Nhập môn hệ điều hành Linux**", "**Quản trị hệ thống Linux**" và "**Bên trong nhân hệ điều hành Linux**". Bộ sách được biên soạn dựa trên những kinh nghiệm tích luỹ ban đầu của chúng tôi trong quá tình thực hiện dự án "Xây dựng hệ thống tính toán hiệu năng cao" (trong*

*khuôn khổ **Chương trình Quốc gia về CNTT**, nay là **Chương trình Kinh tế Kỹ thuật về CNTT**), trong đó hệ điều hành Linux được sử dụng như hệ điều hành mạng với nhiều máy tính được kết chùm với nhau nhằm phục vụ cho việc giải quyết các bài toán cỡ lớn.*

*Cuốn sách "**Quản trị hệ thống Linux**" này được chia làm 5 phần:*

***Chương I. Bắt đầu với hệ thống Linux**, trong đó giới thiệu một số vấn đề cơ bản trong quản trị hệ thống Linux như quá trình khởi động và kết thúc hệ thống, vấn đề phân quyền trong Unix, quản lý người sử dụng.*

***Chương II. Quản lý tài nguyên**, đề cập đến các vấn đề quản lý việc sử dụng tài nguyên của hệ thống như máy in, các thiết bị lưu trữ, các tiến trình. Đồng thời cũng giới thiệu các chương trình lập lịch để thực hiện tự động hoá việc quản trị hệ thống.*

***Chương III. Quản lý cấu hình mạng**, giới thiệu các vấn đề cơ bản trong việc lập cấu hình cho mạng máy tính sử dụng Linux. Bao gồm các vấn đề như mạng máy tính sử dụng giao thức TCP/IP, hệ thống tên miền, việc thiết lập và quản mạng cấu hình cho mạng TCP/IP.*

***Chương IV. Quản lý dịch vụ mạng**, đề cập đến việc sử dụng, thiết lập và quản lý các dịch vụ mạng được cung cấp trong môi trường Linux như dịch vụ tên miền DNS, dịch vụ hệ thống tệp mạng NFS, hệ thông tin mạng NIS và việc tích hợp Linux với mạng Windows sử dụng Samba.*

***Chương V. Khai thác Shell và Kernel**, giới thiệu về các trình thông dịch lệnh (shell) của Linux và việc lập trình trên shell. Đồng thời cũng giới thiệu việc quản lý, thay đổi hay nâng cấp hạt nhân của Linux.*

Trong quá trình biên soạn, chúng tôi đã hết sức cố gắng, song không tránh khỏi những sai sót. Rất mong nhận được những ý kiến đóng góp và nhận xét của độc giả để có thể hoàn thiện cuốn sách trong những lần xuất bản tiếp theo. Mọi ý kiến đóng góp xin gửi về:

Nguyễn Thanh Thuỷ

Khoa Công nghệ Thông tin, Trường Đại học Bách khoa Hà Nội

Tel 8 696 124, email: thuynt@it-hut.edu.vn

*Để cuốn sách sớm ra mắt bạn đọc, chúng tôi đã nhận được nhiều động viên quý báu của Ban chủ nhiệm **Chương trình Kinh tế Kỹ thuật về CNTT**, Ban Giám hiệu trường Đại học Bách khoa Hà nội, và những đóng góp về chuyên môn của các Giáo sư, Thầy cô giáo trong Khoa Công nghệ Thông tin. Chúng tôi xin chânthành cảm ơn những sự giúp đỡ hiệu quả đó.*

Các tác giả

Hà nội, Thu 2000

Lời tựa cho lần tái bản thứ nhất

Kể từ lần xuất bản đầu tiên năm 2000đến nay, cuốn sách của chúng tôi luôn luôn nhận được sự quan tâm, nhiệt tình đóng góp của đông đảo các Thầy Cô, các Chuyên gia, cũng như các bạn sinh viên tại các khoa Công nghệ Thông tin, các khoa chuyên môn có liên quan tại các trường đại học, cao đẳng trong cả nước. Chúng tôi xin trân trọng cảm ơn sự động viên quý báu của bạn đọc đông đảo gần xa.

Trong lần tái bản có chỉnh lý bổ sung này, chúng tôi giới thiệu thêm với bạn đọc một số script tiện ích thường được dùng trong việc quản trị hệ thống. Đây là script khởi tạo các dịch vụ phổ biến cho một máy tính cũng như các máy chủ. Các dịch vụ này bao gồm cả dịch vụ người dùng như dịch vụ máy chủ Web, máy chủ lưu trữ FTP và dịch vụ hệ thống như dịch vụ syslog để ghi nhật ký hoạt động của hệ thống. Chúng tôi muốn nhấn mạnh khả năng vượt trội của script trong công việc quản trị hệ thống.

Trong những năm gần đây, cùng với xu thế tìm hiểu và phát triển các ứng dụng theo tiếp cận nguồn mở, trong đó HĐH LINUX được coi là một sản phẩm điển hình, nhiều tổ chức, công ty và các dự án tin học đã chọn Linux là môi trường để phát triển các ứng dụng của mình. Nhiều khoa Công nghệ thông tin và khoa chuyên môn có liên quan đến công nghệ thông tin đã chính thức đưa việc giảng dạy HĐH LINUX vào chương trình đào tạo. Chúng tôi hy vọng sẽ sớm giới thiệu với bạn đọc các cuốn sách "Bên trong nhân hệ điều hành Linux" và "Lập trình LINUX" trong thời gian sớm nhất.

Trong quá trình biên soạn, chúng tôi đã hết sức cố gắng, song không tránh khỏi những sai sót. Rất mong nhận được những ý kiến đóng góp và nhận xét của độc giả để có thể hoàn thiện cuốn sách trong những lần xuất bản tiếp theo. Mọi ý kiến đóng góp xin gửi về:

Nguyễn Thanh Thuỷ, Khoa Công nghệ Thông tin

Trường Đại học Bách khoa Hà nội

Tel (04)8.696.124, Email: thuynt@it-hut.edu.vn

Chúng tôi xin chân thành cảm ơn sự giúp đỡ hiệu quả của Chương trình Kinh tế Kỹ thuật về CNTT (Bộ KHCN), Ban Giám hiệu trường Đại học Bách khoa Hà nội và những đóng góp về chuyên môn của các Giáo sư, Thầy cô giáo trong Khoa Công nghệ Thông tin.

Các tác giả

Hà nội, xuân 2005

MỤC LỤC

Mở đầu

LINUX VÀ QUẢN TRỊ HỆ THỐNG LINUX

I. Tổng quan về hệ điều hành Linux

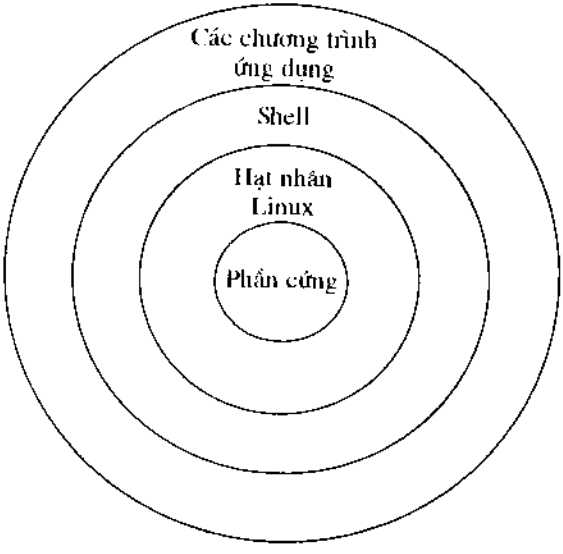
Linux là hệ điều hành thu hút được nhiều sự chú ý của giới tin học trong vòng vài năm trở lại đây. Ngay từ khi xuất hiện, ảnh hưởng của nó đã lan rộng nhanh chóng và được biết tới như một hệ điều hành UNIX - có mã nguồn mở. Thành công của Linux xuất phát từ cơ sở làm lại một trong những hệ điều hành lâu đời nhất và hiện vẫn đang được sử dụng rộng rãi, đó là hệ điều hành UNIX. Linux bao gồm cả các công nghệ cũ và mới.

Nhìn từ góc độ kỹ thuật, Linux chỉ là một nhân hệ điều hành, hỗ trợ đầy đủ các phục vụ cơ bản về quản lý tiến trình, bộ nhớ ảo, quản lý tệp và thiết bị vào ra. Nói cách khác, bản thân Linux là phần thấp nhất của hệ điều hành. Hình 1 dưới đây cho ta một cách nhìn tổng thể về hệ thống Linux.

Tuy nhiên, phần lớn người dùng đều quan niệm "Linux" như một hệ thống hoàn chỉnh bao gồm nhân hệ điều hành cùng với các trình ứng dụng khác: một môi trường làm việc và phát triển đầy đủ bao gồm trình dịch, các hệ soạn thảo, giao diện đồ hoạ, xử lý văn bản, ... Hiện nay, với phiên bản Linux RedHat 6.2, Linux đã trở thành một hệ điều hành đầy đủ cho thương mại, giáo dục hoặc người dùng cá nhân.

Linux có thể được cài đặt trên một máy tính cá nhân và biến nó thành một trạm làm việc với đầy đủ sức mạnh của UNIX. Linux có thể được sử dụng với mục đích thương mại trên một mạng máy tính như một môi trường tính toán và truyền tin. Trong các trường đại học, Linux được sử dụng để giảng dạy về hệ điều hành và

lập trình hệ điều hành. Linux cũng có thể được sử dụng trên các máy tính cá nhân giống như các hệ điều hành khác.



Hình 1. Kiến trúc hệ điều hành Linux

Ngoài chức năng trạm làm việc và người sử dụng đơn lẻ, Linux còn được sử dụng trên các máy chủ lớn. Linux ngày càng tỏ ra là một hệ điều hành mạnh, ổn định và đủ mềm dẻo để chạy trên các hệ thống với bộ nhớ lớn và đa xử lý. Linux được dùng để nối ghép nhiều máy tính với nhau tạo thành các *cluster* rất mạnh để giải quyết các nhu cầu tính toán hiệu năng cao.

Điều làm cho Linux trở nên khác biệt là việc cung cấp mã nguồn mở của nó. Việc này do một nhóm chuyên môn phát triển tự nguyện trên mạng Internet, họ trao đổi mã, phát hiện và sửa lỗi những ứng dụng trên Linux trong một môi trường mở.

II. Các nhiệm vụ quản trị hệ thống

Cũng như việc quản trị hệ thống nói chung, quản trị hệ thống Linux là một công việc tương đối phức tạp. Công việc này đòi hỏi sự kiên nhẫn, sự hiểu biết và kinh nghiệm của người quản trị. Quản trị hệ thống Linux không chỉ dừng lại ở việc cài đặt một máy tính hay mạng máy tính sử dụng hệ điều hành Linux. Thực chất, công việc này đòi hỏi phải lập kế hoạch, thiết kế một mạng máy tính Linux để

những người sử dụng của hệ thống có thể tiến hành hiệu quả công việc của họ.

Công việc của một người quản trị hệ thống là rất nhiều và đa dạng, Nó bao gồm quá trình cài đặt và cấu hình các thiết bị phần cứng, cài đặt phần mềm, sửa đổi cấu hình hạt nhân, thiết lập các hoạt động cho mạng máy tính, bảo mật hệ thống... Các công việc này đòi hỏi người quản trị có mức đặc quyền cao nhất đối với hệ thống mà những người sử dụng bình thường khác không có. Sau đây là những công việc của người quản trị hệ thống.

1. Quản lý người sử dụng

Người quản trị hệ thống có nhiệm vụ cung cấp tài khoản cho người sử dụng mới muốn làm việc trên hệ thống, đồng thời loại bỏ các tài khoản không còn hiệu lực. Quá trình thêm bớt người sử dụng có thể được tiến hành tự động, nhưng vẫn cần có những tuỳ chọn riêng cần đưa vào như tên người dùng, vị trí của thư mục người dùng... trước khi thêm người sử dụng mới.

Khi một người sử dụng không còn truy cập vào hệ thống nữa, tài khoản tương ứng phải được loại khỏi hệ thống vì các lý do an toàn và hành chính. Đồng thời các tệp tin mà người đó chiếm dụng phải được xoá bỏ để giải phóng tài nguyên cho hệ thống.

2. Quản lý các thiết bị phần cứng

Khi có thiết bị phần cứng mới thêm vào, hệ thống phải được cấu hình lại để có thể nhận biết và sử dụng các thiết bị phần cứng mới đó. Công việc này nói chung là khá phức tạp vì các thiết bị phần cứng rất đa dạng và phong phú, trong khi đó việc hỗ trợ phần cứng của hệ điều hành Linux lại có phần hạn chế.

Tương tự như vậy khi một thiết bị phần cứng bị loại bỏ, ta cũng phải cấu hình lại hệ thống để tránh các lỗi có thể xảy ra như hệ thống không khởi động được, hay thu hồi lại bộ nhớ đã cung cấp cho thiết bị đó, nhưng không cần dùng nữa.

3. Tiến hành sao lưu

Sao lưu là một trong những công việc rất quan trọng trong quản trị hệ thống, tuy nhiên trong thực tế công việc này thường bị xem nhẹ hay bỏ qua. Đây là công việc tương đối nhàm chán và tốn thời gian, song chúng lại vô cùng cần thiết trong các hệ thống lớn đòi hỏi độ an toàn cao về dữ liệu. Công việc sao lưu thường được

tiến hành tự động, nhưng nhiệm vụ của người quản trị là phải bảo đảm rằng sao lưu được tiến hành thường xuyên và đúng đắn.

4. Cài đặt phần mềm

Khi có nhu cầu sử dụng phần mềm mới trong hệ thống, nó phải được cài đặt và kiểm tra lại một cách đúng đắn. Khi phần mềm đã hoạt động tốt, người sử dụng tương ứng phải được thông báo đầy đủ về nó. Cài đặt phần mềm phải được tổ chức khoa học, các ứng dụng kèm theo Linux và các ứng dụng của người dùng phải để ở những nơi khác nhau, bảo đảm thuận tiện cho việc loại bỏ phần mềm khỏi hệ thống.

5. Bảo đảm các hoạt động thường ngày

Có vô vàn các công việc thường ngày khác nhau cần phải được tiến hành, bao gồm việc bảo đảm hoạt động của hệ thống thư điện tử, theo dõi các liên kết mạng hay theo dõi việc sử dụng tài nguyên hệ thống như dung lượng đĩa...

6. Khắc phục sự cố

Hệ thống Linux cũng như các thiết bị phần cứng của nó có thể bị lỗi bất cứ lúc nào. Khi đó, nhiệm vụ của người quản trị là phải tìm cách khắc phục hoặc gọi cho các chuyên gia nếu cần thiết. Thông thường, việc tìm ra được nguyên nhân thường khó hơn nhiều so với việc khắc phục sự cố.

7. Cập nhật tài liệu về hệ thống

Mỗi hệ thống thường được cấu hình theo cách riêng cho phù hợp voái hoàn cảnh và điều kiện sử dụng. Do đó người quản trị phải ghi chép vào tài liệu tất cả các vấn đề đặc trưng cho hệ thống mà mình quản lý. Các tài liệu bao gồm việc mô tả lại các phần mềm đã cài đặt, cấu hình của hệ thống, của mạng, các hoạt động bảo trì thiết bị phần cứng và quá trình sao lưu...

8. An toàn và bảo mật thông tin

Người quản trị phải đề ra một chính sách an toàn và bảo mật thông tin, đồng thời thường xuyên kiểm tra xem chính sách đó có được tôn trọng hay không. Trong các hệ thống an toàn cao, cần kiểm tra các truy cập trái phép của người sử dụng hay các truy cập bất hợp pháp từ bên ngoài hệ thống.

9. Giúp đỡ người sử dụng

Mặc dù việc giúp đỡ người sử dụng không được mô tả nhiều trong các công việc của người quản trị nhưng trong thực tế, đây là công việc chiếm mất phần lớn thời gian trong ngày của họ. Người quản trị thường bị ngập đầu với các câu hỏi "khẩn cấp" và đầy vẻ quan trọng" của người dùng, chẳng hạn như "Tại sao chương trình này hôm qua chạy tốt thế mà sao hôm nay nó lại không chạy, hay "Làm sao mà bàn phím lại bị treo cứng thế này"...

10. Theo dõi hoạt động của người dùng hệ thống

Linux cho phép ghi lại các hoạt động của người dùng trong hệ thống như việc sử dụng CPU, máy in hay các tài nguyên khác. Hầu hết trạm làm việc đều không thực hiện công việc này, song trong nhiều trường hợp nó lại rất có ích. Chẳng hạn việc ngăn cản các truy cập trái phép hay việc truy tìm những tin tặc đã xâm nhập hệ thống. Trong một số trường hợp, người dùng sẽ được cung cấp một bản tổng kết việc sử dụng hệ thống của họ vào dịp cuối tháng.

Chương I

BẮT ĐẦU VỚI HỆ THỐNG LINUX

I. KHỞI ĐỘNG VÀ KẾT THÚC LINUX

1. Khởi động

Công việc cơ bản nhất mà người sử dụng thực hiện với hệ thống Linux là khởi động và kết thúc Linux. Mặc dù đây là những công việc đơn giản nhưng cũng có nhiều cách để thực hiện. Nếu thực hiện không đúng sẽ dẫn đến hậu quả không tốt, chẳng hạn việc ngắt điện đột ngột máy có thể làm hư hại hệ thống tệp. Do vậy cần cẩn thận khi tắt hệ thống để bảo vệ thông tin trên máy.

Trong phần này ta sẽ xem xét quá trình khởi động, đóng Linux và chương trình *init* (Còn gọi là daemon *init* - một chương trình thường trú trên Linux). *Init* có thể coi là tiến trình quan trọng nhất chạy trên tất cả các hệ thống Linux. Việc hiểu rõ chương trình này làm gì và như thế nào sẽ ta khai thác Linux tốt hơn.

1.1. Khởi động Linux

Để khởi động Linux người dùng chỉ cần bật máy. Nếu hệ điều hành được cấu hình để tự động nạp thì chỉ sau vài giây Linux đã sẵn sàng. Tuy vậy, số hệ thống chỉ cài đặt Linux không nhiều và số hệ thống được cấu hình để tự động nạp Linux còn ít hơn nữa. Mặc dù khởi động tự động rất thuận tiện nhưng nhiều người dùng vẫn thích được chọn hệ điều hành khởi động (nếu có nhiều hệ điều hành được

cài đặt trong máy) hoặc thay đổi mức độ truy nhập Linux hơn.

Có hai cách khởi động Linux: khởi động dùng đĩa mềm hoặc dùng LILO. Mỗi phương pháp đều có các ưu nhược điểm riêng mà chúng sẽ được xem xét trong các phần sau đây.

1.2. Dùng LILO để khởi động hệ thống

Dùng LILO để khởi động hệ thống là phương pháp hay được sử dụng nhất vì không cần dùng đĩa mềm. LILO là một chương trình nằm ở cung từ khởi động của một phân vùng đĩa cứng hoặc trên bản ghi khởi động của toàn bộ đĩa cứng. Nó trỏ đến phân vùng ổ cứng và vị trí của hạt nhân Linux.

Nếu LILO được cài đặt để nạp chương trình khởi động ở giai đoạn đầu tiên có nghĩa là khởi động Linux tự động, Linux sẽ được khởi động khi nào bật máy. Muốn dừng quá trình khởi động ta có thể nhấn tổ hợp Ctrl+Alt+Del khi máy bắt đầu quá trình khởi động. (Nhất thiết phải chờ đến khi bắt đầu nạp hệ điều hành mới nhấn tổ hợp phím, nếu không chúng ta có thể khởi động lại máy ngoài ý muốn). Tổ hợp phím này sẽ ra lệnh cho LILO dừng và hiển thị dấu nhắc :

```
Boot :
```

Từ dấu nhắc ta có thể thông báo cho Linux hệ điều hành mà nó sẽ nạp (DOS, Linux, OS/2,...). Nếu ta nhấn Tab khi xuất hiện dấu nhắc này, LILO sẽ liệt kê danh sách tất cả các phân vùng và các hệ điều hành nó biết. Thông tin cấu hình hệ điều hành của các phân vùng phải chứa trong thông tin LILO. Việc cung cấp các thông tin này cho LILO chỉ đơn giản là đưa ra định danh phân vùng và tên của hệ điều hành khi cấu hình LILO.

Vì LILO ghi dữ liệu lên đĩa mà các hệ điều hành khác không đọc được, không nên thường xuyên cài đặt thêm hay xoá một hệ điều hành khỏi đĩa cứng. Mỗi khi thay đổi cấu hình của Hệ thống Linux hoặc các phân vùng khác trên ổ cứng, ta phải cập Nhật lại thông tin của LILO bằng cách thộc hiện lệnh *lilo*.

1.3. Khởi động hệ thống sử dụng đĩa mềm

Nếu không muốn dựa vào LILO (vì đòi hỏi phải thay đổi các cung từ của đĩa, điều này có thể phiền toái nếu chúng ta sẽ dụng nhiều hệ điều hành hoặc thay đổi hệ điều hành thường xuyên), ta có thể khởi động bằng đĩa mềm khởi động. Đĩa mềm khởi động chỉ gồm một đĩa là chứa đủ hạt nhân Linux và các lệnh truy nhập phân vùng gốc của đĩa cứng. Đĩa mềm khởi động phải có định dạng phù hợp để

chạy trên ổ đĩa mềm đầu tiên của hệ thống (tương ứng ổ A trong DOS). Linux không thể khởi động từ ổ đĩa mềm thứ hai.

Khởi động từ đĩa mềm là cách khởi động dễ nhất và linh hoạt nhất. Giả sử chúng ta có một đĩa cứng được phân vùng cho cả DOS và Linux, trong đó DOS ở phân vùng khởi động mặc định, chúng ta chỉ cần bật máy là DOS được khởi động. Nếu muốn khởi động từ Linux thì ta chèn đĩa mềm khởi động Linux vào ổ mềm thứ nhất và bật máy, khi đó Linux khởi động từ đĩa mềm và ta có thể truy nhập các ổ cứng như thể đã khởi động từ ổ cứng.

Khi dùng đĩa mềm để khởi động Linux cần chú ý cập nhật ảnh hạt nhân (kernel image) trên đĩa mềm này mỗi khi có thay đổi hệ thống đòi hỏi phải xây dựng lại hạt nhân. Hãy nhớ xây dựng lại hạt nhân mỗi khi thêm một thiết bị hoặc một trình điều khiển thiết bị. Sử dụng thủ tục mô tả sau đây để cập nhật đĩa khởi động, hoặc tốt hơn hết là tạo một đĩa khởi động mới và dùng đĩa này làm đĩa cấp cứu.

Để tạo một đĩa mềm khởi động, ta cần một đĩa mềm đã định dạng mật độ cao (1.44M hoặc 1.2M tuỳ thuộc vào ổ đĩa mềm thứ nhất của máy). Một số loại hạt nhân nhỏ có thể chứa vừa trong đĩa mật độ thấp nhưng ổ đĩa mật độ cao đã trở thành chuẩn. Định dạng đĩa dưới DOS để đặt các cung và rãnh từ cho Linux có thể đọc được. Đĩa có thể chứa cả các thông tin khác và hạt nhân Linux, nhưng cần đảm bảo đủ không gian cho ảnh hạt nhân.

Một số phiên bản của Linux cho phép tạo đĩa mềm khởi động như một phần của quá trình cài đặt. Khi đó hãy chọn tuỳ chọn *Configure* để tạo đĩa khởi động. Trường hợp ngược lại, một số phiên bản có thực đơn chọn lựa riêng để tạo đĩa mềm khởi động.

Nếu muốn tạo đĩa mềm khởi động bằng tay, xác định hạt nhân Linux trong hệ thống, thông thường ở thư mục gốc, một số phiên bản lưu hạt nhân tại thư mục */etc*. Tên hạt nhân khác nhau tuỳ theo phiên bản Linux, thường là *image* hoặc *vmlinux*. Một vài phiên bản lưu hạt nhân dưới dạng nén với đuôi *z* như *vmlinuz* hoặc *vmlinux.z*. Hạt nhân nén tốn ít không gian đĩa hơn và chỉ được giải nén khi hạt nhân Linux khởi động. Tuy vậy hạt nhân nén làm việc khởi động lâu hơn chút ít so với hạt nhân không nén.

Khi đã chỉ ra tệp hạt nhân, cần chỉ cho Linux biết rằng tệp này sẽ là thiết bị gốc (root device) và nó nằm trên phân vùng nào bằng lệnh *rdev*. Ví dụ câu lệnh để

đặt thiết bị gốc cho hạt nhân *vmlinuz* trong thư mục gốc của phân vùng */dev/hda3* là :

```
$ rdev /vmlinuz /dev/hda3
```

Vì phải chỉ ra đường dẫn đầy đủ của hạt nhân nên / đầu tiên để chỉ thư mục gốc. Kết qủa là */dev/hda3* trở thành hệ thống tệp gốc.

Lệnh *rdev* không có tham số cho biết phân vùng hiện tại của hệ thống tệp gốc. Ta có thể dùng lệnh này để kiểm tra phân vùng nào là hệ thống tệp gốc.

```
$ rdev
/dev/hda3 /
```

Sau khi đã đặt thiết bị gốc, ta ghi hạt nhân vào đĩa mềm đã định dạng. Dùng lệnh *cp* với tên thiết bị của ổ đĩa mềm :

```
$ cp /vmlinuz /dev/fd0
```

Khi tệp ảnh đã được chuyển sang, đĩa mềm đã có thể sử dụng để khởi động Linux. Nếu không hoặc là ảnh không được chuyển tốt do thiếu không gian đĩa hoặc đĩa có cung từ hỏng hoặc ảnh hạt nhân có vấn đề.

2. Tạo và sử dụng đĩa bảo trì

Các đĩa bảo trì hệ thống (còn gọi là đĩa khởi động cấp cứu) được dùng để khởi động Linux khi có lỗi xảy ra với hệ thống khởi động (như LILO chẳng hạn). Đĩa bảo trì gồm đĩa khởi động và đĩa root, có thể khởi động toàn bộ hạt nhân Linux mà không phụ thuộc vào việc cài đặt trên ổ cứng. Sau khi nạp đĩa bảo trì, ta có thể sử dụng nó để nối đĩa cứng và sửa chữa các vấn đề trên đĩa, hoặc sử dụng một tiện ích trên đĩa cứng để xây dựng lại LILO hoặc hạt nhân.

Để tạo đĩa bảo trì cần tạo hệ thống tệp gốc trên đĩa mềm, chép các công cụ cần thiết, cài đặt LILO, chép hạt nhân để đĩa khởi động được. Công việc này cần được thực hiện mỗi khi thay đổi hạt nhân Linux để đĩa bảo trì có cùng hạt nhân với đĩa cứng.

Nếu phải rút đĩa bảo trì ra vì lí do nào đó, hãy gắn kết đĩa cứng hiện tại bằng lệnh mount vào một thư mục trống đã có. Ví dụ muốn nối phân vùng */dev/hda2*, với kiểu của hệ thống tệp là *ext2*:

```
$ mount -t ext2 /dev/hda2 /mnt
```

3. Đóng Linux

Khi ngắt nguồn điện tất cả mọi thứ sẽ bị đóng. Tuy nhiên trong thực tế, cách làm này có thể dẫn đến việc toàn bộ nội dung của phân vùng đĩa cứng sẽ bị hỏng, đồng thời cũng làm mất tất cả thông tin mà ta vừa mới làm xong.

Linux quản lí đĩa và không gian dành cho người sử dụng trong RAM, nó sử dụng bảng *i-node* để làm việc dữ liệu trên đĩa và một trình quản lí bộ nhớ để làm việc với dữ liệu của người sử dụng. Bảng *i-node* là bảng các con trỏ trỏ đến các tệp của Linux, mỗi tệp được trỏ bởi một *i-node* chứa một số thông tin về tệp đó. Linux thường xuyên ghi những thay đổi trong bảng *i-node* vào đĩa, nhưng lại coi bản sao dữ liệu trong RAM là bản sao gần nhất vì RAM có tốc độ truy cập lớn, và việc ghi lại bản sao này lên đĩa bị trì hoãn đến khi nào thật cần thiết. Nếu ta tắt nguồn điện trước khi Linux ghi các thay đổi lên đĩa, nội dung của đĩa và bảng *i-node* được ghi trên đĩa có thể không phù hợp với nhau, điều đó dẫn đến việc mất tệp hoặc tạo ra một danh sách không chính xác những không gian còn trống trên đĩa. Trường hợp tồi hơn, nếu ngắt điện trong khi Linux đang trong quá trình ghi bảng *i-node* và các thông tin khác lên đĩa có thể làm hỏng đầu từ hoặc tạo nên các cung hỏng trên đĩa.

Có 2 cách đơn giản để đóng Linux một cách hợp thức. Cách đơn giản nhất là dùng tổ hợp phím Ctrl+Alt+Del. Đa số các phiên bản của Linux bẫy tổ hợp phím này và phát ra lệnh đóng tất cả các tiến trình một cách hợp thức. Với những phiên bản không bẫy tổ hợp phím này thì việc sử dụng nó có tác dụng tương đương với việc ngắt nguồn điện.

Một phương pháp khác là dùng lệnh của Unix :

```
$ shutdown
```

Khi lệnh này được đưa ra, Linux kết thúc tất cả các tiến trình và đóng hạt nhân Linux. Lệnh này có thể đưa ra một vài thông báo tuỳ theo phiên bản Linux, nhằm thông báo quá trình đóng hoặc yêu cầu khẳng định lại rằng ta thực sự muốn đóng hệ thống.

Lệnh *shutdown* cho phép ta đưa ra thời gian chờ cho đến khi đóng hệ thống hoặc thông báo sẽ hiển thị cho những người dùng khác đang đăng nhập vào hệ thống.

```
$ shutdown thoi_gian thong_bao
```

Ví dụ câu lệnh:

```
$ Shutdown 15 "Đa đen luc luu lai !"
```

Sẽ đóng hệ thống sau 15 phút và hiện thông báo "Đa đen luc luu lai !" cho mọi người dùng hệ thống để nhắc hệ thống sắp tắt.

Trong hầu hết các phiên bản của Linux, lệnh *shutdown* có một tuỳ chọn *-r*, cho phép máy tính khởi động lại sau khi đóng hệ thống. Có thể dùng tuỳ chọn này để khởi động hệ điều hành khác hoặc khởi động lại Linux sau khi đã thay đổi hạt nhân hoặc các thiết bị.

Nói chung khi dùng tổ hợp phím Ctrl+Alt+Del hoặc lệnh *shutdown* đều hiển thị một số thông báo trạng thái. Chẳng hạn "The system is halted". Khi có thông báo này, ta có thể tắt điện hoặc khởi động lại một cách an toàn.

4. Chương trình Init

Chương trình *init* liên quan đến bước khởi động cuối cùng của Linux, đây là một trong những daemon quan trọng nhất của Linux vì nó tạo ra các tiến trình cho phần tiếp theo của hệ thống. Sau khi được thực hiện *init* vẫn hoạt động cho đến khi đóng Linux. Hiểu rõ *init* làm gì (và cả tiện ích đi đôi với nó *telinit*) và nó điều khiển điều hành như thế nào sẽ giúp ta quản trị hệ thống Linux tốt hơn.

Cả *init* và *telinit* đều sử dụng một số tệp cấu hình do đó chúng ta sẽ xem xét các tệp này. Đó là các tệp liên quan đến việc khởi động và kết thúc một thiết bị đầu cuối và một phiên làm việc. Để thực thi chương trình *init*, Linux tìm nó lần lượt trong các thư mục : */etc*, */bin* hoặc */sbin*. Chương trình *telinit* cũng ở cùng thư mục với *init*, còn các tệp cấu hình thì luôn ở thư mục */etc*. Trước hết ta tìm hiểu các mức hoạt động.

4.1. Các mức hoạt động

Khi chạy, *init* đọc các lệnh trong tệp */etc/inittab*, mục tiêu chính là để khởi động các tiến trình *getty* cho các *terminal* và các tiến trình khác của Linux. Khi duyệt */etc/inittab*, *init* tìm dòng lệnh có nhãn *initdefaut*, trong đó chỉ ra mức hoạt động mặc định của hệ thống. Nếu không tìm thấy, hệ thống sẽ hỏi mức hoạt động.

Mức hoạt động là một thiết lập đặc biệt của các tiến trình từ mức thấp nhất (chỉ dành cho quản trị hệ thống) cho đến một hoạt động đầy đủ hỗ trợ các thiết bị đã được cấu hình. Mức hoạt động được định nghĩa bằng các con số từ 0 đến 6 cộng

với một mức siêu người dùng được định nghĩa là *s* (còn gọi là mức một người dùng mà chỉ có root mới có quyền tham gia). *Init* biết tiến trình nào tương ứng với mức hoạt động nào nhờ thông tin trong tệp */etc/inittab*.

Khi chạy hệ thống ở mức *s* trong chế độ 1 người dùng, *init* không đọc tệp */etc/inittab* mà đọc tệp */bin/su* liên quan đến *console* hệ thống, được định nghĩa trong */dev/console*. Khi được ra lệnh chuyển sang chế độ một người dùng từ chế độ hoạt động ở mức cao hơn, tiến trình *init* sẽ lưu trạng thái của hệ thống hiện tại trong tệp */etc/ioctl.save* bằng chương trình *ioctl*. Khi *console* được khởi động lại ở mức hoạt động cao hơn, trạng thái lưu sẽ được khôi phục, nếu không có tệp *ioctl.save* trạng thái mặc định được thiết lập.

Khi bắt đầu ở chế độ đa người dùng (mức hoạt động cao hơn mức một người dùng), *init* sẽ thực hiện mọi dòng lệnh *boot* và *bootwait* trong tệp */etc/inittab*, các lệnh này thông thường cho phép nối các hệ thống tệp. Sau đó thực hiện các dòng lệnh khác có mức hoạt động trùng với mức đang được chọn. Điều này rất thuận lợi để nhóm các dòng lệnh trong */etc/inittab* nhằm tạo ra các cấu hình khác nhau. Chẳng hạn, ta quyết định rằng mức 1 chỉ thực hiện một số tối thiểu các script của cấu hình, mức 2 cho phép truy nhập các cổng tuần tự, mức 3 sẽ kích hoạt mạng,...

Mức hoạt động của hệ thống có thể thay đổi được nhờ chương trình */etc/telinit*. Tuy vậy việc truy nhập trương trình này thường bị hạn chế bởi người quản trị hệ thống vì lí do an toàn. Chương trình này gửi đến *init* thông báo yêu cầu mức hoạt động mới. Ví dụ:

```
$ telinit 2
```

chuyển mức hoạt động sang 2, khiến cho *init* đọc lại */etc/initab* và thực hiện tất cả các tiến trình ở mức đó và kết thúc các tiến trình có mức cao hơn. Để chuyển sang mức siêu người dùng, tức là chế độ 1 người dùng, ta đưa ra lệnh

```
$ telinit s
```

Với lựa chọn -t ta có thể yêu cầu chuyển mức hoạt động sau một khoảng thời gian. Ví dụ, để chyển sang mức 3 sau 5 giây (thời gian mặc định của Linux là 20 giây):

```
$ telinit -t5 3
```

Khi thay đổi mức hoạt động, *init* gửi một cảnh báo *SIGTERM* đến tất cả các tiến trình không phù hợp với mức hoạt động mới. Một khoảng thời gian sau khi gửi tín hiệu (mặc định là 20 giây), các tiến trình bị dừng bắt buộc. Nếu một tiến trình

được kích hoạt bằng *init* lại sinh ra tiến trình con không thuộc cùng nhóm thì *init* không dừng tiến trình con này khi thay đổi mức hoạt động. Ta phải tự dừng tiến trình này bằng tay.

4.2. Tệp /etc/inittab

Như đã thấy tệp */etc/inittab* có mối quan hệ chặt chẽ với *init*, từ tệp này *init* xác định mức hoạt động khi Linux khởi động và những tiến trình nào được kích hoạt.

Cấu trúc của mỗi dòng trong tệp */etc/inittab* như sau :

```
<mã> : <mức> : <hành động> : <lệnh>
```

Trong đó :

<mã> chứa một chuỗi duy nhất một hoặc hai kí tự phân biệt dòng này với các dòng khác trong tệp.

<mức> chứa danh sách các mức hoạt động.

<hành động> chỉ cách xử lí lệnh này, chẳng hạn chỉ thực hiện lệnh một lần hay thực hiện lại mỗi khi lệnh thực hiện xong...

<lệnh> chỉ lệnh mà *init* sẽ thực hiện.

Sau đây là các loại hành động và ý nghĩa của chúng

- *boot* : chạy khi *inittab* được đọc lần đầu
- *bootwait* : chạy khi *inittab* được đọc lần đầu
- *initdefault* : đặt mức hoạt động khởi tạo
- *off* : kết thúc tiến trình đang chạy
- *once* : kích hoạt tiến trình một lần
- *ondemand* : luôn giữ tiến trình chạy (cũng như *respawn*)
- *powerfail* : thực hiện khi init nhận tín hiệu mất điện (tín hiệu *UPS* báo cho PC).
- *powerwait* : thực hiện khi init nhận tín hiệu mất điện.

- *sysinit* : thực hiện trước khi truy nhập *console*
- *respawn* : luôn luôn giữ tiến trình chạy, tức là nếu nó kết thúc thì chạy lại
- *wait* : kích hoạt một tiến trình một lần

Chúng ta xem xét sau đây một tệp */etc/inittab* mẫu. Dòng đầu tiên của tệp có trường hành động là *initdefault*, do đó nó xác định mức hoạt động mặc định, ở đây là mức 5.

Id : 5 : initdefault :

Dòng tiếp theo chỉ ra tệp */etc/rc.d/rc.s* chạy ngay khi khởi động (trường hành động bằng *sysinit*)

si: S : sysinit : /etc/rc.d/rc.s

Dòng sau chỉ đến tệp */etc/rc.d/rc.K*, tệp này được thực hiện khi hệ thống chạy ở mức một người dùng.

su : S : wait : /etc/rc.d/rc.K

Dòng tiếp, thực hiện tệp */etc/rc.d/rc.M* khi hệ thống khởi động ở mức nhiều người dùng (bất kỳ mức nào từ 1 đến 6)

rc : 123456 : wait : /etc/rc.d/rc.M

Mức hoạt động chung nhất là mức 5, là mức chuẩn cho chế độ đa người dùng trong Linux, các mức khác ít khi được dùng, chủ yếu để điều khiển truy nhập thiết bị ngoại vi. Về mức hoạt động, tốt nhất là để như hệ thống muốn để tránh rắc rối, có nghĩa là ta sử dụng mức s cho chế độ siêu người dùng và 5 cho mục đích chung.

Vì tổ hợp phím **Ctrl+Alt+Del** không được hỗ trợ để tắt hệ thống trong các hệ thống Unix trên PC, do đó có một lệnh đặc biệt được gán cho tổ hợp này

ca:: ctrlaltdel: /sbin/shutdown -t3 -rf now

Tệp */etc/inittab* cũng chứa các lệnh khởi động các tiến trình *getty* cho mỗi thiết bị đầu cuối và *console* ảo của hệ thống (là các *console* mà ta tạo được khi đăng nhập vào hệ thống trong khi đang ở trong một đăng nhập bằng cách dùng tổ hợp phím *Alt* với các phím từ *F1* đến *F8*). Tệp mẫu này kích hoạt 6 cửa sổ ảo (*tty1*

đến *tty6*) và 2 đường tuần tự (*ttyS0* và *ttyS1*)

```
c1:12345:respawn:/sbin/agetty 38400 tty1
c2:12345:respawn:/sbin/agetty 38400 tty2c3:45:respawn:/sbin/agetty 38400 tty3
c4:45:respawn:/sbin/agetty 38400 tty4c5:45:respawn:/sbin/agetty 38400 tty5
c6:456:respawn:/sbin/agetty 38400 tty6s1:45:respawn:/sbin/agetty 19200 ttyS0
s2:45:respawn:/sbin/agetty 19200 ttyS1
```

Sau khi đọc xong tệp */etc/inittab*, *init* không kết thúc, nó vẫn hoạt động và điều khiển hệ thống cho các lệnh đặc biệt chẳng hạn thay đổi mức hoạt động. Nó đồng thời chịu trách nhiệm theo dõi các tiến trình đã được kích hoạt, bao gồm cả các tiến trình *getty* cho các *terminal*. Mỗi khi một tiến trình con của *init* kết thúc (tiến trình mà nó kích hoạt) *init* ghi mã sự kiện, lí do kết thúc vào tệp */etc/utmp* và */etc/wtmp*.

Mỗi khi *init* phát hiện ra một tiến trình con kết thúc hoặc tín hiệu mất điện, hoặc thay đổi mức hoạt động, nó đọc lại tệp */etc/inittab* để kiểm tra lại các lệnh. Chúng ta có thể thay đổi tệp *inittab* khi hệ thống đang chạy nhưng thay đổi chỉ có tác dụng khi khởi động lại hệ thống hoặc một trong các điều kiện đọc lại xảy ra. Một biện pháp bắt đọc lại là ta dùng tham số q để bắt *init* duyệt lại tệp */etc/inittab* :

```
$ init q
```

Tiến trình *init* luôn kiểm tra số lần phải kích hoạt lại một tiến trình . Nếu một tiến trình bị kích hoạt nhiều hơn 10 lần trong 2 phút, nó sẽ giả thiết rằng có lỗi trong dòng lệnh của tiến trình đó trong tệp */etc/inittab* và đưa ra thông báo lỗi trong cửa sổ hệ thống. *Init* sẽ không kích hoạt lại tiến trình đó trong 5 phút cho đến khi bị siêu người dùng bắt buộc, điều này chống việc lãng phí các chu kì CPU vì các lỗi in trong tệp */etc/inittab*.

5. Sử dụng họ rdev

Lệnh *rdev* không chỉ dùng để xác định ổ đĩa gốc (dùng khi tạo đĩa mềm khởi động và đĩa root) mà còn để lấy thông tin về hệ thống Linux và thay đổi một số cấu hình. Nhưng đây cũng là một công cụ sử dụng khá rắc rối. Nếu ta dùng LILO để khởi động Linux, ta có thể bỏ qua *rdev* vì các tham số của nó đã được đặt trong cấu hình của LILO. Ta chỉ dùng *rdev* khi thay đổi hạt nhân và tạo đĩa khởi động dùng

khi cấp cứu hoặc thay đổi kích thước *RAM disk*.

Chạy *rdev* không tham số sẽ cho biết phân vùng gốc hiện tại và thư mục gốc trên đó.

```
$ rdev
/dev/hda3 /
```

có nghĩa là */dev/hda3* đang là phân vùng gốc.

rdev đổi phân vùng gốc và trỏ đến ảnh hạt nhân sẽ được Linux sử dụng bằng 2 tham số :

```
$ rdev /vmlinuz /dev/hda3
```

Lệnh này chuyển sang ảnh hạt nhân *vmlinuz* trong thư mục gốc của phân vùng thứ 3.

rdev có một số tuỳ chọn :

-h : Hiển thị trợ giúp

-r : *rdev* làm việc giống lệnh *ramsize* (xác định kích thước *RAM disk* theo kilobyte)

-R : *rdev* làm việc giống lệnh *rootflags* (cho phép nối thư mục gốc dưới dạng chỉ đọc)

-s : *rdev* làm việc giống lệnh *swapdev* (xác định đĩa *swap*)

-v : *rdev* làm việc giống lệnh *vidmode* (đặc tả chế độ hiển thị)

Để thay đổi nhiều tham số, ta phải chỉ ra một *offset* là một giá trị thập phân chỉ vị trí hạt nhân trong lệnh *rdev*, đây là lí do các nhà quản trị không thích dùng *rdev*. Để dùng *rdev* hoặc một trong các tiện ích của họ này phải tính địa chỉ *offset* theo luật :

offset 498 : Root flag

offset 504 : kích thước RAM disk

offset 506 : chế độ VGA

offset 508 : đĩa gốc

Như vậy, trong phần này chúng ta đã xem xét thủ tục mở và đóng Linux,

đồng thời xem xét quá trình khởi tạo, một quá trình quan trọng của Linux. Bây giờ chúng ta sẽ xem tiếp một vài khía cạnh quan trọng của quản trị hệ thống.

II. TÊN HỆ THỐNG VÀ QUYỀN TRUY NHẬP

Thay vì phải dùng từ "nó" hay "cái đó" cho hệ thống Linux của mình, bạn có thể đặt cho nó một cái tên có ý nghĩa nào đó. Vấn đề này trở nên đặc biệt quan trọng khi làm việc với e-mail hoặc mạng bởi khi đó các hệ thống khác phải có một phương pháp để phân biệt một máy tính với các máy tính khác trên mạng. Phần này sẽ bắt đầu bằng việc xem xét cơ chế đặt tên cho hệ thống cũng như các quy luật cần tuân theo để đảm bảo các máy khác có thể làm việc với hệ thống mới được đặt tên.

Vấn đề tiếp theo được đề cập đến là quyền truy nhập. Khái niệm quyền thường bị hiểu sai hoàn toàn, và quyền gắn với tệp và thư mục thường được đặt sai, làm cho người dùng cần đến nó lại không truy nhập được, và tệ hơn nữa là cho phép truy nhập rộng rãi các thông tin nhạy cảm. Sau khi trình bày các quyền làm việc như thế nào, phần này sẽ trình bày làm thế nào để đổi và thiết lập các quyền và quyền sở hữu.

1. Đặt tên cho hệ thống

Linux được thiết kế để làm việc trên mạng nên cho phép xác định cho mỗi máy một tên duy nhất. Trong một số trường hợp Linux sẽ hỏi tên hệ thống khi cài đặt, nhưng người sử dụng hoàn toàn có thể thay đổi sau đó.

Tên được sử dụng để xác định một hệ thống Linux được gọi là *hostname*. Việc đặt tên cho phép sử dụng hệ thống trên mạng cũng như các dịch vụ liên quan khác như email.... Có thể sử dụng lệnh sau để hiển thị tên hiện tại của hệ thống :

```
$ hostname
$ may1
```

Lệnh này hiển thị rằng tên hệ thống là *may1*. Nếu hệ thống chưa được đặt *hostname*, Linux sẽ trả lời mặc định là không có tên hoặc một tên mặc định của hệ thống. Thông tin về tên được đọc trong các tệp khởi động của Linux.

Nếu hệ thống không nối mạng thì ta có thể đặt cho nó tên bất kì. Lệnh *hostname* với tham số -S được sử dụng để đặt tên cho một hệ thống, ví dụ

```
hostname -S camap
```

Lệnh này đặt cho hệ thống tên camap. Tên này sẽ được gắn vào mọi thư điện tử của chúng ta và một số tiện ích hệ thống có sinh ra thông tin đầu ra. Một số hệ thống Linux giới hạn độ dài tên máy, thường giới hạn này là 14.

1.1. Đặt tên cho hệ thống

Như đã nói, trong mạng mỗi máy tính phải có một tên duy nhất xác định nó. Nếu không mạng không thể xác định được phải chuyển thông tin cho máy nào trong số các máy trùng tên. Nếu hệ thống nằm trong một mạng cục bộ không nối với Internet hoặc không có một tên mạng chính thức thì người sử dụng có thể lấy bất kì tên mạng mong muốn nào. Tổ hợp tên máy và tên mạng tạo thành tên đầy đủ cho máy. Ví dụ

```
$ hostname -S camap.dongvat
```

bao gồm tên máy camap và tên mạng là dongvat. Miễn là các máy trong mạng có cùng tên mạng thì các máy vẫn có thể giao tiếp tốt với nhau, các máy vẫn được xác định nhờ tổ hợp tên máy và tên mạng.

Nếu hệ thống có thể truy cập Internet thì mạng sẽ được Trung tâm thông tin mạng Internet NIC (Internet Network Information Center) gán các tên, gọi là miền, tuân theo quy ước đặt tên nghiêm ngặt. Mỗi miền có một tên thành phần duy nhất và phần mở rộng xác định kiểu tổ chức mà mạng thuộc vào. Ví dụ trường đại học Bách Khoa Hà Nội có tên miền là hut.edu.vn. Nếu muốn tên máy là **fit10** thì ta gõ lệnh :

```
$ hostname -S fit10.hut.edu.vn
```

Việc tên hệ thống sẽ được đề cập kỹ hơn trong phần quản lý và thiết lập cấu hình mạng.

1.2. Lưu trữ tên máy

Linux lưu tên của trạm làm việc trong tệp */etc/hosts* (hoặc tệp */etc/rc* hoặc */etc/rc.local* hoặc nằm trong thư mục */etc/rc.d*, tuỳ từng hệ thống Linux). Khi mới cài đặt Linux */etc/hosts* bao gồm một số dòng chú thích và một dòng mã :

```
127.0.0.1 localhost
```

Tệp */etc/hosts* gồm hai cột, một cho địa chỉ IP và cột thứ hai cho tên máy. Địa chỉ IP được viết dưới dạng ký pháp thập phân có chấm. Trong đó IP - Internet Protocol - là thành phần thiết yếu của giao thức mạng *TCP/IP* được dùng trên Internet và hầu hết các mạng cục bộ liên quan đến UNIX. Địa chỉ IP của máy nối với Internet được gán bởi NIC, cũng giống như tên domain. (Địa chỉ IP và tên miền được ánh xạ với nhau do đó mạng có thể dùng số thay vì tên). Nếu hệ thống không được nối với Internet thì máy có thể có địa chỉ IP bất kỳ.

Địa chỉ IP gồm định danh mạng và định danh máy. Bốn phần của địa chỉ IP được phân chia thành hai định danh trên theo cách đặc biệt. Địa chỉ 127.0.0.1 là một địa chỉ đặc biệt gọi là địa chỉ nối vòng (loopback address), nó cho phép máy tự kết nối với chính mình. Mọi máy đều có thiết bị nối vòng, nó được định danh là 127.0.0.1 trong */etc/hosts* dưới tên *localhost*.

Nếu máy trạm đã có tên, tên đó nằm trong tệp */etc/hosts*. Ví dụ :

```
127.0.0.1 camap localhost
```

Dòng này báo với hệ thống rằng máy này là máy địa phương và camap là định danh của nó.

Qúa trình gán tên phức tạp hơn một chút khi hệ thống nằm trong mạng, mỗi máy trong mạng chỉ có một địa chỉ IP duy nhất. Nếu bạn có kết nối Internet thì địa chỉ IP mạng đã có sẵn, người quản trị sẽ cung cấp cho bạn địa chỉ IP của máy hoặc bạn có thể chọn một địa chỉ chưa dùng đến.

Chẳng hạn, khi hệ thống nối với Internet với địa chỉ IP là 47.123.23.37 và tên miền là quacks.com. Tệp */etc/hosts* sẽ như sau :

```
127.0.0.1 localhost
47.123.23.37 camap.dongvat.com
```

Tên camap có thể xuất hiện trong dòng *localhost* hoặc không, không bắt buộc. Tệp này có thể gồm các dòng khác khi chúng ta được nối với một mạng lớn mà chúng ta thường xuyên di chuyển xung quanh, nhưng ít nhất là có hai dòng trên

2. Quyền truy cập tệp tin và thư mục

Linux quản lí truy nhập các tệp tin và thư mục trong hệ thống tệp thông qua khối quyền. Khối quyền là một phần của điểm nhập tương ứng với mỗi tệp hoặc thư mục của bảng *i-node*. Chúng ta có thể hiển thị khối quyền cho một tệp hoặc thư

mục bằng cách hiển thị danh sách tệp. Ví dụ :

```
$ls - la
total 2
drwxrwx--- 25  svtt    svtt      4096    Fab 20 1999 svtt
-rw-r--r--   1   nvviet   parallel   324102   Mar  15  1999
Figure.fig
```

Cột đầu tiên của danh sách là khối quyền gồm 10 kí tự. Mỗi tệp và thư mục đều có một khối quyền gắn với nó. Khối quyền được tạo thành bởi hai kiểu thông tin. Kí tự thứ nhất là kí tự chỉ kiểu tệp, còn 9 kí tự tiếp theo là các quyền truy nhập. Các phần sau đây sẽ xem xét hai kiểu thông tin kĩ hơn đôi chút.

2.1. Các kiểu tệp

Linux dùng kí tự đầu tiên của khối quyền để chỉ kiểu tệp của một điểm nhập trong bảng *i-node*. Vì Linux không có sự phân biệt giữa tệp và thư mục trong bảng *i-node*, nên kí tự này là cách duy nhất để hệ điều hành biết điểm nhập tham chiếu đến nó là một tệp thông thường hay một thư mục. Trong bảng *i-node* thư mục và tệp rất giống nhau.

Linux hỗ trợ một số kiểu tệp, mỗi kiểu có một kí tự được dùng làm kí tự đầu tiên trong khối quyền. Các kí tự kiểu tệp chung nhất mà Linux dùng được chỉ trong bảng 1.1.

Một số phiên bản khác của Linux và Unix hỗ trợ các kiểu tệp khác (như s cho kiểu đặc biệt), nhưng các kiểu này ít được dùng và không quan hệ đến các quyền trên tệp.

Bảng 1.1. Các kiểu tệp trong Linux

-	Tệp thông thường
b	Thiết bị với chế độ truy nhập theo khối
c	Thiết bị với chế độ truy nhập theo kí tự
d	Thư mục
l	Liên kết (link)

Hầu hết các tệp trong Linux đều là các tệp thông thường, tức là tệp có thể chứa dữ liệu, một ứng dụng, tệp văn bản, hoặc bất kì tệp nào chứa thông tin (người dùng có thể đọc được trực tiếp hoặc không). Chúng được chỉ bằng dấu "-" trong

khối kiểu. Mọi tệp mà người dùng tạo đều là tệp thông thường.

Hai kiểu b, c là các kiểu tệp thiết bị (vì mọi thiết bị đều được UNIX coi là tệp), chúng gồm các lệnh cho phép Linux nói chuyện với các thiết bị ngoại vi. Hầu hết chúng được lưu trong thư mục */dev*, nhưng cũng có thể ở bất cứ đâu trong hệ thống tệp. Kiểu b xác định cách ghi và đọc trên thiết bị theo từng khối dữ liệu, kiểu c xác định cách ghi, đọc trên thiết bị theo từng kí tự.

Kiểu thư mục chỉ ra rằng điểm nhập trong bảng *i-node* tham chiếu đến một thư mục chứ không phải một tệp.

Liên kết (chỉ đến một tệp khác) được xác định trong kí tự kiểu tệp bằng kí tự "l", không phải mọi hệ điều hành đều hỗ trợ kí tự này. Nếu một phiên bản của Linux không dùng kiểu tệp "l" để chỉ một liên kết, ta sẽ căn cứ vào cột thứ hai của đầu ra của lệnh hiển thị danh sách tệp, con số này chỉ số liên kết mà điểm nhập này có.

2.2. Các quyền truy nhập

Mọi hệ thống UNIX điều khiển truy nhập tệp và thư mục thông qua việc đọc các quyền trên khối quyền. Quyền truy nhập tệp hoặc thư mục có thể gồm một trong ba kiểu giá trị sau :

Bảng 1.2. Quyền truy cập tệp và thư mục

r	Đọc (read)
w	Ghi (write)
x	Thực thi (execute)

Nếu có quyền đọc ta có thể đọc được nội dung của tệp (sử dụng lệnh *cat* chẳng hạn). Nếu có quyền ghi vào một tệp, ta có thể sửa nội dung của tệp và ghi các thay đổi này lên tệp. Nếu có quyền thực thi, ta có thể thực thi tệp đó, giả thiết rằng đó là tệp nhị phân hoặc một shell script. Nếu đó là tệp ASCII và ta thực hiện nó, thì chỉ thu được các thông báo lỗi mà thôi.

Giá trị của ba quyền này tạo thành một khối 3 kí tự theo thứ tự *rwe* cho đọc, viết, thực thi. Nếu một quyền được đặt là không được phép thì dấu gạch ngang được thay vào vị trí quyền đó để thể hiện là không có quyền đó. Ví dụ *r-x* chỉ ra là tệp này có quyền đọc, thực thi nhưng không có quyền ghi, hay --- chỉ ra rằng tệp

này không có quyền truy nhập nào cả.

Các quyền này cũng được sử dụng cho thư mục dù rằng ý nghĩa của nó có hơi khác. Có quyền đọc một thư mục tức là có thể hiển thị nội dung của thư mục (sử dụng lệnh *ls* chẳng hạn). Quyền ghi có nghĩa là có thể thêm các tệp vào thư mục. Quyền thực thi thể hiện có thể chuyển đến thư mục đó (bằng lệnh *cd* chẳng hạn). Ví dụ khối quyền *r-x* của một thư mục có nghĩa là có thể hiển thị nội dung, chuyển đến thư mục, nhưng không thể thêm tệp mới vào thư mục.

Ba quyền này được đặt cho mỗi một trong ba mức truy nhập khác nhau. Đó là :

- Khối quyền cho người chủ tệp.
- Khối quyền nhóm, tức là cho những người thuộc cùng nhóm với chủ tệp.
- Khối quyền cho người khác, tức là cho mọi người khác trong hệ thống.

Các khối 3 kí tự cho quyền đọc - ghi - thực thi này được tổ hợp cho 3 đối tượng : chủ tệp, nhóm, người khác (*user*, *group*, *other*) tạo thành khối quyền 9 kí tự như nhìn thấy khi hiển thị danh sách tệp.

Khi đã quen với các khái niệm chủ, nhóm, người khác ta có thể dễ dàng đọc các khối quyền của tệp. Ví dụ :

```
rw-r--r--
```

có nghĩa chủ tệp có quyền đọc và ghi, những người cùng nhóm với chủ tệp (khối ba kí tự tiếp theo) chỉ có quyền đọc, mọi người khác trong hệ thống cũng chỉ có quyền đọc.

Ý nghĩa quyền sử dụng thư mục cũng vậy. Ví dụ, một thư mục có khối quyền như sau :

```
drwxr-xr-x
```

Có nghĩa người chủ thư mục có quyền hiển thị nội dung thư mục, thêm tệp vào thư mục và chuyển vào thư mục, mọi người khác trong hệ thống, trong cùng nhóm với người chủ và mọi người khác, có thể hiển thị nội dung thư mục và chuyển vào thư mục, nhưng không thể thêm tệp vào thư mục.

2.3. Các quyền mặc định

Khi ghi hoặc tạo tệp hoặc một thư mục mới, các quyền mặc định được gán cho chúng. Các quyền này được UNIX đặt dựa trên mặt nạ tạo tệp gọi là *umask*.

Mỗi người sử dụng trong hệ thống có một *umask*, hoặc là được họ đặt trong các tệp khởi tạo *(.profile, .cshrc, ...)* hoặc *umask* mặc định của hệ thống khi người dùng đăng nhập hệ thống.

Ta có thể hiển thị giá trị hiện thời của *umask* bằng cách gõ lệnh *umask* tại dấu nhắc của shell :

```
$ umask
022
```

Khối 3 được lệnh *umask* trả về là giá trị *umask* hiện tại. Ba số này biểu diễn các quyền đọc - ghi - thực thi mà ta thấy trong khối quyền của tệp. Các con số đó có ý nghĩa như sau:

Bảng 1.3. Các nhóm quyền trên tệp và thư mục

0	Đọc và ghi (và thực thi cho thư mục)
1	Đọc và ghi (không thực thi cho thư mục)
2	Đọc (và thực thi cho thư mục)
3	Đọc
4	Ghi (và thực thi cho thư mục)
5	Ghi
6	Thực thi
7	Không có quyền

Với danh sách trên, ta thấy *umask 022* có nghĩa, người dùng có quyền đọc, ghi cho tệp của họ (0), nhóm của người dùng có quyền đọc (2 đầu tiên), mọi người khác có quyền đọc (2 thứ hai). Khi người dùng sử dụng umask này để tạo một tệp, khối quyền như sau:

```
rw-r--r—
```

Để thay đổi *umask* có thể dùng lệnh hoặc sửa trong các tệp khởi tạo. Bằng cách dùng lệnh, ta dùng lệnh *umask* với bộ ba số quyền truy nhập mà ta muốn. Ví dụ

```
$ umask 077
```

đặt cho người chủ quyền đọc và ghi, mọi người khác không có quyền gì cả. Giá trị *umask*. Như vậy, rất thuận tiện cho việc hạn chế các truy nhập tệp.

Nếu muốn thay đổi *umask* tạm thời, gõ lệnh *umask* và các thiết lập mới tại dấu nhắc của shell. Các giá trị mới có giá trị cho đến khi đổi nó lần nữa. Nếu muốn thay đổi lâu dài *umask*, thêm dòng lệnh trên vào tệp khởi tạo shell *(.profile, .cshrc,...)*.

2.4. Thay đổi các quyền

Ta có thể thay đổi quyền được gắn với tệp hoặc thư mục bằng lệnh *chmod*, có thể thực hiện lệnh này theo chế độ dùng biểu tượng hoặc tuyệt đối. Chế độ biểu tượng đơn giản hơn nhưng chế độ tuyệt đối cho phép điều khiển tốt hơn.

Trong chế độ biểu tượng cần tuân theo cú pháp chặt chẽ. Cú pháp chung là

```
$ chmod who-change-perms files
```

trong đó *who* chỉ đối tượng mà ta muốn việc thay đổi được áp dụng, gồm có u cho người sở hữu tệp, *g* cho nhóm và o cho những người khác. *Change* chỉ ra rằng ta muốn bỏ đi các quyền (-), thêm các quyền (+) hoặc đặt chúng một cách tường minh (=). Ta chỉ có thể dùng một biểu tượng của change trong mỗi lệnh *chmod*. *Perms* chỉ ra, ta muốn thay đổi quyền đọc *(r)*, ghi (*w*) hay thực thi (*x*). Ba thành phần *who, change, perms* nằm liền nhau, không có khoảng cách. Để sáng tỏ ta xem một vài ví dụ :

```
$ chmod u+rwx bigfile
```

thêm quyền đọc, ghi, thực thi cho chủ tệp. Nếu một trong các quyền này đã được đặt thì chúng vẫn được giữ nguyên, nếu chưa có thì được thêm vào. Các quyền trên nhóm và của người khác không bị ảnh hưởng, lệnh chỉ làm việc trên các quyền của chủ tệp.

Trong khi đó lệnh

```
$ chmod go-x bigfile
```

Bỏ quyền thực thi của nhóm và người khác mà không thay đổi quyền đọc, ghi trên hai đối tượng này và cả các quyền của chủ tệp trên tệp `bigfile`.

Lệnh

```
$ chmod uo+w chapter*
```

Thêm quyền ghi cho chủ và những người dùng khác cho mọi tệp bắt đầu bằng `chapter`.

Nếu không chỉ rõ là lệnh được áp dụng trên chủ, nhóm, người khác thì cả ba

đều bị ảnh hưởng. Ví dụ:

```
$ chmod +rwx
```

Thêm quyền đọc, ghi và thực thi cho cả chủ, nhóm và người khác.

Ta cũng có thể dùng các lệnh tương tự để đặt quyền cho thư mục. Ví dụ lệnh :

```
$ chmod go+rx mydir
```

Cho phép những người dùng trong nhóm và những người khác quyền liệt kê nội dung thư mục và chuyển đến thư mục *mydir*, nhưng không thể thêm tệp vào thư mục này.

Đôi khi ta có thể muốn đặt quyền một cách tường minh cho một khối quyền, khi đó ta dùng dấu bằng. Ví dụ lệnh :

```
$ chmod u=rx bigfile
```

Đặt quyền đọc và thực thi cho *user* và bỏ quyền ghi mà không quan tâm đến trước đó quyền ghi có được đặt hay không. Tuy vậy, khối quyền của nhóm và người khác không bị ảnh hưởng.

Nếu muốn thay đổi cả ba khối quyền cùng một lúc, ta dùng *chmod* trong chế độ tuyệt đối. Chế độ này sử dụng các số để chỉ các quyền, có 3 số một cho chủ tệp, một cho nhóm, một cho người khác. Cả 3 được chỉ ra trên cùng một dòng lệnh. Mỗi số là tổng của các giá trị thể hiện quyền đọc, ghi, thực thi sau đây:

Bảng 1.4. Các thuộc tính của một tệp

000	không có quyền
001	người khác có quyền thực thi
002	người khác có quyền ghi
004	người khác có quyền đọc
010	mọi người trong nhóm có quyền thực thi
020	mọi người trong nhóm có quyền ghi
040	mọi người trong nhóm có quyền đọc
100	chủ tệp có quyền thực thi
200	chủ tệp có quyền ghi
400	chủ tệp có quyền đọc

Ta thấy các chữ số được chia thành 3 cột, từ trái sang phải thể hiện quyền của chủ, nhóm và người khác. Để sử dụng các số này, ta cộng các giá trị lại với nhau 1(thực thi), 2 (ghi), 4 (đọc) để được tổ hợp mong muốn. Các giá trị này nếu được viết dưới dạng nhị phân và đặt tương ứng với khối quyền *rwx* sẽ có giá trị 1 tại vị trí tương ứng với quyền được phép, và 0 tại các quyền không được phép. Ta sử dụng các số này trong lệnh *chmod*. Ví dụ :

```
$ chmod 644 bigfile
```

cho chủ quyền đọc và ghi (giá trị 6), nhóm quyền đọc (4), người khác quyền đọc (4). Các quyền không được đặt sẽ bị đặt là không được phép. Kết quả là khối quyền của tệp như sau :

```
rw-r--r--
```

Ta có thể thấy nó giống với khối quyền mặc định của chủ với *umask 022*. Điều này cho thấy sơ đồ các số trong *umask* và *chmod* không giống nhau.

Sử dụng chế độ *chmod* nào phụ thuộc và kiểu thay đổi quyền mà ta muốn. Nếu chỉ muốn thay đổi một quyền (như thêm quyền thực thi cho chính mình hoặc quyền đọc ghi cho nhóm), dạng biểu tượng thuận tiện hơn. Để đặt khối quyền chi tiết, dùng dạng tuyệt đối là nhanh nhất.

2.5. Thay đổi chủ và nhóm

Mọi tệp hay thư mục trong Linux đều có một chủ (*owner*) và một nhóm, có thể thấy chúng khi liệt kê tệp. Chủ của tệp thường biểu diễn là tên người dùng (*username*) của người tạo tệp, và nhóm là nhóm của người chủ khi tạo tệp. Ta có thể thay đổi chủ và nhóm khi chia sẻ tệp hoặc chuyển chúng cho người khác. Để là điều này dùng lệnh *chown* và *chgrp*.

Để thay đổi chủ tệp hoặc thư mục, dùng lệnh *chown* với tên người chủ mới. Ví dụ:

```
$ chown dungkh datafile
```

đổi chủ của `datafile` thành `dungkh`. Khi đưa ra lệnh này Linux kiểm tra xem người chủ mới đưa ra có hợp lệ không (tìm trong */etc/passwd*), và xem người thực hiện lệnh có thực sự sở hữu tệp này không vì chỉ có root và chủ tệp mới có quyền đổi chủ cho nó. Cũng có thể dùng kí hiệu thay thế để đổi chủ cho nhiều tệp hay thư mục trong một lần. Ví dụ :

```
$ chown chinq chapter
```

đổi chủ của tất cả các tệp bắt đầu bằng `chapter` sang cho `chinq`.

Để đổi nhóm sở hữu một tệp hoặc thư mục, dùng *chgrp*. Ví dụ :

```
$ chgrp svtt bigfile
```

Đổi nhóm của `bigfile` sang `svtt`. Một lần nữa Linux kiểm tra tên nhóm trong */etc/group* và người thực hiện chuyển nhóm có ở trong nhóm hiện đang sở hữu tệp không. Giống như *chown* ta cũng có thể dùng các kí tự thay thế cho lệnh này.

Nếu biết UID hoặc GID của người dùng hoặc nhóm mới, ta có thể dùng chúng trong dòng lệnh thay vì tên. Linux cũng tìm trong */etc/paswd* và */etc/group* xem UID hay GID có hợp lệ không, đồng thời ta cũng phải có quyền chuyển đổi chủ.

Chú ý rằng khi đổi chủ, ta dễ dàng đổi chủ hoặc nhóm và khi đó ta đã tự khoá mình ở ngoài tệp.

III. QUẢN LÍ NGƯỜI DÙNG

Mọi truy nhập vào hệ thống Linux đều thông qua một tài khoản của người sử dụng. Mỗi tài khoản được thiết lập bởi người quản trị hệ thống ngoại trừ tài khoản root (và một số tài khoản hệ thống mà người sử dụng ít khi dùng tới). Mặc dù một số hệ Linux chỉ có một người dùng nhưng cũng không nên dùng tài khoản root cho các truy nhập hàng ngày. Hầu hết các hệ thống đều cho phép nhiều người truy trong *console* chính, qua *modem*, mạng, hoặc các thiết bị đầu cuối được nối cứng. Vì vậy, phương pháp đặt và quản lí các tài khoản, các thư mục cũng như các tệp liên quan đến nó là một khía cạnh quan trọng của quản trị hệ thống Linux.

Phần này sẽ xem xét tài khoản root, tài khoản mạnh nhất, từ đó xem một số khía cạnh của việc thiết lập tài khoản mới trong Linux, đồng thời cả các nhóm trong Linux.

1. Tài khoản của người quản trị

Khi cài đặt phần mềm Linux, một tài khoản đặc biệt được tạo tự động, đó là root, (đôi khi được gọi là siêu người dùng vì khi đăng nhập vào tài khoản này

người sử dụng có quyền sử dụng tuyệt đối trên hệ thống). Các tài khoản người sử dụng của Linux đều bị hạn chế một số quyền để tránh người dùng chẳng may làm hỏng hệ thống

Khả năng của tài khoản root là không giới hạn. Khi đăng nhập với tài khoản root người sử dụng có thể thêm các quyền mới cho nó. Chính vì root có sức mạnh như vậy cho nên nó rất hấp dẫn đối với người mới sử dụng, và khi sử dụng hệ thống với tài khoản này họ rất dễ gây rối loạn hệ thống. Do vậy, chỉ nên sử dụng root để bảo trì hệ thống chứ không nên sử dụng nó cho các mục đích hàng ngày.

Chúng ta cũng có thể tạo một số tài khoản đặc biệt mà không cần nhiều quyền truy nhập cho quản trị hệ thống. Chẳng hạn để cho sao lưu ra băng từ, ta có thể thiết lập một tài khoản chỉ có quyền đọc của root tới toàn bộ hệ thống tệp mà không có quyền ghi để tránh gây hỏng hóc vô tình đến hệ thống tệp như chẳng may xoá mất hạt nhân. Có thể thiết lập một số tài khoản tương tự cho truy nhập e-mail, gateway vào Internet,...

Một điều quan trọng là không phải tài khoản siêu người dùng nào cũng gọi là root, mặc dù nó được tạo mặc định là root khi cài đặt Linux. Nó có thể có tên bất kì nhưng thường được dùng nhất dưới tên root. Tài khoản này được định nghĩa là tài khoản có *userID* là 0, các *userID* được định nghĩa trong */etc/passwd*.

3. Thiết lập tài khoản người sử dụng

Để hệ thống có thể được nhiều người sử dụng ta phải tạo tài khoản cho mỗi người trong số họ hoặc một tài khoản khách (guest) cho nhiều người sử dụng mà không cần mật khẩu.

Mỗi người dùng Linux phải có một tên sử dụng, một mật mã riêng, trừ tài khoản guest hoặc tài khoản truy nhập một ứng dụng đặc biệt như cơ sở dữ liệu. Bằng cách cấp cho mỗi người sử dụng một tài khoản riêng, mức độ bảo mật của hệ thống sẽ được nâng lên và người quản trị hệ thống có thể biết rõ ai truy nhập hệ thống và đang làm gì. Ánh xạ một người dùng và một tài khoản cho phép theo dõi các hoạt động dễ hơn nhiều.

Tệp */etc/passwd* chứa các thông tin về tài khoản của người sử dụng, nên được đặt dưới quyền sở hữu của root và *groupID* đặt bằng 0 (0 thường dùng để chỉ root hoặc nhóm hệ thống, được định nghĩa trong tệp */etc/group*). Nên đặt quyền để chỉ root có quyền ghi còn người dùng khác chỉ có quyền đọc.

Các dòng trong tệp */etc/passwd* đều có khuôn dạng :

```
username:password:userID:groupID:comment:home directory:login
command
```

Ví dụ một phần tệp */etc/passwd* được tạo khi Linux mới được cài đặt.

```
root::0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
operator:*:11:0:operator:/root:/bin/bash
games:*:12:100:games:/usr/games:
nobody:*:-1:100:nobody:/dev/null:
ftp:*:404:1::/home/ftp:/bin/bash
```

Mỗi dòng đều có 7 trường phân cách nhau bởi dấu ":". Nếu 1 trường không có giá trị thì trường đó bị bỏ trắng. Các trường lần lượt có ý nghĩa như sau :

username :	Tên duy nhất xác định một người sử dụng
password :	Mật khẩu của người sử dụng đã được mã hoá
user ID (UID):	Số duy nhất xác định một người đối với hệ điều hành.
GroupID (GID) :	Số duy nhất xác định nhóm của người dùng được xác định trong dòng này (phục vụ cho đặt quyền trên tệp).
Comment :	Các thông tin ghi chú, thường dùng là tên thật của người sử dụng, nhưng có thể là thông tin khác.
Home directory :	Thư mục mà người sử dụng sẽ được đặt vào khi họ đăng nhập (*login*) vào hệ thống.
Login command :	Lệnh sẽ thực hiện ngay sau khi người sử dụng đăng nhập vào hệ thống, thông thường là lệnh gọi chương trình shell

Bây giờ ta sẽ tìm hiểu chi tiết về từng trường và các chương trình sử dụng các trường này như thế nào trong Linux. Tệp kiểu này được sử dụng trong hầu hết các hệ Unix, do đó các hiểu biết về nó trên Linux cũng sẽ cho ta các hiểu biết trên hầu hết các hệ Unix khác.

2.1. Tên người dùng (Username)

Tên người dùng có thể là một xâu kí tự, thường là 8 hoặc nhỏ hơn, xác định duy nhất một người dùng. Tên này là phương tiện thường dùng nhất để giao tiếp giữa các người dùng với nhau và với các máy khác, vì vậy nên đơn giản, dễ đoán. Có thể dùng bất kì cách đặt tên nào cho tên người dùng của mình, miễn là đảm bảo tên đó là duy nhất trong mạng, nhưng đôi khi các tên độc đáo lại làm cho những người dùng khác, máy khác không biết họ là ai và gây khó khăn cho giao tiếp. Các tên quá đơn giản như kiểu chỉ gồm tên thật không có họ, lại dễ bị trùng trong các hệ thống mạng lớn. Thông thường tên này là một hoán vị của tên thật của người dùng, thường là tổ hợp của tên viết tắt và họ đối với người nước ngoài và họ viết tắt cộng với tên đối với người Việt Nam. Ví dụ Tomat Parker sẽ có tên người dùng: `tiennc`, hay Nguyễn Văn Việt có tên người dùng là `nvviet`.

Linux cũng như mọi hệ Unix khác có phân biệt chữ hoa và chữ thường, vì vậy các tên người dùng viết hoa và viết thường sẽ được hiểu là hai tên khác nhau. Vì hầu hết các lệnh của Linux đều được viết bằng chữ thường nên chúng ta nên để tên người dùng ở dạng chữ thường. Các kí tự gạch dưới, cách, số, và một vài kí tự đặc biệt có thể được dùng trong tên người dùng nhưng lại làm cho tên trông có vẻ lạ và đôi khi gây rắc rối cho một vài ứng dụng.

2.2. Mật khẩu (Password)

Hệ thống lưu mật khẩu đã được mã hoá của người dùng vào trường *password*. Chỉ người quản trị hệ thống mới có thể đổi mật khẩu, hoặc chính người sử dụng tự đổi mật khẩu của mình bằng lệnh *passwd*. Mọi thay đổi trực tiếp khác trên trường này đều có thể làm cho tài khoản tương ứng không thể sử dụng được nữa.

Một số phiên bản Unix không để mật khẩu đã mã hoá trong tệp */etc/passwd* vì vấn đề bảo mật, trong trường *passwd* ta chỉ thấy thấy chữ *x*, mà để trong tệp */etc/shadows*. Nói chung trường này vẫn được để theo mặc định.

Khi người sử dụng đăng nhập vào hệ thống, chương trình *login* mã hoá mật khẩu được nhập, đem so sánh kết quả với trường *password* của người đó. Đăng nhập sẽ không thực hiện được nếu có sự khác nhau.

Nếu kí tự đầu tiên của trường *password* là "*" thì có nghĩa là tài khoản này không được sử dụng để truy nhập hệ thống, nó hạn chế mọi truy nhập. Như thấy trong tệp */etc/passwd* ở trên, có rất nhiều tài khoản có trường *password* là dấu *.

Nếu trường *password* bị bỏ trắng thì mọi người sử dụng đều có thể đăng nhập tài khoản này một cách không hạn chế mà không yêu cầu mật khẩu, chẳng hạn như tài khoản guest.

2.3. Định danh người sử dụng (UserID)

Mỗi người sử dụng được tương ứng với một số hiệu duy nhất gọi là UID, dùng để xác định mọi thứ có liên quan đến người sử dụng đó. Chẳng hạn Linux sử dụng định danh này để theo dõi mọi tiến trình do một người sử dụng kích hoạt. Định danh người sử dụng thường được sử dụng nhiều hơn tên vì chúng chỉ gồm những con số, dễ làm việc hơn là chuỗi kí tự tên người sử dụng, và cũng tốn ít không gian lưu trữ hơn. Có một số tiện ích cho phép chuyển đổi từ định danh người sử dụng sang tên bằng cách duyệt tệp */etc/passwd* và tìm đến UID phù hợp để lấy tên.

Hầu hết các hệ Unix dùng các số từ 0 đến 99 cho các tài khoản dành cho hệ thống, và từ 100 trở đi dành cho người sử dụng. Như ta thấy trong tệp ví dụ ở trên root có UID bằng 0, các tài khoản khác của hệ thống dùng các số lớn hơn. Đặc biệt, *nobody* là một tài khoản đặc biệt chỉ dùng cho NFS (Network File System) có UID -1. Khi gán số hiệu cho người sử dụng, chúng ta gán lần lượt từ 100, 101,...

2.4. Số hiệu nhóm (GroupID)

Số hiệu nhóm được dùng để xác định nhóm mà người sử dụng thuộc vào khi đăng nhập vào hệ thống. Nó dùng cho mục đích đặt quyền trên tệp. Số hiệu nhóm là các số bắt đầu từ 0, đa số các hệ thống Unix dùng các số từ 0 đến 49 (hoặc 0 đến 9) cho các nhóm hệ thống và từ 50 trở đi cho người dùng. Nhóm mặc định là 50.

Các nhóm và số hiệu nhóm được liệt kê trong tệp */etc/group*.

2.5. Thông tin ghi chú (Comment)

Người quản trị hệ thống dùng trường này để cung cấp thêm thông tin cần thiết cho mỗi tài khoản tương ứng trong */etc/passwd*, làm cho chúng trở nên dễ hiểu hơn. Nó thường chứa tên đầy đủ của người sử dụng, nhưng đôi khi là phòng ban hoặc một con số mở rộng nào đó tuỳ theo người quản trị. (Trường này đôi khi được gọi là GECOS, General Electric Comprehensive Operating System, tuỳ thuộc vào hệ điều hành dùng nó).

Một số tiện ích dùng trường này để hiển thị thông tin về người sử dụng. Chẳng hạn hệ thống e-mail truy nhập vào nó để đưa ra ai là người gửi thư đi. Do vậy, không nên để các thông tin nhạy cảm với việc sửa đổi ở đây.

2.6. Thư mục cá nhân (Home Directory)

Trường này chỉ cho tiến trình *login* nơi đặt người sử dụng khi họ đăng nhập vào hệ thống (tức là thư mục làm việc khởi đầu của người đó), thường là thư mục cá nhân của người đó. Mỗi người dùng nên có một thư mục cá nhân và tệp khởi động của họ sẽ khởi tạo cho biến môi trường *HOME* giá trị thư mục này. Người sử dụng không bị hạn chế gì trong thư mục chỉ ra trong trường *home directory* ngoại trừ một số quyền trên tệp được đặt để tránh di chuyển.

Nhìn chung thư mục cá nhân của người sử dụng được đặt trong một vùng chung. Unix có xu hướng dùng thư mục */home*, do đó người sử dụng có tài khoản nvviet sẽ có thư mục cá nhân */home/nvviet*. Một số phiên bản khác của Unix dùng thư mục */usr*, */user* hoặc */u* để chứa các thư mục cá nhân, hoặc một số nhà quản trị hệ thống lại thích dùng cấu trúc thư mục khác để thuận tiện cho họ. Linux không quan tâm lắm đến vấn đề này, chỉ cần thư mục đó có thể vào được.

2.7. Lệnh đăng nhập (Login command)

Trường này chỉ lệnh được thực hiện ngay khi chương trình login kết thúc. Lệnh này thường là một lệnh shell kích hoạt một chương trình C shell hoặc Bourne shell nhằm cung cấp cho người dùng một môi trường shell. Trong một số trường hợp nó có thể là một ứng dụng đơn giản hoặc một ứng dụng đầy đủ của hệ thống để hạn chế những gì người dùng có thể thực hiện. Ví dụ tài khoản *uucp* (sử dụng cho e-mail hoặc cho các công việc về mạng đơn giản khác) chỉ thực hiện lệnh *uucp*. Nếu trường này rỗng, hệ điều hành mặc định lệnh là lời gọi Bourne shell (mặc định này cũng tuỳ theo hệ điều hành được cài đặt).

Nhiều phiên bản của Linux cho phép người dùng thay đổi shell khởi đầu (sau khi vào hệ thống) bằng lệnh *chsh* hoặc *passwd -s*. Khi nhận lệnh này Linux tìm trong tệp */etc/shell s* đến một shell phù hợp. Chỉ các lệnh trong */etc/shells* mới là những lệnh mà người dùng được phép dùng thay thế cho lệnh gọi shell khởi đầu được đặc tả trong trường login command. Điều này làm cho hệ thống chặt chẽ hơn về mặt an toàn. Các dòng trong */etc/shells* có thể thêm hoặc bớt đổi bằng các trình

soạn thảo. Chúng ta nên để cho tệp này có cùng quyền truy nhập và cùng người sở hữu như */etc/passwd* để tránh người dùng sửa lệnh khởi đầu khi họ đăng nhập vào hệ thống.

3. Một số tên người dùng mặc định của hệ thống

Như thấy trong tệp */etc/passwd*, có rất nhiều tên người dùng trong hệ thống, một số trong đó được dùng cho hệ điều hành và người quản trị.

Tài khoản root:	Tài khoản siêu người dùng (UID 0) không bị hạn chế quyền truy nhập và là chủ của nhiều tệp hệ thống.
Tài khoản daemon:	Được sử dụng cho các tiến trình hệ thống, để làm chủ các tiến trình và đặt các quyền của chúng một cách chính xác.
Tài khoản bin:	Làm chủ các tệp thực thi được.
Tài khoản sys:	Làm chủ các tệp thực thi được.
Tài khoản adm:	Làm chủ các tệp sổ sách (các tệp ghi lại thông tin hoạt động, các quá trình của hệ thống) và tệp log.
Tài khoản uucp:	Dùng cho các giao tiếp và tệp UUCP.

Ngoài ra còn có một số tài khoản khác dùng cho mục đích riêng như postmaster cho e-mail... Ta không nên thay đổi bất kì tài khoản nào của hệ thống. Nói chung, chúng đều có trường password là dấu * để tránh truy nhập.

4. Thêm người dùng

Ta có thể thêm người sử dụng hệ thống bằng cách sửa tệp */etc/passwd* bằng tay hoặc sử dụng một script tự động, nó sẽ nhắc các thông tin chi tiết về người dùng mới phải nhập và ghi dòng thông tin mới vào tệp */etc/passwd*. Script này làm cho công việc thêm người dùng trở nên dễ dàng hơn và tránh được các lỗi gặp phải khi phải thêm nhiều người dùng bằng tay. Như chúng ta đã biết, ta chỉ có thể thay đổi */etc/passwd* khi có quyền root vì vậy phải đăng nhập dưới tên root. Để cho an toàn, hãy sao lưu */etc/passwd* trước khi sửa đổi nó, nếu không khi vô tình làm hỏng tệp này chúng ta không thể vào lại hệ thống kể cả từ tài khoản root, và hệ thống chỉ có thể làm việc trong chế độ quản trị hệ thống.

Để thực hiện bằng tay, quá trình thêm người sử dụng gồm có :

- Thêm dòng tương ứng với người mới vào tệp */etc/passwd*.
- Tạo thư mục cá nhân cho người này (tất nhiên trùng với thư mục cá nhân đã chỉ ra trong */etc/passwd*).
- Chép các tệp khởi động shell và sửa đổi các thiết lập và quyền sở hữu các tệp đó.

Đầu tiên, để thêm một dòng của người sử dụng mới vào tệp */etc/passwd*, ta phải dùng một trình soạn thảo mà nó ghi thông tin dưới dạng mã ASCII. Thêm người mới vào cuối tệp, mỗi người một dòng. Lưu ý là tên người dùng và số hiệu người dùng phải là duy nhất với mỗi người. Ví dụ thêm một người với tên người dùng là nvviet, theo thứ tự tuần tự người này có UID là 514, thuộc nhóm mặc định GID=505, thư mục cá nhân mà ta sẽ tạo là */home/nvviet*, và shell khởi đầu là Bourne shell (do đó lệnh gọi shell là */bin/sh*)

```
nvviet::514:506:Nguyen Van Viet:/home/vietnv:/bin/bash
```

Chúng ta để trường *password* trống vì ta không thể tạo ra một mật mã đã mã hoá được. Sau khi ghi tệp này, ta sẽ tạo cho người này một mật khẩu tạm thời bằng lệnh

```
$ passwd nvviet
```

Lệnh này sẽ nhắc ta nhập mật khẩu cho người này và yêu cầu anh ta phải đổi mật khẩu ngay lần đăng nhập hệ thống đầu tiên. Thông thường mật khẩu tạm thời là một xâu chung chung, có thể là tên đăng nhập hệ thống (tên người dùng).

Mỗi người dùng đều thuộc về một nhóm, ta thêm tên người dùng mới vào tệp */etc/group* tại dòng biểu diễn nhóm của họ (nếu người này thuộc nhiều nhóm thì phải thêm vào tất cả các dòng định nghĩa các nhóm đó).

Tiếp theo, ta tạo thư mục cá nhân cho họ và đặt cho họ quyền làm chủ thư mục. Theo như trên thì lệnh phải thực hiện sẽ là :

```
$ mkdir /home/nvviet
$ chown nvviet /home/nvviet
```

Cuối cùng, chép các tệp cấu hình cho shell vào thư mục cá nhân của họ và đặt hệ thống để họ có thể sửa đổi tệp này theo ý mình. Nếu ta chép tệp này từ một người dùng khác là *chinq* chẳng hạn thì các lệnh phải dùng là :

```
$ cp /home/chinq/.profile /home/nvviet/.profile
$ chown nvviet /home/.profile
```

Kiểm tra lại bằng tay tệp cấu hình để đảm bảo rằng các biến môi trường không bị đặt sai khi người sử dụng đăng nhập (việc kiểm tra này cũng phải thực hiện bằng một trình soạn thảo ASCII) . Ví dụ dòng định nghĩa biến *HOME*, hay thư mục *spool* cho máy in và e-mail. Nếu dùng Bourne shell và các phiên bản tương thích thì chỉ cần tệp *.profile*. Nhưng nếu dùng *C* shell và các bản tương thích với nó cần *.login* và *.cshrc*. Korn và các bản tương thích cần *.profile* và thường một tệp khác với các biến môi trường nhúng trong đó.

Hệ Linux có một lệnh lấy từ phiên bản Berkeley BSD UNIX, lệnh vipw cho phép soạn thảo một bản sao tạm của */etc/passwd* bằng **vi** (hoặc một trình soạn thảo mặc định khác). Cách dùng một tệp tạm và khoá tệp tạo thành một cơ chế chống nhiều người cùng sửa một tệp đồng thời. Khi ghi tệp, vipw thực hiện một kiểm tra đơn giản trên tệp thay đổi, nếu tất cả đều đúng tệp */etc/passwd* được cập nhật.

Script tự động thực hiện việc thêm người dùng có tên *useradd* hoặc *adduser*. Khi chạy chúng sẽ nhắc chúng ta các thông tin cần thiết cho tệp */etc/passwd*. Chúng đều yêu cầu nhập mật mã, ta có thể bỏ trống. Một ưu điểm của script tự động là tự chép các tệp hỗ trợ shell, và trong một số trường hợp nó thay đổi biến môi trường. Những script này làm đơn giản quá trình thêm một số lượng lớn người dùng mới.

5. Xoá người dùng

Giống như thêm người dùng mới, có thể xoá người dùng bằng một script tự động hoặc bằng tay. Script tự động *deluser* hoặc *userdel* sẽ hỏi chúng ta muốn xoá người dùng nào, sau đó xoá điểm nhập (dòng) của người dùng đó trong tệp */etc/passwd*. Một số script xoá cả *spool* và các tệp trong thư mục cá nhân nếu muốn. Để thực hiện việc loại bỏ người dùng tất nhiên ta phải đăng nhập dưới tên root để có thể xoá trong tệp */etc/passwd*.

Quá trình xoá người dùng bằng tay (hoặc bằng *script* tự động mà không xoá các thư mục và tệp) như sau :

- Xoá điểm nhập tương ứng với người dùng trong */etc/passwd* và trong */etc/group*
- Xoá các tệp *mail* và các *mail alias* của người dùng.
- Xoá mọi *cron* và *at*.
- Xoá thư mục cá nhân nếu không muốn giữ các tệp trong đó.

Để xoá người dùng bằng tay, đầu tiên phải xoá điểm nhập tương ứng trong */etc/passwd*. Đơn giản nhất là xoá dòng tương ứng trong đó. Nếu muốn giữ lại người dùng đó vì một lí do nào đó (chẳng hạn vấn đề sở hữu tệp, một tài khoản dùng chung, hay vấn đề lưu trữ tính toán...), chỉ cần làm cho tài khoản của họ không truy nhập được bằng thay trường *passwd* của họ trong */etc/passwd* bằng dấu *. Như vậy muốn khôi phục lại ta phải dùng lệnh *passwd* để thay đổi mật mã cho họ.

Tiếp theo phải xoá thư mục cá nhân của người này bằng lệnh sau :

```
$ rm -r /home/userdir
```

với */home/userdir* là đường dẫn đầy đủ của thư mục cá nhân.

Sau đó xoá tệp *spool* của *mail* (chỉ có 1 tệp), tệp này thường nằm trong thư mục */user/spool/mail*. Ví dụ muốn xoá của người `sonnt`, ta viết lệnh.

```
$ rm /usr/spool/mail/sonnt
```

Để kết thúc việc xoá `mail`, kiểm tra xem người dùng đó có điểm nhập nào trong tệp mail *alias* không (thông thường là tệp */usr/lib/aliases*). Nếu có, có thể đưa tất cả các mail cho người dùng đó vào một tài khoản khác (như root) với một điểm nhập trong tệp *aliases*. Cuối cùng, phải đảm bảo chắc chắn rằng không còn điểm nhập nào trong các tệp cấu hình *cron* và *at* của người dùng được hệ thống tiếp tục thực hiện. Để hiển thị tệp *crontab* (liệt kê danh sách các lệnh mà cron sẽ thực hiện), hãy dùng lệnh *crontab*.

Để xoá tạm thời một tài khoản, chỉ việc vô hiệu việc đăng nhập vào nó bằng cách thêm dấu * vào đầu trường *passwd* tương ứng trong */etc/passwd* (vẫn giữ nguyên *passwd*). Khi muốn khôi phục lại, chỉ việc xoá dấu * này đi.

6. Sử dụng nhóm

Mỗi người dùng trong các hệ ENIX và Linux đều thuộc về một nhóm. Nhóm là một tập hợp các cá nhân đơn lẻ được gộp lại theo một lí do nào đó. Chẳng hạn người dùng trong nhóm có thể làm cùng phòng, có thể cùng cần truy nhập "một chương trình tiện ích, hoặc cùng truy nhập đến một thiết bị đặc biệt như máy in, máy quét... Mỗi người có thể thuộc nhiều Nhóm. Tuy vậy, họ chỉ có thể là thành viên của một nhóm tại một thời điểm khi nhóm được dùng để xác định quyền trên tệp và Linux chỉ cho phép một số hiệu nhóm tại một thời điểm.

Các nhóm được đặt quyền để các thành viên của nó có thể truy nhập đến các

thiết bị, tệp, các hệ thống tệp, hoặc toàn bộ máy mà những người khác không thuộc nhóm có thể bị hạn chế.

Nhiều hệ Linux chỉ có một nhóm, nhóm mặc định, đây là cách đơn giản nhất để quản lí hệ thống. Trong trường hợp này, truy nhập của người dùng đến các thiết bị và tệp được điều khiển bằng quyền trên tệp chứ không phải bằng nhóm.

Các thông tin về nhóm được để trong tệp */etc/group* có cấu trúc tương tự như tệp */etc/passwd*. Đây là một phần tệp */etc/group* được tạo khi mới cài đặt Linux

```
root::0:root
bin::1:root,bin,daemon
adm::4:root,adm,daemon
wheel::10:root
floppy::11:root
users::100:games
nogroup::-1:
```

Các dòng có khuôn dạng như dưới đây, các trường phân cách bởi dấu ":", hai dấu " :" cạnh nhau thể hiện trường tương ứng rỗng.

```
group name:group password:group ID:users
```

Trong đó các trường có ý nghĩa như sau :

group name : Tên duy nhất xác định một nhóm thường gồm nhiều nhất là 8 kí tự.

password : Trường mật khẩu đã được mã hoá, thường để trắng hoặc là dấu *. Cũng có thể là mật khẩu mà một người muốn ra nhập nhóm phải nhập vào. Không phải mọi phiên bản của Linux hay Unix đều sử dụng trường này. Do đó nó được bỏ trắng để có thể tương thích với nhau.

group ID : Số duy nhất cho mỗi nhóm, được sử dụng bởi hệ điều hành.

Users : Chứa danh sách mọi tên người dùng (user name) thuộc nhóm đó, phân cách bởi các dấu " ,". Danh sách này không kể những người dùng thuộc nhóm đó theo số hiệu nhóm đã được ghi trong tệp */etc/passwd* của người đó (tức là những thành viên mặc định của nhóm)

Mọi hệ *Linux* đều có một số các nhóm mặc định thuộc hệ điều hành. Các nhóm này thường là *bin*, *mail*, *uucp*, *sys* .... Trừ hai nhóm cuối các nhóm còn lại đều là các nhóm phụ thuộc hệ thống. Do vậy, không nên cho một người sử dụng

thuộc vào các nhóm này, vì chúng sẽ cho họ quyền tương tự như root. Chỉ có các đăng nhập hệ thống (vào hệ thống theo các tài khoản của hệ thống) mới cho phép truy nhập đến các nhóm của hệ điều hành.

6.1. Các nhóm mặc định của hệ thống

Các nhóm dùng để đặt quyền trên tệp và quyền truy nhập các tiện ích. Chúng ta xem qua các nhóm quan trọng nhất và chức năng của chúng.

- Nhóm *root/wheel/system* : thường dùng để cho phép người dùng sử dụng lệnh *su* để truy nhập dưới quyền root. Nhóm này làm chủ hầu hết các tệp
- Nhóm *daemon* : dùng để chỉ những người làm chủ thư mục *spool* (*mail*, máy in...)
- Nhóm *kmem* : được dùng cho các chương trình cần truy nhập hạt nhân, bộ nhớ trực tiếp (bao gồm cả lệnh *ps*)
- Nhóm *sys* làm chủ tệp hệ thống. Trong một số hệ thống nó có vai trò như *kmem*.
- Nhóm *tty* làm chủ tất cả các tệp đặc biệt làm việc với *terminal*.

Nhóm mặc định của phiên bản SlackWare của Linux nêu trên có nhóm mặc định là *users* có GID là 100. Các phiên bản khác có nhóm mặc định tên là *group*, cũng là chuẩn của hầu hết các hệ thống Linux.

6.2. Thêm nhóm

Để thêm một nhóm, ta có thể thao tác trực tiếp trên tệp */etc/group* bằng tay sử dụng bất kì một trình soạn thảo ASCII nào, hoặc có thể dùng một tiện ích của shell như *addgroup* hoặc *groupadd*. Đa số các nhà quản trị hệ thống thích làm bằng tay hơn vì có thể nhìn thấy toàn bộ tệp *group* trong khi soạn thảo trên nó. Hơn nữa, không phải hệ thống Linux nào cũng có *addgroup* hay *groupadd*.

Để thêm một nhóm bằng tay, đầu tiên sao lưu tệp */etc/group*. Dùng một trình soạn thảo ASCII và thêm một dòng cho mỗi nhóm mới muốn tạo. các dòng nhất thiết phải tuân theo cú pháp của tệp như đã nêu trên. Nếu không với dòng ghi sai sẽ không cho người dùng không thể thuộc về nhóm đó. Ví dụ, sau đây sẽ có hai nhóm được tạo ra:

```
prof::51:ntthuy
staff::52:chinq
```

Cũng như số hiệu người dùng, số hiệu nhóm nên được gán tuần tự cho tiện, và cũng nên sắp xếp tệp theo thứ tự của số hiệu nhóm (hoặc tên nhóm) dù điều này không bắt buộc. Như trong ví dụ cùng với việc tạo nhóm mới 2 người dùng cũng được đưa vào mỗi nhóm. Để thêm nhiều người dùng vào một nhóm ta sẽ xem trong phần tiếp.

Cuối cùng, phải kiểm tra lại quyền trên tệp và quền sở hữu. Nên để root làm chủ và nhóm làm chủ là root (hoặc system tuỳ theo nhóm có GID bằng 0) để tránh người dùng khác viết vào tệp.

6.3. Thêm người dùng vào một nhóm mới

Mỗi người dùng có thể thuộc nhiều nhóm mà tên người dùng của họ được ghi trong dòng tương ứng với nhóm đó trong tệp */etc/group*. Tên người dùng trong một dòng được phân cách bằng dấu ",", thứ tự người dùng trong mỗi dòng không quan trọng. Ví dụ một vài dòng trong */etc/group* tương ứng với nhóm có nhiều người dùng.

```
staff::52:nvviet,chinq,quanle,sonnh,ducnh,root
laptrinh:53:nvviet,quanl,sonnh,vinhlq,phucnt,sonnt,root
ftpadmin:54:sonnh,quanl,root
svtt:55:khanhnk,hieupd,dungdm,tim
```

Một khi đã đăng nhập vào hệ thống người dùng chỉ có thể thuộc về một nhóm tại một thời điểm, ban đầu sau khi đăng nhập , họ thuộc nhóm có số hiệu được ghi trong trường GID trong tệp */etc/passwd*. Để chuyển đổi giữa các nhóm mà họ là thành viên, phải dùng lệnh *newgrp*.

6.4. Xoá một nhóm

Để xoá hoàn toàn một nhóm, hãy xoá dòng tương ứng nhóm đó khỏi */etc/group*. Đồng thời kiểm tra tệp */etc/passwd* xem có người dùng nào lấy nhóm vừa xoá làm nhóm khởi đầu khi đăng nhập hệ thống không, nếu có phải chuyển họ sang một nhóm mà họ là thành viên. Nếu ta không chuyển nhóm cho họ (tức là đổi trường GID trong */etc/passwd* của họ) thì người dùng này không thể đăng nhập hệ thống được nữa vì có nhóm không hợp lệ. Thêm vào đó chúng ta phải kiểm tra toàn bộ hệ thống tệp xem có tệp hay thư mục nào lấy nhóm này làm chủ sở hữu hay không, nếu có đổi nó sang nhóm khác.

Một số phiên bản của Linux có các script của shell cho phép xoá các dòng

trong */etc/group*, các tiện ích này có tên *delgroup* hay *groupdel*, nhưng hầu hết các phiên bản của Unix không dùng chúng.

6.5. Lệnh su *(switch user)*

Đôi khi chúng ta muốn thực hiện một số lệnh như một người dùng khác. Nếu đang đăng nhập dưới tên siêu người dùng và muốn tạo tệp và đặt các quyền trên tệp và quyền sở hữu của `nvviet` thì đăng nhập lại dưới tên `nvviet` sẽ dễ làm việc hơn dưới tên root rồi đổi lại các tham số. Tương tự nếu người sử dụng bình thườngmuốn trở thành người quản trị trong một khoảng thời gian, họ phải thoát khỏi phiên làm việc, đăng nhập lại dưới tên mới và làm các thay đổi mong muốn. Để tránh những công việc đó, ta dùng lệnh *su*.

Lệnh *su* chuyển người dùng sang tên người dùng mới và trao cho họ các quyền mà người dùng mới có. Lệnh *su* lấy tên người dùng làm tham số, ví dụ khi người sử dụng đăng nhập dưới tên một người dùng bình thường và muốn có quyền sử dụng hệ thống như root họ có thể thực hiện câu lệnh sau :

```
$ su root
```

khi đó Linux sẽ yêu cầu nhập mật khẩu của root, nếu gõ đúng, người sử dụng sẽ trở thành root cho đến khi nhấn Ctrl+D để thoát khỏi tài khoản này và trở về với tài khoản ban đầu.

Tương tự, nếu đang sử dụng hệ thống với tài khoản root mà người sử dụng muốn chuyển sang một người dùng khác ví dụ `tiennc`, khi đó có thể thực hiện lệnh :

```
$ su tiennc
```

Linux không yêu cầu người sử dụng nhập mật mã khi chuyển từ root sang vì họ đang có quyền của siêu người dùng. Nếu đang đăng nhập dưới tên một người dùng bình thường khác root mà muốn chuyển sang một người dùng khác thì cần phải nhập mật mã.

Như vậy có thể thấy */etc/passwd* và */etc/group* là hai tệp cơ bản liên quan đến truy nhập Linux. Người quản trị hệ thống có thể dễ dàng sửa đổi hai tệp này để thêm người dùng hay nhóm bất kì lúc nào. Nhưng cần ghi nhớ rằng đây là hai tệp sống còn và cần kiểm tra lại quyền truy nhập vào chúng sau khi sửa.

Chương II

QUẢN LÝ TÀI NGUYÊN

I. HỆ THỐNG TỆP VÀ THIẾT BỊ LƯU TRỮ

Một trong những nhiệm vụ quan trọng nhất của người quản trị hệ thống là quản lí đĩa cứng và hệ thống tệp của Linux, giữ cho cả hai ở trong tình trạng tốt giúp cho Linux thể hiện được hết khả năng của mình. Điều này đòi hỏi thực hiện thường xuyên một số việc mà chúng ta sẽ xem xét trong phần này. Nhìn chung những công việc này gồm :

- Tìm các cung từ hỏng của hệ thống tệp.
- Kiểm tra tính chặt chẽ và tính đúng đắn của bảng *i-node* của hệ thống tệp.
- Kiểm tra các quyền truy nhập và sở hữu để đảm bảo các truy nhập là thích đáng.
- Làm cho các hệ thống tệp (địa phương hoặc ở xa) sẵn sàng cho người sử dụng.
- Quản lí không gian đĩa của hệ thống Linux.
- Thường xuyên sao lưu dữ liệu để đảm bảo an toàn.

Mặc dù một vài trong số các hành động kể trên được thực hiện tự động khi khởi động Linux (như kiểm tra các cung từ hỏng), ta cũng nên biết cách thực hiện các thủ tục này bằng tay cũng như cách chúng thực hiện để sửa chữa các lỗi có thể

xảy ra. Với vấn đề sao lưu dữ liệu ta sẽ xem trong phần "Sao lưu và khôi phục" sau, còn việc kiểm tra quyền trên tệp, chúng ta đã xem trong phần trước.

1. Gắn kết và tháo bỏ các hệ thống tệp

Trong Linux khái niệm hệ thống tệp để chỉ một thiết bị ngoại vi được định dạng để có thể lưu các tệp cho phép truy cập ngẫu nhiên ví dụ đĩa cứng, đĩa mềm, CD-ROM... (băng từ chỉ cho phép truy nhập tuần tự nên không thể chứa một hệ thống tệp)

Nguyên nhân của cơ chế gắn kết các hệ thống tệp là cách Linux tổ chức đĩa và các hệ thống tệp tạo nên toàn bộ cấu trúc thư mục. Linux sử dụng một cấu trúc thư mục duy nhất, không quan tâm đến việc lưu trữ trên bao nhiêu đĩa và bao nhiêu phân vùng. Mỗi hệ thống tệp của mỗi phân vùng phải là một phần của cấu trúc thư mục lớn hơn. Toàn bộ cây thư mục chỉ có một thư mục gốc, các hệ thống tệp khác được gắn kết vào ở các mức thấp hơn.

Để có thể hình dung khái niệm này, hãy tưởng tượng một hệ thống tệp Linux chuẩn với một phân vùng gốc (*/*) ở trên cùng, các phân vùng khác được chia nhánh từ phân vùng gốc. Phân vùng gốc nằm trên một phân vùng của đĩa cứng đầu tiên. Thông thường, đĩa đó có cả các thư mục khác trên nó như */dev*, */lib*, */etc*, ... tất cả các thư mục cần thiết cho việc khởi động một hệ điều hành Linux nhỏ nhất phải được đặt trên phân vùng gốc.

Tuy vậy, giả sử rằng bạn muốn có một hệ thống tệp */usr* rất lớn để hỗ trợ nhiều người dùng với các cơ sở dữ liệu tệp rất lớn. Phân vùng gốc có thể không có đủ chỗ cho tất cả các tệp mà bạn muốn lưu. Khi đó bạn có thể dùng một phân vùng khác (trên cùng hoặc trên một đĩa cứng khác), định dạng nó thành một hệ thống tệp của Linux, và sau đó gắn kết chúng vào hệ thống tệp gốc dưới thư mục */usr*. Mỗi khi người dùng di chuyển từ thư mục */bin* (trong phân vùng thứ nhất) đến */usr/tienmc* chẳng hạn, người dùng đã di chuyển đến một phân vùng hay đĩa khác. Người dùng không thể nhận thấy được sự di chuyển giữa các phân vùng vì các phân hoạch trông giống như một cây thư mục duy nhất. Thư mục */usr* được gọi là được gắn kết vào thư mục gốc.

Một cách chính xác, phân vùng chứa hệ thống tệp */usr* được gắn kết vào hệ thống tệp gốc tại vị trí thư mục */usr*. Nó cũng có thể được gắn kết vào thư mục */home*, Linux không quan tâm đến việc ta gắn kết hệ thống tệp vào đâu, miễn là nó được gắn kết vào một thư mục tồn tại trong hệ thống tệp gốc và còn trống, nếu không nội dung của thư mục sẽ bị xoá.

Ta cũng có thể gắn kết một hệ thống tệp vào một hệ thống tệp khác đã được gắn kết với hệ thống tệp gốc. Ví dụ gắn kết một CD-ROM vào thư mục */usr/tienmc/cd rom*, trong đó */usr* là một hệ thống tệp mà ta đã gắn kết vào hệ thống tệp gốc. Một lần nữa sự di chuyển giữa các hệ thống tệp là trong suốt với người dùng.

Linux cho phép ta gắn kết các phân vùng vào bất kì chỗ nào, từ bất kì nguồn nào, miễn là chúng phù hợp với toàn bộ cấu trúc hệ thống tệp. Chỉ có một chỗ không thể gắn kết các hệ thống tệp là thư mục gốc nằm trong hệ thống tệp gốc. Linux cũng cho phép gắn kết một số hệ thống tệp của các hệ điều hành khác như DOS hay OS/2 vào hệ thống tệp của Linux. Để thực hiện điều đó Linux cần biết chỗ truy nhập hệ thống tệp đó (thư mục */dos*, */usr/dos*, hoặc một số thư mục khác) và kiểu của hệ thống tệp đó. Sau khi đã gắn kết Linux sẽ cho phép người sử dụng di chuyển giữa các thư mục và tệp của hệ thống đó như trong bất kì thư mục nào khác của Linux. Tuy vậy, tại một thời điểm, một hệ thống tệp chỉ có thể được gắn kết vào một vị trí, chẳng hạn ta không thể gắn kết một hệ thống tệp dưới cả */usr* và */home*.

Khả năng gắn kết các hệ thống tệp làm cho Linux trở nên linh hoạt. Nếu muốn gắn kết một ổ cứng chứa dữ liệu là hệ thống tệp mà Linux có thể hiểu được, người sử dụng có thể cắm ổ cứng này vào máy, sau đó gắn kết hệ thống tệp của nó với bất kì chỗ nào trên hệ thống tệp của Linux. Và như vậy người sử dụng có thể truy nhập được dữ liệu trên ổ cứng đó. Mọi thiết bị có thể chứa hệ thống tệp đều có thể được gắn kết: CD-ROM, đĩa mềm, đĩa quang, ...

Linux cung cấp lệnh *mount* để gắn kết một hệ thống tệp. Cú pháp câu lệnh như sau:

```
$ mount ten-thiet-bi diem-noi
```

trong đó *ten-thiet-bi* là tên thiết bị lưu trữ (phân vùng, đĩa cứng, CD-ROM,...), *diem-noi* là tên thư mục mà thiết bị được gắn kết vào. Ví dụ câu lệnh để gắn kết */dev/sda4* (phân vùng thứ 4 của ổ đĩa SCSI thứ nhất) vào thư mục */usr* là:

```
$ mount /dev/sda4 /usr
```

Để gắn kết một hệ thống tệp CD-ROM (như */dev/cdrom* vào thư mục */cdrom* (giả sử đã tồn tại), ta dùng lệnh

```
$ mount /dev/cdrom /cdrom
```

Lưu ý là chỉ có root mới đủ quyền thực hiện công việc này. Mặc dù có thể cho phép người dùng gắn kết các hệ thống tệp nhưng điều này có thể dẫn đến các

vấn đề an toàn, do đó không được khuyến khích.

Ta có thể gắn kết một hệ thống tệp theo kiểu chỉ đọc, đặc điểm này cho phép cấm ghi lên hệ thống tệp được gắn kết (có thể chứa những dữ liệu mà ta không muốn làm hỏng), mọi cố gắng ghi lên nó sẽ sinh ra thông báo lỗi. Để gắn kết một hệ thống tệp theo kiểu chỉ đọc, dùng tuỳ chọn -r :

```
$ mount -r /dev/cdrom /cdrom
```

hoặc như một số phiên bản cũ của UNIX và Linux cho phép viết -r ở cuối

```
$ mount /dev/cdrom /cdrom -r
```

Khi một hệ thống tệp không còn được gắn kết trong hệ thống tệp của Linux nữa (do đó người dùng không thể truy nhập đến các thư mục nơi gắn kết nó), hệ thống tệp được gọi là đã bị tháo bỏ. Tất cả các hệ thống tệp đều có thể bị tháo bỏ trừ hệ thống tệp gốc. Để tháo bỏ một hệ thống tệp, người sử dụng dùng lệnh umount. Lệnh umount lấy tên của thiết bị hoặc điểm nối làm tham số. Để bỏ nối CD-ROM vừa được gắn kết trong ví dụ trên, ta dùng một trong hai lệnh sau :

```
$ umount /dev/cdrom
$ umount /cdrom
```

Ta không cần bỏ gắn kết cho mọi hệ thống tệp trước khi đóng hệ thống, Linux sẽ tự động thực hiện công việc đó trong quá trình đóng hệ thống.

2. Gắn kết các hệ thống tệp tự động bằng tệp /etc/fstab

Các hệ thống tệp đã được gắn kết từ trước đều không cần gắn kết tự động khi hệ thống khởi động lại (khác với root, luôn được gắn kết tự động khi khởi động hệ thống). Khi khởi động Linux, cần phải biết tìm các hệ thống tệp ở đâu để gắn kết. Linux dùng tệp */etc/rc* (chạy khi Linux khởi động) để thực hiện lệnh:

```
$ mount -av
```

Khi thực hiện lệnh này, Linux biết được phải đọc tệp */etc/fstab* để tìm ra hệ thống tệp cần gắn kết cũng như nơi cần gắn kết.

Linux cũng cung cấp lệnh *mountall* để nối tất cả các hệ thống tệp trong tệp */etc/fstab*. Tuy vậy, không phải mọi phiên bản của Linux đều hỗ trợ câu lệnh này.

Mỗi dòng trong tệp */etc/fstab* (là tệp ASCII) có khuôn dạng như sau :

```
ten-thiet-bi    diem-noi    he-thong-tep_kieu    cac-tuy-chon
dump_tan-so thu-tu
```

device là tên thiết bị lưu trữ, tiếp theo là điểm gắn kết, kiểu hệ thống tệp, tiếp

theo là các chỉ dẫn làm thế nào để xử lí hệ thống tệp.

Ví dụ một tệp /etc/fstab :

```
/dev/sda1 / ext2 defaults 1 1
/dev/sda2 /usr ext2 defaults 1 1
/dev/sda3 /usr/data ext2 defaults 1 1
/dev/cdrom /cdrom iso9660 ro 1 1
/dev/sda4 /dos msdos defaults 1 1
/dev/sdb1 /data ext2 defaults 1 1
```

Trong đó ta thấy hệ thống tệp gốc là */dev/sda1* là hệ thống tệp Linux điển hình ext2. CD-ROM được gắn kết vào */cdrom*, nó là hệ thống tệp theo kiểu ISO 9660 (CD-ROM) và được gắn kết theo kiểu chỉ đọc.

Linux gắn kết các hệ thống tệp theo thứ tự của chúng trong tệp */etc/fstab*. Ta có thể thấy trong ví dụ trên dòng chỉ ra gắn kết */usr/data* nằm sau dòng chỉ ra gắn kết */usr*. Nếu dòng nối */usr/data* nằm trước thì lệnh *mount* tương ứng với nó không thể làm việc vì thư mục */usr* chưa tồn tại. Trong trường hợp một gắn kết không thực hiện được, Linux bỏ qua nó và thực hiện các gắn kết còn lại. Nếu một gắn kết của một thư mục sẽ được sử dụng trong các gắn kết sau không thực hiện được, các gắn kết phụ thuộc vào nó cũng sẽ không thực hiện được. Ví dụ nếu kết nối */usr* bị hỏng vì lí do nào đó, thì kết nối */usr/data* cũng sẽ không thực hiện được vì thư mục */usr* không tồn tại.

Hai con số cuối cùng của mỗi dòng trong */etc/fstab* là tần số sao lưu và thứ tự kiểm tra, nói chung không có ý nghĩa với một số phiên bản Linux. Tần số sao lưu chỉ mức độ thường xuyên mà Linux sao lưu lại hệ thống tệp đó. Nếu là 1 có nghĩa là sao lưu hàng ngày, 2 có nghĩa là sao lưu hai ngày một lần. Con số này được dùng cho các chương trình sao lưu tự động, nó sẽ phân tích tệp */etc/fstab* để lấy thông tin này.

Thứ tự kiểm tra là thứ tự mà chương trình *fsck* kiểm tra hệ thống tệp. Hệ thống tệp có thứ tự kiểm tra 1 sẽ được kiểm tra đầu tiên, hệ thống tệp có thứ tự 2 được kiểm tra thứ 2,... Nếu có nhiều hệ thống tệp có thứ tự kiểm tra bằng 1, các hệ thống tệp này sẽ được kiểm tra theo thứ tự xuất hiện trong */etc/fstab*. Hệ thống tệp gốc phải có giá trị 1, theo lệ thường, các phân vùng khác được đặt số cao hơn. Tuy vậy, đa số hệ Linux không dùng con số này.

Ta có thể để cả các phân vùng *swap* trong */etc/fstab*. Khi đó, đặt thư mục gắn kết đến là *none* và tần số sao lưu và thứ tự kiểm tra bằng 0. Ví dụ :

```
/dev/sda2 none swap sw 0 0
```

Khi để phân vùng *swap* trong */etc/fstab*, ta có thể kích hoạt nó bằng lệnh *swapon*

```
$ swapon -a
```

Linux đọc tệp */etc/fstab* và kích hoạt tất cả các phân vùng *swap*. Lệnh này thường được để trong tệp */etc/rc* vì thế nó được thực hiện tự động khi khởi động Linux.

Tiếp theo, ta sẽ xem xét các kiểu hệ thống tệp và các giá trị cho trường `tuy-chon` trong tệp */etc/fstab* (tức là hướng dẫn xử lí hệ thống tệp)

2.1. Các kiểu hệ thống tệp

Các kiểu hệ thống tệp mà Linux hỗ trợ khác nhau tuỳ theo phiên bản Linux. Hầu hết các phiên bản hỗ trợ các hệ thống tệp sau :

Bảng 2.1. Các kiểu hệ thống tệp

ext2	Kiểu hệ thống tệp mở rộng thứ hai (second extended), là kiểu phổ biến nhất của phân vùng Linux.
ext	Kiểu hệ thống tệp mở rộng gốc của Linux(extended), đã bị thay thế bởi ext2.
minix	Kiểu hệ thống tệp Minix gốc, ít được dùng nhưng vẫn được hỗ trợ vì là khuôn dạng hệ thống tệp đầu tiên của Linux
xia	Kiểu hệ thống tệp Xia, ít sử dụng vì bị thay thế bởi ext2.
umsdos	Kiểu hệ thống tệp UMS-DOS, dùng khi cài đặt Linux trên một phân vùng DOS (không có phân vùng riêng cho Linux) .
proc	Kiểu tệp dựa trên proc, dùng cho một số tiến trình sử dụng thông tin hệ thống
iso9660	Kiểu hệ thống tệp ISO 9660 sử dụng trên hầu hết CD-ROM.
xenix	Kiểu hệ thống tệp SCO Xenix cung cấp hỗ trợ Xenix dưới Linux.
sysv	Kiểu hệ thống tệp UNIX System V, cho phép hỗ trợ cho ổ System V dưới Linux.
coherent	Kiểu phiên bản Coherent Unix của Mark Williams dùng cho hầu hết các CD-ROM, cho phép hỗ trợ các hệ thống tệp Coherent dưới Linux
msdos	Kiểu phân vùng DOS, Linux có thể truy nhập
hpfs	Kiểu hệ thống tệp High Performance, cho phép hỗ trợ cho HPFS hưới Linux.

Một số phiên bản của Linux không hỗ trợ cho tất cả các hệ thống tệp trên, đặc biệt là các hệ thống tệp như *coheren* và *minix*. Hệ thống tệp mạng NFS (Network File System), được các phiên bản gần đây của Linux hỗ trợ sẽ được xem xét trong phần sau.

2.2. Các tuỳ chọn

Trường tuỳ chọn (`tuy-chon`) trong tệp */etc/fstab* có thể có một số giá trị khác nhau, tuỳ vào phiên bản của Linux. Với hầu hết các phiên bản của Linux (mà dựa trên BSD UNIX), ta có thể sử dụng các tuỳ chọn sau để mô tả đặc điểm hệ thống tệp.

Bảng 2.2. Các tuỳ chọn mô tả hệ thống tệp

default	Khác nhau tuỳ theo phiên bản, nhưng thông thường là đọc-ghi, suid và quota
rw	Đọc-ghi
ro	Chỉ đọc
suid	Cho phép truy nhập theo chế độ SUID
nosuid	Không được truy nhập theo chế độ SUID
quota	để quota có tác dụng
noquota	Quota không có tác dụng

Nếu kiểu hệ thống tệp là nfs, còn có nhiều tuỳ chọn khác được hỗ trợ. Tuỳ chọn *default* có xu hướng là tuỳ chọn tốt nhất cho các hệ thống tệp truyền thống được nối vào một ổ cứng địa phương.

Thuật ngữ SUID thường dùng khi quản trị hệ thống. SUID (Set User ID) là bit quyền gắn với tất cả các tệp và thư mục. Tương tự ta cũng có bit SGID (Set Group ID). Mọi tệp hay thư mục có các bit này được thiết lập sẽ hoạt động với quyền của chủ tệp khi nó được thực thi. Ví dụ, ta đăng nhập hệ thống dưới một tên người dùng thông thường và thực hiện một tệp nhị phân có SUID được thiết lập, tệp nhị phân sẽ thực thi như nó được thực hiện chạy bởi root. Cả hai SUID và SGID đều đáng ngại vì chúng chúng có thể dẫn đến vấn đề an toàn và bảo mật.

3. Quản lí không gian đĩa

Hệ điều hành, các ứng dụng, chương trình dịch và đặc biệt các tệp của người dùng trong hệ thống chia sẻ làm cho bao nhiêu không gian đĩa cũng nhanh chóng trở nên không đủ. Đĩa cứng hiện nay không đắt nữa nên nhiều người quản trị hệ thống chọn giải pháp mua thêm ổ đĩa cứng, điều đó tránh khỏi phải thường xuyên don dẹp các tệp để lấy không gian nhưng vẫn cần có chính sách sử dụng đĩa cho người dùng sao cho không gian đĩa không bị sử dụng một cách lãng phí. Để có được một chính sách sử dụng đĩa như vậy, người quản trị hệ thống cần phải biết cách xác định không gian đĩa được sử dụng, quản lí đĩa một cách hiệu quả và dọn dẹp đĩa.

3.1. Kiểm tra hệ thống tệp

Một phần của trình khởi động Linux (điều khiển bởi một dòng trong */etc/rc*) là kiểm tra tất cả các hệ thống tệp để đảm bảo chúng không bị hỏng. Công việc này thực hiện mỗi khi khởi động lại hệ thống. Tuy vậy, nếu máy không thường xuyên khởi động lại hoặc bạn là người có kinh nghiệm về về các lỗi của đĩa thì cũng nên tự chạy chương trình kiểm tra hệ thống tệp.

Nhìn chung, người sử dụng dùng chương trình *fsck* (file system check) để kiểm tra hệ thống tệp. Linux sử dụng một số phiên bản khác của *fsck* để kiểm tra các hệ thống tệp phụ thuộc Linux. Vì vậy, ta có thể không trực tiếp truy nhập *fsck*. Ví dụ nhiều phiên bản Linux có phiên bản riêng của fsdk để kiểm tra các hệ thống *ext2* là *e2fsck*.

Nếu *fsck* có trong một phiên bản của Linux, phiên bản thích hợp của *fsck* sẽ được tìm trong các thư mục */bin*, */sbim*, */etc/fs*, */etc* và thực hiện phiên bản đó. Quá trình tìm và thực hiện là trong suốt đối với người sử dụng trong hầu hết các trường hợp.

fsck thực hiện một số công việc, phần lớn hoạt động của nó là quét toàn bộ hệ thống tệp để tìm các lỗi sau :

- Một block được dùng bởi nhiều tệp (vấn đề liên kết chéo : *cross-linked*) .
- Các block được dùng nhưng lại được đánh dấu là không.
- Các điểm nhập mâu thuẫn giữa các tệp và bảng *i-node*.
- Đếm số liên kết không đúng.
- Các điểm nhập không hợp lệ trong bảng *i-node*.

- Mâu thuẫn giữa giá trị kích thước bảng *i-node* và không gian được sử dụng bởi các tệp.
- Các giá trị không hợp lệ trong tệp.
- Các tệp đã mất mà không xuất hiện trong bảng *i-node*.

Toàn bộ quá trình diễn ra rất nhanh, do đó không có lí do gì mà không chạy *fsck* thường xuyên. Nếu *fsck* báo cáo các lỗi, hãy đóng hệ thống và chuyển sang chế độ chỉ dành cho siêu người dùng và chạy lại *fsck*. Các vấn đề có thể đã nảy sinh do ứng dụng của người dùng, vì vậy bước này sẽ xác định các lỗi kiểu đó. Nếu đĩa vẫn có lỗi, ta có thể sửa nó trong chế độ siêu người dùng.

Đa số các trường hợp, *fsck* chỉ chạy trên các hệ thống tệp đã được bỏ gắn kết trừ hệ thống tệp gốc. Nếu muốn kiểm tra các hệ thống tệp, trước hết bỏ gắn kết cho nó và sau đó chạy *fsck*. Để kiểm tra hệ thống tệp gốc, chuyển hệ thống sang chế độ một người dùng và chạy *fsck*.

Lệnh *fsck* lấy tham số là thiết bị hoặc điểm nối hệ thống tệp mà ta muốn kiểm tra. Ví dụ, cả hai dòng lệnh này đều đúng :

```
$ fsck /dev/sda1
$ fsck /usr
```

Nếu fsck làm việc trên nhiều ổ đĩa (nhờ gắn kết), nó sẽ cố gắng thực hiện song song chừng nào có thể để giảm thời gian cần thiết để kiểm tra đĩa.

Các tuỳ chọn của *fsck* được hỗ trợ bởi hầu hết các hệ Linux và thường được sử dụng bởi người quản trị hệ thống là:

Bảng 2.3. Các tùy chọn của lệnh fsck

-a	Tự động sửa hệ thống tệp mà không cần hỏi lại người dùng
-r	Sửa hệ thống tệp có tương tác với người sử dụng (hỏi người dùng các chỉ dẫn). Dùng nó chỉ khi kiểm tra một hệ thống tệp đơn lẻ.
-t <type>	Chỉ ra kiểu hệ thống tệp kiểm tra. Nếu type bắt đầu bằng no, chỉ các kiểu hệ thống tệp khác mới được kiểm tra. Tuỳ chọn này dùng các kiểu hệ thống tệp trong tệp */etc/fstab*.
-v	Tuỳ chọn này cho phép đưa ra báo cáo dài

Chú ý: Thỉnh thoảng chạy *fsck* để kiểm tra tính chặt chẽ của hệ thống. Mỗi khi gặp một thông báo lỗi đĩa, hãy chạy *fsck* ngay.

3.2. Hiển thị các thống kê về hệ thống tệp

Hai lệnh thường được dùng để xem các thống kê hệ thống tệp nhất là *df* (disk filesystem) và *du* (disk usage), chúng cho biết không gian đã dùng, không gian còn trống, ...

Lệnh df

Lệnh *df* được dùng nhiều nhất để lấy các thống kê về hệ thống tệp. Nó hiển thị các thông tin về tất cả các hệ thống tệp trong hệ thống, dung lượng tổng cộng, lượng còn trống trên mỗi hệ thống tệp, các vị trí nối hiện tại. Ví dụ, một lệnh *df* cho kết quả :

```
$ df
Filesystem          1k-blocks   Used Available Use% Mounted on
/dev/hda3             3826584   1412516   2219684  39% /
/dev/scd0              663516   663516          0  100% /cdrom
```

Như vậy hệ thống này có một đĩa cứng có 1 phân vùng cho Linux */dev/hda3* có dung lượng tổng cộng 3.826.584K, đã dùng 1.412.516K, không gian còn trống 2.219.684K, phân vùng này đã sử dụng %, và nó là phân vùng gốc. Ngoài ra nó có gắn kết với CD-ROM tại thư mục */cdrom*

Đôi khi, chúng ta thấy thông báo phần trăm sử dụng đĩa là 100%. Đó là do Linux giữ lại 10% đĩa cho siêu người dùng sử dụng. Như vậy, root có thể sử dụng 110% dung lượng được hiển thị. Tuy vậy, khi lượng sử dụng lên đến 100% thì cũng là lúc phải dọn dẹp đĩa.

Tuỳ chọn -i của lệnh đưa ra các thông tin tương tự về bảng i-node

```
$ df -i
Filesystem          Inodes   IUsed   IFree IUse% Mounted on
/dev/hda3           486720   72246  414474   15% /
/dev/scd0                0       0       0    0% /cdrom
```

Cũng với hệ thống như trên, *df* đưa ra các con số về các *i-node* sẵn có, lượng đã dùng, số còn rỗi, và phần trăm đã sử dụng. Không có sự tương quan giữa thông tin sử dụng bảng *i-node* và thông tin sử dụng đĩa. Vì vậy, nên hiển thị cả hai thông tin. Một *i-node* được sử dụng khi một tệp được dùng. Nếu nhiều tệp nhỏ được lưu thì bảng *i-node* cũng có thể đầy. Do đó phải kiểm tra cả tình trạng sử dụng đĩa và

bảng *i-node*.

Lệnh *df* bỏ qua các hệ thống tệp được đánh dấu bỏ qua trong tệp */etc/fstab* (thường chỉ có các tệp *swap* có thiết lập này). Mặc định, lệnh *df* hiển thị tất cả các hệ thống tệp được gắn kết vào hệ thống, trừ phi ta đặc tả một hệ thống tệp cho lệnh.

Lệnh *df* hiển thị không gian sử dụng đĩa theo đơn vị 1KB, trừ phi ta đặt biến môi trường *POSIXLY_CORRECT* trong tệp khởi tạo hệ thống. Nếu biến này được thiết lập, thông tin sẽ được hiển thị theo đơn vị 512 byte. Cách đặt này thuận lợi khi dùng các hệ thống tệp cũ quy định kích thước cung từ là 512 byte.

Một số tuỳ chọn của lệnh *df*, hầu hết được các phiên bản của Linux hỗ trợ :

Bảng 2.4. Các tuỳ chọn của lệnh df

-a, -all	Hiển thị cho cả các hệ thống tệp không có *block* nào (Các hệ thống tệp không có *block* nào thường dùng cho các mục đích đặc biệt như tự động gắn kết các thiết bị đặc biệt.)
-help	Hiển thị các thông tin trợ giúp
-i, -inode	Hiển thị các thông tin *i-node*
-k, -kilobyte	Dùng để hiện không gian theo đơn vị 1K thay vì 512 byte
-p 4	Dùng khuôn dạng *POSIX* để hiển thị thông tin về hệ thống tệp trên một dòng mà không có cuốn dòng. Nếu tên hệ thống tệp dài quá 20 kí tự, tuỳ chọn này sẽ bắt buộc các cột không căn lề nữa.
-T	Hiển thị thêm thông tin kiểu của các hệ thống tệp.
-t<type>	Chỉ hiển thị các hệ thống tệp có kiểu phù hợp với kiểu đưa ra.
-v	Hiển thị số phiên bản của *df*
-x<type>	Hiển thị mọi hệ thống tệp khác kiểu đã chỉ ra.

Các tuỳ chọn trên có thể được tổ hợp với nhau khi dùng. Ta cũng có thể đưa các lệnh thường dùng với nhau vào một shell script để chạy mỗi khi cần.

Lệnh du

Lệnh *du* dùng để hiển thị các thông tin thống kê tình trạng sử dụng đĩa có ích. Khi chạy lệnh du không có tham số, nó hiển thị lượng đĩa đã dùng bởi các tệp, thư mục con dưới một thư mục đưa ra hay thư mục hiện tại nếu không đưa ra thư mục nào. Hãy xem hai ví dụ sau:

```
$ du
125 /info/a_temp
4 /info/data
265 /info/data/book
726 /info/data/book/chap_1
```

Và :

```
$ du /usr/tiennc
35 /usr/tiennc/bin
2736 /usr/tiennc/book
3 /usr/tiennc/source
```

Lệnh *du* hiển thị không gian sử dụng của mỗi thư mục theo block trong cột đầu và tên thư mục trong cột thứ hai.

Nếu chạy *du* trên một cây thư mục lớn, thông tin đưa ra sẽ rất dài, ta có thể tóm tắt nó bằng tuỳ chọn -s (summarize)

```
$ du -s /usr/linhtd
3315 /usr/linhtd
```

Lệnh *du* có một số tuỳ chọn sau :

Bảng 2.6. Các tuỳ chọn của lệnh du

-a, -all	Hiển thị thông tin cho mọi thư mục và tệp
-b, -bytes	Hiển thị kích thước theo byte
-c, -total	Hiển thị toàn bộ
-k	Hiển thị kích thước theo kilobyte , thay vì 512 byte (nếu trước đó đã đặt là 512 byte)
-l	Hiển thị kích thước của tất cả các tệp, kể cả các tệp liên kết
-s	Chỉ hiển thị tổng số
-v	Hiển thị phiên bản
-x	Bỏ qua các thư mục trong các hệ thống tệp khác được gắn kết vào hệ thống tệp hiện tại.

3.3. Dọn dẹp đĩa cứng

Khi thiếu không gian đĩa, có thể dùng giải pháp mua thêm ổ đĩa thêm phân vùng cho Linux. Nhưng như vậy không được thực tế lắm và cũng không phải điều ta mong muốn. Giải pháp trong trường hợp này là xoay xở tối đa với những gì ta có.

Hệ thống hoạt động kém trước tiên là do đĩa bị phân mảnh. Các đầu đọc phải di chuyển nhiều để truy nhập hay ghi một tệp. Vì vậy cần loại bỏ sự phân mảnh đĩa.

Để giảm không gian sử dụng đĩa, đầu tiên nên loại bỏ các ứng dụng và phần mềm không cần dùng đến, nhất là những phần mềm được cài đặt tự động khi cài đặt Linux, chẳng hạn như khi gỡ bỏ bộ dịch C, không gian đĩa có thể giảm được tới 50MB.

Tiếp theo, yêu cầu người dùng hệ thống dọn dẹp không gian dành cho họ, xoá các bản sao của các tệp, bỏ các tệp lưu tự động và các tệp *log*. Chỉ riêng việc xoá bỏ các tệp *log* cũng có thể tiết kiệm không gian đĩa tới 30MB. Các tệp *log* cơ bản bao gồm:

Bảng 2.7. Các tệp tin log cơ bản

/usr/spool/lp/log	Printing log
/usr/lib/cron.log	File log của cron
/usr/spool/uucp/LOGFILE	File log của UUCP

Kích thước các tệp này có thể tăng nhanh một cách đáng kinh ngạc. Nếu muốn giữ lại một số dòng trong tệp log, hãy dùng lệnh *tail*. Ví dụ để giữ lại 100 dòng cuối còn xoá các dòng còn lại.

```
$ cd /usr/spool/lp
$ tail -100 log > tmp
$ mv tmp log
```

Sao lưu các chương trình, dữ liệu không hay dùng đến vào các thiết bị nhớ khác và xoá chúng khỏi ổ cứng. Nếu thật cần thiết, hãy giữ chúng trong ổ cứng dưới dạng nén.

Nên nhớ rằng tệp *core*, *.log*, *.error* là các tệp mà kích thước tăng nhanh đến khá lớn và chúng ta có thể xoá đi mà chẳng hại gì. Để xoá đi các tệp này, ví dụ tất

cả các tệp *core*, ta dùng lệnh

```
$ find / -name core -exec rm {} \;
```

4. Giới hạn không gian đĩa cho người dùng

Hoạt động một hệ thống đa người dùng như Linux luôn đặt hệ thống vào tình trạng bị thiếu không gian đĩa. Giải pháp lí tưởng nhất là hạn chế không gian đĩa mà mỗi người sử dụng dùng, điều này được thực hiện nhờ lệnh *quota*.

Quota là một trong các công cụ quản lí tài nguyên tốt nhất, nó cho phép đưa ra và giới hạn không gian sử dụng đĩa của người dùng. Nó duyệt tệp */etc/fstab* và kiểm tra tình trạng dùng đĩa theo thứ tự các hệ thống tệp trong */etc/fstab*.

Quota đặt sẵn một không gian đĩa cho người dùng hay nhóm, khi người dùng đã dùng đến giới hạn có thể sẽ có một thông báo xuất hiện trên màn hình và các báo cáo sử dụng đĩa sẽ được gửi cho root.

4.1. Giới hạn cứng và mềm

Để giới hạn không gian sử dụng của người dùng, ta có thể dùng giới hạn cứng hoặc giới hạn mềm. Giới hạn cứng không thể bị vượt qua trong bất kì hoàn cảnh nào và khi đạt tới giới hạn họ không nhận được một cảnh báo nào cả. Một giới hạn mềm cho phép người sử dụng đạt tới giới hạn nhưng họ sẽ nhận được một cảnh báo. Lí tưởng nhất là ta đặt một giới hạn mềm nhỏ hơn giới hạn cứng để người dùng có thể nhận được cảnh báo trước khi họ không còn ghi được gì nữa.

4.2. Khi nào dùng Quota

Không phải mọi hệ thống tệp đều cần đặt giới hạn. Nếu có một số ổ đĩa cứng được chia thành nhiều hệ thống tệp, ta có thể dùng một số cho không gian lưu trữ không giới hạn, và số còn lại có thể cần *quota*. Nói chung thì quyết định dùng quota cho hệ thống tệp nào phụ thuộc vào chính người sử dụng. Hầu hết các hệ Linux đòi hỏi phải sửa tệp */etc/fstab* để chỉ ra hệ thống tệp nào dùng *quota*, điều này được chỉ ra ở cột thứ 4 của mỗi dòng. Ví dụ:

```
$ /dev/sda3 /usr rw,quota 1 3
```

Dòng này chỉ ra rằng hệ thống tệp */usr* dùng *quota*. Nếu dòng này có chữ *noquota* ta có thể đổi thành *quota* để dùng *quota* cho hệ thống tệp.

4.3. Đặt các quota người dùng

Người quản trị hệ thống đặt các *quota* người dùng trong tệp *quota.user* trong thư mục gốc của hệ thống tệp áp dụng *quota*. Tương tự, *quota* nhóm (nếu được dùng) cũng được đặt trong tệp *quota.group* cũng trong thư mục gốc của hệ thống tệp. Ta phải tạo các tệp *quota* bằng tay bằng cách dùng cat để tạo một tệp hoặc dùng lệnh *touch*. Sau đây

Lệnh này kích hoạt một trình soạn thảo (mặc định là *vi*). Nếu ta đưa ra tên là các lệnh đặt quota cho hệ thống tệp */usr*.

```
$ cd /usr
$ touch quota.user
$ chmod 600 quota.user
$ touch quota.group
$ chmod 600 quota.group
```

Lệnh *chmod* làm cho tệp chỉ ghi được bởi root.

Lúc này chúng ta cần khởi động lại hệ thống, khi đó tệp *quota.user* hay *quota.group* sẽ được hệ thống soạn với các thông tin khởi đầu đều bằng 0 và công việc của chúng ta sẽ là sửa các thông tin này để đặt quota cho từng người dùng cụ thể. Tuy vậy tệp *quota.user* và *quota.group* đều không phải là các tệp ASCII, để sửa thông tin trong chúng ta phảI dùng lệnh *edquota* và lệnh này cũng chỉ có root mới dùng được. Tiếp theo *edquota* ta chỉ ra tên người dùng hoặc nhóm mà ta muốn đặt quota. Ví dụ :

```
/usr/sbin/edquota linhtd
```

Lệnh edquota sẽ bật trình soạn thảo mặc định của hệ thống (thường là vi) để chúng ta sửa quota. Ví dụ tệp quota như sau :

```
Quotas for user linhtd :
/dev/hda1 : blocks in use 1128, limits (soft = 50000, hard = 50000)
        inodes in use 112, limits (soft = 10000, hard = 10000)
```

Một tệp tạm sẽ lưu các thông tin về người dùng trong nhóm và viết thông tin này vào tệp *quota.group*.

Một số tuỳ chọn của lệnh *edquota* :

Bảng 2.8. Các tuỳ chọn của lệnh edquota

-g	Soạn quota của một nhóm
-p	Chép hoàn toàn một quota của người này cho người kia
-t	Sửa giới hạn thời gian mềm (trước khi đạt đến giới hạn cứng)
-u	Sửa một quota của người dùng (tuỳ chọn mặc định)

Sau khi đặt quota cho một hệ thống tệp, ta phải bật kiểm tra quota với lệnh *quotaon*. Ví dụ để bật quota cho hệ thống tệp */dev/sda3*:

```
$ quotaon /dev/sda3
```

để bật kiểm tra quota cho tất cả các phân vùng, ta dùng lệnh :

```
$ quotaon -a
```

Lệnh *quotaoff* thực hiện các hành động ngược lại với cùng tham số.

5. Dùng lệnh quota

Cú pháp của lệnh *quota* :

```
$ quota [ options] [ user] [ group]
```

Các tuỳ chọn cho lệnh quota :

Bảng 2.9. Tuỳ chọn của lệnh quota

-g	Hiển thị quota nhóm cho nhóm mà người dùng là thành viên
-q	Hiện thị các hệ thống tệp đã dùng quá giới hạn
-u	Hiển thị các quota của người dùng
-v	Hiển thị quota của hệ thống tệp chưa lưu trữ gì.

Người sử dụng cũng có thể tổ hợp các tuỳ chọn, ví dụ dùng cả -g và -u khi đó cả quota của người dùng và nhóm đều được hiển thị. Người quản trị hệ thống khi đăng nhập dưới root có thể hiển thị quota cho người dùng bất kì với tuỳ chọn -u kèm theo tên, ví dụ

```
$ quota -u tiennc
```

6. Dùng lệnh quotacheck

Khi đã đặt và chạy *quota*, người sử dụng có thể dùng lệnh *quotacheck* bất kì lúc nào để kiểm tra tình trạng sử dụng không gian hiện tại của hệ thống tệp. Kết quả sẽ được viết vào tệp *quota.user* hay *quota.group* trong thư mục gốc.

Nên đặt lệnh *quotacheck* để nó chạy thường xuyên mỗi khi khởi động, khi đó tệp quota luôn được cập nhật tự động, như vậy tốt nhất là đặt lệnh này trong tệp khởi động *rc*. Nếu không nên dùng *quotacheck* thường xuyên khi thấy có lỗi trong một hệ thống tệp.

Các tuỳ chọn của lệnh *quotacheck* :

Bảng 2.10. Tuỳ chọn của lệnh quotacheck

-a	Kiểm tra mọi hệ thống tệp
-d	Hiển thị các thông tin kiểm tra từng bước một (debug)
-g	Kèm theo một nhóm, chỉ kiểm tra nhóm đó.
-u	Kèm theo một userID, chỉ kiểm tra người dùng đó (kiểm tra tất cả người dùng nếu không chỉ ra người dùng).
-v	Đưa ra thông tin bài hơn, chỉ rõ quotacheck đang làm gì.

Sử dụng *quotacheck* thường xuyên có thể giúp đảm bảo quota của hệ thống tệp được tuân thủ.

7. Tệp liên kết

Tệp liên kết là một khái niệm thường gây nhầm lẫn trong hệ thống tệp nhưng thực ra nó rất đơn giản. Một cách đơn giản nhất, một liên kết tạo một tên tệp thứ hai cho một tệp. Có hai loại tệp liên kết, liên kết cứng (hard link) và liên kết biểu tượng (symbolique link).

Liên kết cứng

Liên kết cứng không thực sự là một kiểu tệp mà đúng hơn là một tên khác của tệp được đặt tại một vị trí khác trong hệ thống tệp. Mỗi tệp có ít nhất là một liên kết cứng, thường là tên mà nó được tạo ra ban đầu. Khi tạo một liên kết mới, tệp được đặt một tên alias mới. Liên kết và tệp gốc của nó là hai điểm *i-node* cùng trỏ đến một nội dung vật lí (do đó có cùng số *i-node*). Dưới UNIX một liên kết và

tệp mà nó trỏ tới được coi là một.

Ví dụ ta tạo một tệp liên kết của tệp */usr/nvviet/testfile* bằng lệnh sau :

```
$ ln /usr/nvviet/testfile /usr/toandv/testfile
```

Khuôn dạng của lệnh này luôn là tên tệp hiện tại rồi đến tên tệp liên kết.

Khi đó để biết số *i-node* của 2 tệp, ta dùng

```
$ ls -i /usr/nvviet/testfile /usr/toandv/testfile
14253 /usr/nvviet/testfile 14253 /usr/toandv/testfile
```

Cả hai tệp */usr/nvviet/testfile*, */usr/toandv/testfile* cùng trỏ đến một nội dung vật lí chung, vì vậy mọi thay đổi bởi `nvviet` hoặc `toandv` đều phản ánh ngay lập tức trên thư mục kia (việc này tránh cho chúng ta phải thường xuyên sao lại các tệp này). Cả `nvviet` và `toandv` đều có thể thay đổi tệp, miễn là không thay đổi cùng một lúc.

Một vấn đề là quyền truy nhập tệp và sở hữu tệp như thế nào. Nếu `nvviet` là người chủ tệp và là người duy nhất có quyền ghi tệp, anh ta có thể tạo một móc nối mới cho tệp của anh ta là */usr/toandv/testfile* và đặt quyền sở hữu của tệp liên kết mới cho `toandv`. Theo cách đó cả `nvviet` và `toandv` có thể làm việc trên cùng một tệp bất chấp các quyền sở hữu và truy nhập vì mỗi tệp có một người sở hữu riêng. Nếu được đặt đúng, quyền sở hữu và quyền truy nhập có thể chống người khác đọc và ghi lên tệp.

Việc xoá một tên tệp liên kết không xoá mất tệp vật lí tương ứng, tệp vật lí chỉ bị xoá khi nào không còn một liên kết nào đến nó nữa.

Liên kết biểu tượng

Liên kết biểu tượng là một kiểu liên kết khác không dùng điểm *i-node* cho liên kết, nó chỉ đơn giản là một tệp chứa tên của tệp mà nó liên kết đến. Khi UNIX mở hoặc đi theo một liên kết biểu tượng, nó sẽ đi thẳng tới tệp mà liên kết biểu tượng trỏ đến hơn là bản thân liên kết này. Điểm khác nhau cơ bản giữa liên kết cứng và liên kết biểu tượng là liên kết cứng là một tham chiếu trực tiếp còn liên kết biểu tượng là một tham chiếu bằng tên.

Ta đã sử dụng các liên kết này khi tạo các trình điều khiển thiết bị ngoại vi như */dev/modem* hay */dev/cua1*. Lựa chon -s của lệnh ln cho phép tạo liên kết biểu tượng. Ví dụ có tệp *bigfile*, ta tạo một liên kết biểu tượng của nó, tệp *anotherfile*.

```
$ ls -i bigfile
6253 bigfile
$ ln -s bigfile anotherfile
$ ls -i bigfile anotherfile
6253 bigfile 8358 anotherfile
```

Như ta thấy các số *i-node* của các tệp là khác nhau điũu này là đương nhiên vì 2 tệp là hoàn toàn khác nhau, chúng trỏ đến 2 nội dung vật lí khác nhau. Khi liệt kê tệp, các liên kết biểu tượng được biểu diễn bằng mũi tên

```
lrwxrwxrwx 1 root root 6 Sep 16:35 anotherfile -> bigfile
 -rw-rw-r-- 1 root root 2 Sep 17:23 bigfile
```

Quyền của tệp liên kết biểu tượng luôn được đặt là *lrwxrwxrwx*. Quyền truy nhập tệp liên kết biểu tượng được xác định bởi quyền truy nhập và sở hữu của tệp mà nó liên kết đến (ở đây là *bigfile*).

Điểm khác biệt giữa liên kết cứng và liên kết biểu tượng không chỉ là các điểm nhập *i-node* trong bảng *i-node*. Ta có thể tạo một liên kết biểu tượng đến một tệp chưa tồn tại, điều này không thể làm được với liên kết cứng. Ta có thể đi theo các liên kết biểu tượng để tìm ra tệp mà chúng trỏ tới nhưng hầu như không thể làm được với liên kết cứng. Hạt nhân Linux xử lí hai kiểu tệp này cũng khác nhau.

II. QUẢN LÍ IN ẤN

Quản lí máy in trong Linux khá khác với dưới DOS hay OS/2. Vì vậy nó có thể gây ra một số vấn đề đối với người quản trị hệ thống. Khả năng in của Linux cũng không mạnh và dễ dùng như các phiên bản thương mại khác của UNIX. Khi chúng ta làm việc trên một môi trường máy in mạng lớn, các hạn chế của Linux trong vấn đề in ấn càng bộc lộ.

1. Thêm máy in

Linux hỗ trợ cả máy in song song và tuần tự, cũng như máy in mạng (được nối với một máy trong mạng cục bộ). Hầu hết chúng đều làm việc theo chế độ kí tự (character mode). Mặc dù vậy, một số ít với tốc độ cao lại hoạt động trong chế độ khối (block mode). Linux cũng có một tiện ích tiện ích phục vụ cho cài đặt và cấu

hình máy in, song khả năng của nó rất hạn chế so với nhiều phiên bản khác của UNIX. Vì vậy, tốt hơn là tạo các tệp điều khiển thiết bị cho máy in bằng tay.

Các máy in song song được tham chiếu bằng các tệp điều khiển thiết bị */dev/lp0*, */dev/lp1*, */dev/lp2*, tuỳ theo số của cổng song song mà chúng sử dụng. Hầu hết các máy in được nối vào cổng song song của PC tương ứng với */dev/lp0*, cổng song song đầu tiên.

Bảng 2.11. Các cổng song song

Parallel port device	I/O address	DOS equivalent
/dev/lp0	0x03bc	LPT1
/dev/lp1.	0x0378	LPT2
/dev/lp2	0x0278	LPT3

Nếu không chắc địa chỉ của cổng song song mà máy in được cắm, ta hãy thử lần lượt các cổng song song từ */dev/lp0*, và xem xem liệu có thể in được từ cổng đó không.

Linux dùng lệnh *mknod* (make node) để tạo tệp thiết bị cho máy in song song. Để tạo tệp điều khiển thiết bị ở cổng song song thứ nhất (*/dev/lp0*) :

S mknod -m 620 /dev/lp0 c 6 0

Theo ví dụ này, tệp điều khiển thiết bị */dev/lp0* được tạo cho thiết bị hoạt động trong chế độ kí tự với số thiết bị chính là 6 và số thiết bị phụ là 0.

Để hiểu rõ, ta nói thêm một chút về hai con số này. Một hệ thống có thể có nhiều hơn một thiết bị ngoại vi cùng kiểu, do đó chúng dùng chung trình điều khiển. Ví dụ, hệ thống có hai máy in Hewlett Packard LaserJet được nối vào hai cổng song song cùng dùng chung trình điều khiển. Hệ điều hành phải có cách phân biệt hai máy in này. Một phương pháp là dùng số của thiết bị. Mỗi thiết bị được xác định bởi số chính (major number) xác định trình điều khiển sẽ được sử dụng, và một số phụ (minor number) xác định số của thiết bị. Như vậy hai máy in trên sẽ có cùng số chính (trỏ đến tệp điều khiển thiết bị trong thư mục */dev*), nhưng có số phụ khác nhau giúp cho hệ điều hành xác định máy in.

Thông thường số thiết bị phụ được đặt bắt đầu từ 0 và tăng dần lên. Vì máy in này là cái đầu tiên được đặt nên nó có số phụ bằng 0. Tuỳ chọn -m đặt mặt nạ quyền cho tệp (ở đây là 620).

Sau khi tạo tệp điều khiển cho máy in, ta phải đổi quyền làm chủ của tệp điều khiển này cho root, daemon hay `root.daemon` (đặt quyền làm chủ cho root và nhóm cho daemon cùng một lúc). Ta có thể dùng một lệnh với `root.daemon` :

```
$ chown root.daemon /dev/lp0
```

Sau đó đặt quyền truy nhập tệp bằng 620

```
$ chmod 620 /dev/lp0
```

Để đặt cấu hình cho thiết bị không phải ở cổng song song đầu tiên, ta phải đổi số thiết bị trong tên thiết bị. Với mỗi cổng song song có thể, lệnh *mknod* là :

```
$ mknod -m 620 /dev/lp0 c 6 0
$ mknod -m 620 /dev/lp1 c 6 1
$ mknod -m 620 /dev/lp2 c 6 2
```

Trong các trường hợp này số thiết bị phụ tăng tương ứng với số cổng. Mặc dù đánh số các thiết bị theo cách này là không cần thiết nhưng nó có thể giúp ta biết được ta cắm thiết bị vào cổng nào.

Ta cũng cần phải tạo một thư mục *spool* cho máy in. Quyền truy nhập và quyền sở hữu cho thư mục *spool* sẽ được nói đến trong phần sau, "tệp */etc/printcap*".

2. Chương trình lpd

Các dịch vụ in được quản lí bởi một daemon gọi là *lpd* (line printer daemon). Daemon *lpd* thường được khởi động tự động trong quá trình khởi động */etc/rc* khi hệ thống chuyển sang chế độ đa người dùng. *Lpd* daemon gồm một số nhiệm vụ và chạy trong suốt quá trình Linux hoạt động (trừ phi bị kết thúc bởi người quản trị hoặc khi daemon bị hỏng). Một trong những phần quan trọng nhất của thủ tục khởi động của daemon này là đọc tệp cấu hình máy in, */etc/printcap*.

Tệp */etc/printcap* dùng để xác định các lệnh phục vụ giao tiếp với tất cả các máy in đã được cấu hình và nối vào hệ thống. Khi đã tự khởi động, *lpd* khởi động hai daemon khác gọi là *listen* và *accept* quản lí các yêu cầu in gửi đến.

Daemon của Linux chạy không ổn định lắm, vì vậy người sử dụng có thể phải khởi động lại hoặc kết thúc nó khi thay đổi cấu hình. Để khởi động daemon, sử dụng cú pháp:

```
$ lpd [ -l] [ port]
```

Tuỳ chọn -l khởi động một tiến trình *log*, nó ghi một ghi chú vào tệp *log* mỗi khi có một yêu cầu in được xử lí.

Tuỳ chọn *port* cho phép chúng ta chỉ ra số cổng Internet cho daemon nếu muốn bỏ qua thông tin mặc định của hệ thống. Chúng ta có thể sẽ không bao giờ phải sử dụng tuỳ chọn này trên một máy đơn hoặc mạng nhỏ, nhưng nó sẽ rất hữu dụng trong các hệ thống máy in của mạng rất lớn.

Khi nhận được một yêu cầu in qua mạng (hoặc ngay tại máy), daemon *lpd* thực hiện một chương trình kiểm tra ngắn để xem người gửi yêu cầu có được phép dùng máy in hay không. Chương trình này dùng các tệp */etc/hosts.equiv* và file */etc/hosts.lpd*. Nếu tên máy của người gửi không nằm trong một trong hai tệp trên, yêu cầu in không được đáp ứng. Máy cục bộ của ta luôn nằm trong *hosts.equiv* (dưới tên *localhost*), nên mọi người dùng trên máy đều có thể in được.

Để kết thúc daemon *lpd*, lấy số hiệu tiến trình của nó (process ID) sử dụng lệnh *ps*, và sau đó sử dụng lệnh *kill* với số hiệu tiến trình đó. Khi đó các yêu cầu in đều không được chấp nhận.

2.1. Spool của máy in

Khi một yêu cầu in (một *print job*) được *lpd* nhận (hoặc nhận bởi các tiến trình *listen* và *accept* của nó), các trang được in sẽ được chép vào một vùng, gọi là vùng *spool* in của hệ thống. Hành động này sẽ giải phóng *console* khi ta đưa ra yêu cầu in và cho phép ta tiếp tục thay đổi nội dung các tệp mà ta muốn in sau khi chúng đã được gửi cho daemon mà không ảnh hưởng đến nội dung in ra.

Nói chung vùng *spool* in nằm trong thư mục */usr/spool/lp*. Mỗi máy in được cài đặt đều có một thư mục nằm trong thư mục *spool*. Ví dụ, máy in *hplaser* dùng thư mục */usr/spool/hplaser*. Mọi yêu cầu in cho mỗi máy in đều nằm trong thư mục của nó tương ứng với một tệp có tên duy nhất và một số hiệu yêu cầu in duy nhất. Daemon cho máy in này sẽ thêm số hiệu yêu cầu in vào một hàng đợi và thông báo cho ta biết số hiệu này là bao nhiêu. Người sử dụng có thể dùng số hiệu này để kiểm tra trạng thái của yêu cầu in và bỏ yêu cầu in khỏi hàng đợi.

Một số phiên bản của Linux cho phép đặt kích thước của vùng *spool* in thông qua tệp *minfree* trong thư mục *spool*. Tệp *mintree* đưa ra số *block* đĩa (thường 1block = 1K) dành cho *spool*, tệp này hoàn toàn có thể sửa được nhờ bất kì chương trình soạn thảo ASCII nào. Nếu có nhiều không gian đĩa, người sử dụng không cần quan tâm đến con số này vì *spool* sẽ dùng không gian trống cần thiết.

Kích thước của *spool* nên đặt tuỳ theo số lượng người dùng và lượng thông tin in, nói chung nên đặt 100K cho mỗi người dùng bình thường.

Mỗi thư mục *spool* có thể có 2 tệp đặc biệt gọi là *status* và *log* chứa các mô tả trạng thái hiện tại của máy in. Chúng được tạo và quản lí bởi daemon *lpd*, một số các lệnh máy in dùng chúng để hiển thị các thông tin trạng thái. Các tệp này đều có thể sửa bằng các trình soạn thảo ASCII.

2.2. Quá trình in

Như vậy daemon máy in làm việc và Linux xử lí các giai đoạn yêu cầu như thế nào. Khi ta đưa ra một yêu cầu in bằng một lệnh in (ví dụ *lpr*), lệnh này sẽ sinh ra đầu ra để máy in in, đầu ra được chép vào hàng đợi trong thư mục *spool* của máy in mà ta vừa yêu cầu.

Ta có thể chỉ ra máy in trên dòng lệnh hoặc đặt một máy in mặc định như một biến môi trường, nhờ đó hệ thống luôn biết phải dùng máy in nào. Sau khi xác định xong tên máy in đích, *lpr* đọc tệp */etc/printcap* lấy thông tin cấu hình máy in

Lpr tìm các lệnh đặc biệt chỉ ra cách in tệp. Các lệnh này có thể chỉ font, cỡ, màu, ngôn ngữ xử lí hoặc bất kì một thông tin cấu hình máy in nào khác. Các lệnh này có thể lấy từ dòng lệnh (dưới dạng các tham số mà ta đưa ra cùng với lệnh in), từ các biến môi trường (đặt bởi các tệp khởi động shell hoặc do ta tự đặt), hoặc từ các giá trị mặc định của hệ thống.

Khi yêu cầu in được chép vào thư mục *spool*, *lrp* tạo ra 2 tệp. Một tệp có chữ *cf* (tệp điều khiển) nối tiếp bởi số hiệu in của yêu cầu in đó. Tệp này gồm thông tin về yêu cầu in bao gồm tên người yêu cầu in và các lệnh in đặc biệt như khoảng cách dòng hoặc chọn giấy. Tệp còn lại bắt đầu bằng chữ *df* (tệp dữ liệu) chứa nội dung tệp cần in. Sau khi *lpr* đã tạo tệp *df*, nó gửi tín hiệu đến daemon *lpd* chỉ ra rằng có một yêu cầu in đang chờ trong *spool*. Khi đó daemon *lpd* sẽ kích hoạt một daemon để quản lí hàng đợi in (nếu daemon này chưa được kích hoạt). Chừng nào hàng đợi này rỗng, daemon này mới kết thúc.

Khi *lpd* nhận được tín hiệu in từ *lpr*, nó sẽ kiểm tra tệp */etc/printcap* xem máy in được nối trực tiếp hay là một máy in ở xa. Nếu là máy in ở xa tức là được nối vào một máy khác trong mạng, *lpd* tạo một liên kết mạng đến máy đó và chuyển toàn bộ tệp điều khiển và dữ liệu đến thư mục *spool* của máy đó và báo cho *lpd* của máy đó rằng đang có yêu cầu in trong hàng đợi. Để kết thúc quá trình in từ xa, *lpd* xoá các bản sao của tệp *cf* và *df* tại máy yêu cầu in. Với các yêu cầu in tại

chỗ, *lpd* kiểm tra chắc chắn rằng có máy in và đã sẵn sàng, sau đó gửi yêu cầu in cho daemon đang điều khiển máy in đó. Khi các tệp đã in xong, chúng được xóa khỏi thư mục *spool*.

2.3. Tệp etc/printcap

Ta đã thấy là tệp */etc/printcap* được cả lệnh in (như *lpr*) và daemon *lpd* sử dụng. Nó gồm thông tin về tất cả các máy in truy nhập được, cả máy in ở xa đã được cấu hình lẫn máy in được nối trực tiếp. Ví dụ trích từ một tệp */etc/printcap* cho máy in laser Hewlett Packard LaserJet 4M.

```
# HP Lacerjet
lp|hplj|hplaserj-tiennc|HP LaserJet 4M ben canh voi nuoc:\
 :lp=/dev/lp0:\
 :sd=/usr/spool/lp/lp0:\
 :lf=/usr/spool/lp/errorlog:\
 :mx#0:\
 :of=/usr/spool/lp0/hpjlp:\
```

Cú pháp của tệp này như sau :

Trường đầu tiên là danh sách các tên mà người dùng có thể sử dụng để chỉ máy in này. Các tên được phân cách bởi dấu gạch đứng. Thông thường có ít nhất 3 tên : Tên ngắn, từ 4 kí tự trở xuống (ví dụ : `hplj`), tên đầy đủ hơn với tên người chủ (ví dụ hplaser-tiennc), và một tên mô tả đầy đủ với các thông tin cần thiết xác định máy in cho một người dùng (như `HP LaserJet 4M ben canh voi nuoc`). Trong đó tên `lp`, tên máy in mặc định của Linux được đặt đầu tiên để tránh các thông báo lỗi khi ta không chỉ định máy in trong lệnh in.

Trường thứ hai là tập các tham số 2 kí tự và các giá trị định nghĩa thông tin cấu hình máy in. Khuôn dạng của trường này tuân theo một trong các kiểu sau :

- NN — Một giá trị kiểu boolean
- NN=string — Đặt bằng một xâu
- NN#number — Đặt khác một số.

Hầu hết các phép gán được đặt trong 2 bằng dấu ":"

Có nhiều mã trong tệp */etc/printcap*. Sau đây là một số tham số quan trọng thường thấy :

Bảng 2.12. Các tham số của /etc/printcap

sd	Thư mục *spool*
lf	Thư mục *log* cho các thông báo lỗi
af	Tệp log tính toán, ghi số lần in của một người dùng
mx	Kiểu tệp có thể in được, mx#0 chỉ ra rằng không hạn chế kiểu tệp.
of	Các chương trình lọc dùng khi in

Khi thêm một máy in, ta có thể phải tạo thư mục *spool* bằng tay. Thư mục này phải được đặt quyền sở hữu là root hoặc daemon.

3. Quản lí máy in bằng lpc

Linux điều khiển máy in qua một tiện ích gọi là *lpc*, đó là chương trình cho phép chúng ta thực hiện một số chức năng liên quan đến máy in trong hệ thống Linux.

- Hiển thị thông tin trạng thái máy in
- Cho phép in hoặc không cho phép in
- Cho phép hoặc không cho phép hàng đợi hoạt động
- Xoá mọi yêu cầu in trong hàng đợi máy in
- Đẩy một yêu cầu in đặc biệt lên đầu hàng đợi
- Thay đổi daemon *lpd*

Chương trình *lpc* chỉ được sử dụng với các máy in được nối trực tiếp vào máy, không thể áp dụng nó cho các máy ở xa. Để thực hiện trên các máy ở xa, ta phải đăng nhập vào máy đó dưới tài khoản root rồi thực hiện các thay đổi cần thiết. Cũng phải lưu ý rằng *lpc* là chương trình hoạt động tương đối lạ, nó có thể bị treo khi không có lỗi nào xảy ra.

Khi thực hiện *lpc* không tham số, *lpc* yêu cầu ta đưa ra lệnh. Sau đây là các lệnh có thể, (dấu gạch đứng thể hiện một lựa chọn các tham số):

- *abort tên_máy_in | all*. Lệnh này tương tự như lệnh *stop* chỉ khác là nó không cho phép in nốt yêu cầu in hiện tại trước khi dừng máy in. Khi dùng lệnh này với tham số all, tất cả các máy in bị dừng. Khi máy in

được bật lại, mọi yêu cầu in bị dừng bởi lệnh *abord* sẽ lại được đưa vào hàng đợi.

- *clean tên_máy_in / all*. Xoá mọi yêu cầu in khỏi hàng đợi và cả yêu cầu in đang thực hiện. (Nhiều khi yêu cầu in hiện tại vẫn được thực hiện vì nó đã được chuyển cho daemon của máy hoặc vùng đệm RAM của máy in và không thể bị dừng bởi *lpc*). Nếu dùng với tham số *all*, mọi hàng đợi của các máy in sẽ bị xoá.

- *disable tên_máy_in / all*. Lệnh này vô hiệu hoá các hành động đưa yêu cầu in vào hàng đợi đối với một máy in (hoặc tất cả các máy in nếu dùng tham số all). Mọi yêu cầu in đã nằm trong hàng đợi đều mất tác dụng. Bất kì người dùng nào cố gửi các yêu cầu in đến máy in đã bị vô hiệu hoá đều nhận được một thông báo máy in không hoạt động và yêu cầu bị từ chối. Máy in được đặt hoạt động trở lại hoặc vô hiệu nhờ thay đổi trong tệp *lock* trong thư mục *spool*.

- *down tên_máy_in thông_báo*. Lệnh này dùng để ngắt máy in khỏi hệ thống. Thông báo sẽ được đặt vào tệp trạng thái của thư mục *spool* và hiển thị cho người dùng đang thử xếp yêu cầu in vào hàng đợi. Lệnh này dùng khi máy in gặp phải vấn đề nghiêm trọng cần ngắt khỏi hệ thống.

- *enable tên_máy_in / all*. Cho phép lại xếp hàng các yêu cầu in cho máy in (hoặc tất cả máy in) sau khi dừng.

- *exit*. Thoát khỏi *lpc*, giống *quit*.

- *Help* hoặc *?*. Hiển thị ngắn gọn danh sách tất cả các lệnh của *lpc*. Nếu gõ một lệnh *lpc* sau lệnh *help*, hệ thống sẽ hiển thị mô tả trên 1 dòng lệnh vừa gõ.

- *restart tên_máy_in / all*. Khởi động lại daemon của một máy in hoặc của tất cả các máy in nếu dùng tham số *all*

- *start tên_máy_in*. Khởi động daemon hàng đợi của máy in được chỉ ra, cho phép nó in các yêu cầu.

- *status tên_máy_in*. Hiển thị các thông tin: tên máy in, hàng đợi *spool* ở trạng thái hoạt động (enabled) hay không, máy in ở trạng thái hoạt động hay không, số các yêu cầu in trong hàng đợi, trạng thái của daemon cho

máy in đó. Nếu không có yêu cầu nào trong hàng đợi thì sẽ không có daemon nào hoạt động. Nếu có yêu cầu in trong hàng đợi mà không thấy có daemon nào thì tức là tiến trình daemon đã chết và cần phải khởi động lại daemon với lệnh *restart*.

- *stop tên_máy_in*. Lệnh này dừng máy in. Các yêu cầu in có thể vẫn ở trong *spool*, nhưng không được in cho đến khi máy in được khởi động. Nếu có một yêu cầu đang được in khi có lệnh *stop*, máy in sẽ dừng sau khi in xong. Lệnh này sẽ kết thúc daemon phục vụ xếp hàng các yêu cầu in. Lệnh *start* và *stop* thay đổi nội dung tệp *log* trong thư mục *spool*.
- *topq tên máy_in số_hiệu_in*. Chuyển yêu cầu in với yêu cầu in có số hiệu đã chỉ ra lên đầu hàng đợi.
- *topq tên máy in tên người_dùng*. Chuyển tất cả các yêu cầu in của người dùng có tên đã chỉ ra lên đầu hàng đợi.
- *up printer name*. Kích hoạt lại một máy in đã được ngắt bằng lệnh *down*.

Lpc là tiện ích không thân thiện lắm với người dùng nhưng nó là cách duy nhất để quản lí máy in và các hàng đợi của nó trong Linux.

4. Quản lí hàng đợi máy in bằng lpq và lprm

Thay vì hoàn toàn tin tưởng vào lệnh *lpc*, ta có thể dùng một số lệnh cho phép quản trị trực tiếp trên hàng đợi máy in. Các lệnh này được thiết kế để làm đơn giản 2 nhiệm vụ thường được người quản trị yêu cầu : hiển thị trạng thái hàng đợi hiện tại và xoá bỏ các nhiệm vụ in khỏi hàng đợi.

Để hiển thị hàng đợi in hiện tại của một máy in, dùng lệnh *lpq* với cú pháp :

```
lpq [ -l] [ -Pprinter_name] [ job_ID ...] [ username ...]
```

Nếu không có tham số, *lpd* hiển thị thông tin về hàng đợi in hiện tại: ai đang xếp hàng yêu cầu in, ở đâu trong hàng đợi, các tệp đang in, kích thước tổng cộng của các tệp này. Tuỳ chọn -l hiển thị thông tin chi tiết hơn cho mỗi yêu cầu in ..

Tuỳ chọn -P nối tiếp bởi tên máy in cho phép hiển thị thông tin về một máy in, hoặc máy in mặc định nếu không đưa ra tên máy in. Nếu đưa ra một hoặc nhiều số hiệu yêu cầu in (job_ID) hoặc tên người dùng, thì chỉ các thông tin về yêu cầu in đó hoặc các yêu cầu in đang được xếp hàng bởi người dùng đã chỉ ra là được hiển thị.

Để xoá một tệp khỏi hàng đợi, dùng lệnh *lprm*. Lệnh này đòi hỏi số hiệu in (job_ID).

```
lprm [ -Pprinter_name] [ -] [ job_ID ...] [ username ...]
```

Nếu tham số một gạch ngang "-" được dùng, *lprm* xoá tất cả các yêu cầu in đang xếp hàng của người đưa ra lệnh *lprm*, nếu đang ở trong đăng nhập root, mọi yêu cầu in sẽ bị xoá.

Người sử dụng có thể bỏ các yêu cầu in của một máy in xác định bằng tuỳ chọn -P. Ví dụ :

```
$ lprm -Phplj -
```

sẽ xoá mọi yêu cầu in trong hàng đợi của máy in *hplj* của người dùng đưa ra lệnh, hoặc mọi yêu cầu in trên máy in đó nếu đang có quyền root. Nếu có các tham số về số hiệu in hay tên người dùng , *lprm* sẽ xoá yêu cầu in được chỉ ra hoặc mọi yêu cầu in của người dùng được nêu ra. Nếu không có tham số, yêu cầu in hiện thời của người dùng bị xoá.

Nếu người sử dụng cố dùng lệnh *lprm* trên một yêu cầu in đang được in thì có thể không dừng được yêu cầu in đó vì tệp cần in có thể đã nằm hoàn toàn trên vùng đệm máy in. Trong một số trường hợp khác có thể dẫn đến khoá máy in.

III. QUẢN LÍ TIẾN TRÌNH

Phần này chúng ta sẽ xem xét cách tìm ra các tiến trình đang chạy trong hệ thống, xác định chúng đang làm gì cũng như việc sử dụng các thông tin này để quản lí các tiến trình.

1. Các tiến trình

Một tiến trình là một chương trình hoạt động một cách độc lập, có một tập hợp các tài nguyên riêng. Theo cách định nghĩa đó thì mọi thứ chạy trên hệ thống Linux đều là các tiến trình. Một dòng lệnh được đưa ra tại dấu nhắc của shell có thể gồm một hoặc nhiều tiến trình, đặc biệc khi chúng được nối nhau theo kiểu ống nước (pipe) hoặc có định hướng đầu ra hoặc đầu vào tiến trình.

Một số kiểu tiến trình trong hệ điều hành Linux :

- Tiến trình tương tác: Là một tiến trình được điều khiển bởi shell, có thể chạy ở mức *foreground* hoặc mức *background*.
- Tiến trình *batch* : Là tiến trình không liên kết với một *terminal* nào nhưng được đưa vào một hàng đợi để thực hiện một cách tuần tự.
- Tiến trình *daemon* : Là tiến trình chạy trên *background* cho đến khi nó được yêu cầu. Kiểu tiến trình này thường được khởi tạo khi khởi động Linux.

2. Lệnh ps

Cách dễ nhất để tìm ra tiến trình nào đang chạy là dùng lệnh *ps* (process status). Mọi người dùng đều có thể sử dụng lệnh này. Dưới một đăng nhập thông thường, ta đưa ra lệnh *ps* thì nó sẽ hiển thị tất cả các thông tin về các tiến trình mà ta đang chạy. Ví dụ:

```
$ ps
PID TTY STAT TIME COMMAND
41 v01 S 0:00 -bash
134 v01 R 0:00 ps
```

Thông tin mà *ps* đưa ra được biểu thị dưới dạng các dòng với mỗi tiến trình, thông tin trong từng dòng được chiếu theo các cột.

Cột thứ nhất PID là số hiệu tiến trình do Linux gán cho mỗi tiến trình để quản lí chúng. Các tiến trình được đánh số lần lượt từ 0 đến một con số được định ra bởi hệ thống (ví dụ 65564). Khi đã đánh đến số lớn nhất, nó lại quay lại dùng từ số nhỏ nhất, mà đang không bị sử dụng cho một tiến trình đang hoạt động, để đánh số. Thông thường các tiến trình có số nhỏ là các chương trình của hệ thống hoặc daemon.

Cột TTY cho biết tiến trình được kích hoạt từ *terminal* nào. Nếu người sử dụng đã chạy nhiều cửa sổ *console*, lệnh *ps* sẽ hiển thị tất cả các tiến trình do người sử dụng đó kích hoạt trên các cửa sổ đó.

Cột STAT cho thấy trạng thái hiện tại của tiến trình. Các trạng thái có thể là S : đang ngủ (tức là tiến trình hiện tại không hoạt động), R : đang chạy (tiến trình đang thực hiện trong CPU). Các tiến trình có thể thường xuyên chuyển đổi giữa trạng thái ngủ và chạy nhiều lần.

Cột TIME đưa ra thời gian sử dụng CPU tổng cộng của tiến trình cho đến

thời điểm hiện tại.

Cột NAME chứa tên của các tiến trình đang chạy. Thông thường tên này là tên lệnh mà người sử dụng đã gõ. Tuy vậy một số lệnh lại kích hoạt các tiến trình khác, các tiến trình này được gọi là tiến trình con, chúng được thể hiện trong *ps* như thể đã được thực hiện bởi câu lệnh của người sử dụng.

Theo quy ước chung, các shell được kích hoạt ngay khi đang nhập hệ thống sẽ được biểu thị trong đầu ra của lệnh *ps* bằng một gạch ngang trước tên để phân biệt chúng với các shell mà ta kích hoạt sau đó. Ví dụ :

```
$ ps
PID TTY STAT TIME COMMAND
46 v01 S 0:01 -bash
75 v01 S 0:00 phksh
96 v01 R 0:00 bash
123 v01 R 0:00 ps
```

Theo đó, bash (PID 46) là shell được kích hoạt tự động ngay từ khi đăng nhập còn Korn shell (pdksh, PID 75) và Bourne shell (bash, PID 96) được người dùng kích hoạt sau. Tiến trình *ps* luôn luôn xuất hiện trên đầu ra của nó vì nó là đang chạy khi ta đưa ra lệnh.

Để đọc thông tin đầu ra, đôi khi quá dài, ta có thể nối tiếp lệnh *ps* với *more*, hoặc định hướng đầu ra của *ps* ra một tệp.

```
$ ps | more
$ ps > /tmp/ps_file
```

Lệnh *ps* có một số tuỳ chọn và tham số hay được dùng. Để kiểm tra xem ai là người kích hoạt và sở hữu các tiến trình ta dùng tuỳ chọn -u, nó sẽ thêm cột USER vào đầu ra của *ps*. Ví dụ sau đây được thực hiện dưới tên người dùng thông thường (không phải là root).

```
$ ps -u
USER PID %CPU %MEM SIZE RSS TTY STAT START TIME COMMAND
nvviet 41 0.1 6.8 364 472 v01 S 23:19 0:01 -bash
nvviet 138 0.0 3.3 72 228 v01 R 23:34 0:00 ps -u
```

Tuỳ chọn này đồng thời cũng thêm vào phần trăm sử dụng CPU và phần trăm sử dụng bộ nhớ (%MEM) của tiến trình cho đến nay.

Nếu lệnh *ps* được đưa ra dưới đăng nhập root, người sử dụng sẽ thấy tất cả các tiến trình trong hệ thống vì root sở hữu tất cả những gì đang chạy. Với tuỳ chọn u kém theo tên một người dùng, ta sẽ chỉ thấy các tiến trình chạy bởi người dùng đó. Ví dụ

```
$ ps -u chinq
```

Hầu hết mọi người dùng đều có thể dùng lệnh này để xem các tiến trình của người khác.

Người dùng cũng có thể xem tất cả các tiến trình chạy trong hệ thống (thay vì chỉ tiến trình mà họ kích hoạt) bằng tuỳ chọn -a.

```
$ ps -a
```

Để có thể biết cả ai là người đưa ra tiến trình nào ta kết hợp với tuỳ chọn -u. Thứ tự các tuỳ chọn không quan trọng có thể là -au hoặc -ua. Ví dụ :

```
$ ps -au
USER PID %CPU %MEM SIZE RSS TTY STAT START TIME COMMAND
root 1 0.0 3.0 44 208 psf S 23:19 0:00 init
root 6 0.0 1.8 24 128 psf S 23:19 0:00 update (sync)
root 23 0.0 3.0 56 212 psf S 23:19 0:00 /usr/sbin/crond -110
root 29 0.0 3.4 61 236 psf S 23:19 0:00 /usr/sbin/syslogd
root 31 0.0 2.8 36 200 psf S 23:19 0:00 /usr/sbin/klogd
root 33 0.0 2.9 64 204 psf S 23:19 0:00 /usr/sbin/lpd
root 40 0.0 2.0 32 140 psf S 23:19 0:00 selection -t ms
root 42 0.1 6.9 372 480 v02 S 23:19 0:01 -bash
root 43 0.0 2.3 37 164 v03 S 23:19 0:00 /sbin/agetty 38400 tt
root 44 0.0 2.3 37 164 v04 S 23:19 0:00 /sbin/agetty 38400 tt
root 45 0.0 2.3 37 164 v05 S 23:19 0:00 /sbin/agetty 38400 tt
root 46 0.0 2.3 37 164 v06 S 23:19 0:00 /sbin/agetty 38400 tt
chinq 41 0.0 6.8 364 472 v01 S 23:19 0:01 -bash
chinq 2519 0.0 3.4 80 236 v01 R 23:39 0:00 ps -ua
```

Trong ví dụ này, hầu hết các tiến trình được liệt kê đều là các tiến trình của hạt nhân Linux hoặc daemon, chỉ có 2 tiến trình cuối là được kích hoạt bởi người sử dụng.

Tuỳ chọn -l thêm thông tin các tiến trình nào kích hoạt tiến trình nào (điều này rất có ích khi ta muốn xác định các tiến trình con) :

```
$ ps -l
 F UID PID PPID PRI NI SIZE RSS WCHAN STAT TTY TIME COMMAND
 0 501 41 1 15 0 364 472 114d9c S v01 0:00 -bash
 0 501 121 41 29 0 64 208 0 R v01 0:00 ps -l
```

Cột PPID (Parent Process ID) đưa ra tiến trình cha đã kích hoạt tiến trình đó. Ví dụ trên cho thấy lệnh *ps* được kích hoạt từ tiến trình *bash*, vì shell là tiến trình cha của tất cả các lệnh của người dùng. PPID của Bourne shell là 1, là tiến trình *init* của hệ điều hành. (Mối quan hệ này cho thấy nếu *init* kết thúc thì tất cả các tiến trình khác cũng kết thúc).

3. Dùng lệnh kill

Một tiến trình làm một máy bị treo hoặc tiến trình không làm gì cả được gọi là tiến trình treo. Đôi khi người dùng có một số tiến trình không kết thúc đúng cách, loại tiến trình này gọi là tiến trình chạy trốn (runaway process). Để loại bỏ cả hai loại tiến trình này và trả lại cho hệ thống trạng thái bình thường, người sử dụng có thể dùng lệnh *kill*.

Để sử dụng lệnh *kill*, người sử dụng phải truy nhập vào một cửa sổ hoặc một *console* khác để có thể ra lệnh. Nếu máy hoàn toàn bị treo, người sử dụng phải tìm một máy khác và đăng nhập vào. Nếu đăng nhập dưới một người dùng thông thường, người sử dụng chỉ có thể loại bỏ được các tiến trình của chính mình. Khi đăng nhập với tài khoản root người sử dụng có thể loại bỏ bất kì tiến trình nào.

Để loại bỏ một tiến trình, người sử dụng phải biết PID của nó (bằng lệnh *ps* chẳng hạn), sau đó dùng lệnh *kill* với tham số PID tìm được. Ví dụ loại bỏ tiến trình *badProg* được kích hoạt bởi *sonnt* đã bị treo ;

```
$ ps -u sonnt
USER PID %CPU %MEM SIZE RSS TTY STAT START TIME COMMAND
sonnt 561 0.1 6.8 364 472 v01 S 13:19 0:01 -bash
sonnt 598 9.3 4.1 2736 472 v01 R 15:26 2:01 badProg
$ kill 598
```

Một điều lưu ý là khi một tiến trình bị kết thúc, các tiến trình con của nó cũng bị kết thúc theo.

Lệnh *kill* không có thông báo nếu nó hoạt động tốt, vì để kiểm tra là tiến

trình đã bị loại bỏ ta phải dùng lệnh *ps* lần nữa và tìm đến PID hoặc tên tiến trình.

Lệnh *kill* có một vài mức độ thực hiện. Thông thường lệnh *kill* không tham số sẽ loại bỏ tiến trình một cách nhẹ nhàng (đóng tệp đang mở và loại bỏ tiến trình). Nếu lệnh này không hoạt động, dùng tuỳ chọn -9 để thực hiện một loại bỏ bắt buộc.Ví dụ, loại bỏ tiến trình có PID là 726.

```
$ kill -9 726
```

Nếu vẫn không kết thúc được tiến trình, ta dùng tuỳ chọn -15, hình thức mạnh nhất của lệnh *kill*. Nhưng chỉ nên dùng lệnh này khi các dạng khác của *kill* không hoạt động, bởi vì nó thực hiện không nhẹ nhàng chút nào đối với các tiến trình và các tệp đang mở. Ví dụ của tuỳ chọn này :

```
$ kill -15 726
```

Nếu nó vẫn không hoạt động thì có lẽ là tiến trình này không thể loại bỏ được, giải pháp duy nhất là tắt và khởi động lại máy.

Để tránh người này loại bỏ tiến trình của người kia, *ps* kiểm tra quyền sở hữu tiến trình khi ta đưa ra lệnh *kill*. Trong trường hợp một người dùng cố loại bỏ một tiến trình của người khác, thông báo sau sẽ hiển thị

kill: - Not owner

Đương nhiên siêu người dùng không gặp thông báo này vì họ có thể loại bỏ mọi tiến trình trừ một số tiến trình hệ thống (như *init*).

4. Lệnh top

Đôi khi ta phải xem cách hệ thống xử lí với các vấn đề rắc rối, quản lý tải của hệ thống (monitor system loading), kiểm tra các tiến trình, thay vì liên tục dùng lệnh *ps*, Linux cung cấp lệnh *top*. Kết quả của lệnh *top* là các hình ảnh nhanh của hệ thống sau 5 giây một (ta có thể sửa tham số này). Một cách mặc định *top* đưa ra các nhiệm vụ dùng nhiều CPU nhất trong hệ thống cho đến khi đầy màn hình.

Cú pháp của *top* như sau:

```
top [ -] [ d delay] [ q] [ S] [ s] [ i]
```

Các tuỳ chọn trên dòng lệnh của *top* :

Bảng 2.13. Các tuỳ chọn của top

d	Chỉ ra độ trễ giữa các lần cập nhật màn hình
q	Bắt buộc *top* cập nhật lại mà không chờ tới thời gian trễ
S	Sử dụng chế độ tích lũy
s	Chạy *top* trong chế độ bảo đảm (cấm mọi lệnh tương tác)
i	Bỏ qua các tiến trình không làm gì cả.

Đầu ra của lệnh *top* đưa ra một vài dòng tóm tắt ở đầu màn hình, tiếp theo sau là danh sách các tiến trình dùng CPU với cường độ cao nhất :

```
1:58pm up 59 min, 2 users, load average: 0.13, 0.34, 0.98
26 processes: 25 sleeping, 1 running, 0 zombie, 0 stopped
CPU states: 0.9% user, 6.4% system, 0.0% nice, 92.7% idle
Mem: 14620K av, 6408K used, 8212K free, 4632K shrd, 2328K
buff
Swap: 0K av, 0K used, 0K free
PID USER PRI NI SIZE RES SHRD STAT %CPU %MEM TIME COMMAND
236 root 19 0 93 316 344 R 7.3 2.1 0:00 top
1 root 1 0 48 232 308 S 0.0 1.5 0:00 init
63 root 2 0 388 556 572 S 0.0 3.8 0:00 -bash
209 root 1 0 98 320 356 S 0.0 2.1 0:00 in.telnetd
24 root 1 0 60 228 296 S 0.0 1.5 0:00 /usr/sbin/crond -110
K
6 root 1 0 36 164 336 S 0.0 1.1 0:00 bdflush (daemon)
7 root 1 0 36 168 340 S 0.0 1.1 0:00 update (bdflush)
38 root 1 0 73 280 332 S 0.0 1.9 0:00 /usr/sbin/syslogd
40 root 1 0 44 240 320 S 0.0 1.6 0:00 /usr/sbin/klogd
42 bin 1 0 84 240 320 S 0.0 1.6 0:00 /usr/sbin/rpc.portmap
44 root 1 0 76 292 320 S 0.0 1.9 0:00 /usr/sbin/inetd
46 root 1 0 68 212 304 S 0.0 1.4 0:00 /usr/sbin/lpd
51 root 1 0 116 280 376 S 0.0 1.9 0:00 /usr/sbin/rpc.nfsd
```

Tiếp theo sau *uptime* là 3 số tiến trình trung bình chạy trong 1 phút, 5 phút, 15 phút trước.

Số tổng cộng các tiến trình chạy trong thời gian lấy hình ảnh nhanh của hệ thống được đưa ra trong dòng thứ hai, tiếp theo là số tiến trình đang ngủ (không hoạt động), ở trạng thái không rõ hoặc không tồn tại, và đã bị dừng.

Dòng thứ 3, trạng thái của CPU, cho thấy phần trăm sử dụng CPU trong chế độ người dùng, chế độ hệ thống, chế độ công việc chính xác, chế độ không làm gì. (Các công việc chính xác được Linux tính gồm cả công việc của người dùng và của hệ thống, vì vậy giá trị tổng cộng của các phần trăm này có thể lớn hơn 100%)

Dòng thứ 4, đưa ra tình trạng sử dụng bộ nhớ, bao gồm lượng bộ nhớ , bộ nhớ sẵn có, lượng còn rỗi, lượng đã dùng, lượng được chia sẻ, lượng được dùng cho vùng đệm.

Dòng thứ 5, đưa ra thống kê tình trạng dùng không gian swap của hệ thống : tổng không gian *swap*, còn trống, và đã sử dụng.

Tiếp theo là danh sách các tiến trình dùng CPU với cường độ cao nhất, có cấu trúc giống đầu ra của lệnh *ps*.

Khi đang chạy lệnh *top*, người sử dụng có thể đưa ra một số lệnh để thay đổi cách xử lí của nó (trừ phi người sử dụng đã kích hoạt *top* với tuỳ chọn -s để cấm các lệnh tương tác. Có các lệnh tương tác sau :

Bảng 2.14. Các phím tắt của top

^L	Vẽ lại màn hình
h/?	Hiển thị trợ giúp
k	Loại bỏ một tiến trình
I	Bỏ qua tiến trình không làm gì hoặc trạng thái không rõ ràng.
n/#	Thay số lượng tiến trình được hiển thị
q	Thoát
S	Chốt vào chế độ tích luỹ
s	Thay đổi độ trễ giữa các cập nhật

Khi chạy lệnh *top* ta nên xoá toàn bộ màn hình. Đôi khi kết quả lệnh *top* không được hiển thị một cách đúng đắn. Vấn đề này thường xảy ra khi ta *telnet* qua mạng.

IV. SAO LƯU VÀ KHÔI PHỤC

Khi chạy một hệ thống nhiều người dùng, có truy nhập mạng, email,... sao lưu sẽ là một khía cạnh rất quan trọng hàng ngày. Chỉ đến khi đã mất các thông tin quan trọng, các cấu hình, nhiều người mới nhận ra được vai trò của sao lưu.

1. Tại sao phải sao lưu

Sao lưu là thực hiện sao một hệ thống tệp hoặc một phần của một hệ thống tệp vào một thiết bị lưu trữ khác để sau đó có thể dùng cho việc khôi phục hệ thống cũ. Hầu hết các hệ thống UNIX dùng băng từ làm thiết bị sao lưu, ta cũng có thể dùng các đĩa mềm hoặc đĩa cứng thứ cấp có thể tháo ra được.

Có rất nhiều khả năng dẫn đến hỏng dữ liệu trên đĩa trong các hệ thống máy tín hiện đại, hỏng do lỗi phần cứng, do mất nguồn điện, do các lệnh gõ sai... Đối với Linux nguyên nhân hỏng là do bản thân hệ điều hành. Vì Linux là một hệ thống đa người dùng và là hệ điều hành đa nhiệm, do đó bất kì lúc nào cũng có nhiều tệp hệ thống được mở. Linux giữ rất nhiều thông tin trong bộ nhớ về trạng thái hiện tại của nó và của các hệ thống tệp. Các thông tin này phải được ghi lên đĩa thường xuyên. Khi các tiến trình của CPU bị ngắt các tệp và các bảng hệ thống trong bộ nhớ có thể bị mất, do đó các tệp trên đĩa có thể bị đặt trong trạng thái tạm, điều đó không phù hợp với trạng thái hệ thống tệp thực.

Với các nguy cơ hỏng hệ thống tệp như vậy, hầu như ta không thể điều khiển được chúng. Giải pháp sao lưu đôi khi là duy nhất để lấy lại các thông tin đã mất. Điều này có thể làm được dễ dàng nhờ các tiện ích như *cron*, vì các tiện ích này cho phép chúng ta thực hiện công việc một cách tự động định kì.

Một vấn đề phải quan tâm khi sao lưu là thiết bị dùng để sao lưu. Thông thường đó là đĩa mềm, băng từ, đĩa cứng... Cần đảm bảo rằng chúng được giữ xa các vùng có từ trường như dây điện thoại, modem, tivi, ... Thêm nữa là nên giữ chúng ở xa các máy chính, kiểu giữ các bản sao ở xa như thế này giúp cho ta có thể sử dụng chúng cho khôi phục trong các trường hợp xảy ra tai nạn như cháy, có thể làm hỏng cả hệ thống và thiết bị sao lưu.

2. Chọn thiết bị để sao lưu

Thông thường người ta hay dùng băng từ để sao lưu bởi các đặc tính giá thành băng thấp, các lưu trữ cũng khá đơn giản và tốc độ chấp nhận được. Quá

trình đọc và ghi dữ liệu vào băng là đáng tin cậy, băng lại có thể chuyển từ máy này sang máy khác (tất nhiên là phải có ổ băng từ).

Cũng có thể dùng các thiết bị khác như ổ cứng có thể tháo lắp được dưới các dạng khác nhau như Iomega Bernoulli hay ổ ZIP. Chúng đều dùng công nghệ đầu đọc từ như các ổ cứng bình thường. Nói chung, giá của loại thiết bị này có thể cạnh tranh được với ổ băng từ.

Một số hệ thống băng từ quang học cho DOS và Windows cũng sử được dưới Linux. Chúng có hướng tới các băng có kích thước nhỏ khoảng 3.5 inch vừa với một ổ đĩa nhỏ. Chúng có ưu điểm nổi bật là không nhạy cảm đối với từ trường và có thể giữa được lâu. Loại băng nhỏ khoảng 230M có thể rẻ hơn cả một số ổ băng từ, tuy vậy loại lớn 2.4G có giá xấp xỉ giá máy tính.

Đương nhiên ta cũng có thể dùng ổ cứng khác để sao lưu.

Một loại thiết bị khác là ổ CD-ROM ghi được và WORM (ghi một lần đọc nhiều lần) cũng có thể dùng được. Tuy vậy, chúng chỉ có thể ghi được một lần, do đó lí tưởng để lưu các thông tin vĩnh viễn.

Ổ đĩa mềm được xếp vào hàng cuối cùng trong các thiết bị sao lưu cho các hệ thống tệp lớn, mặc dù nó tốt cho các hệ thống tệp nhỏ.

3. Đặt kế hoạch sao lưu

Một vấn đề quan trọng của sao lưu là sao lưu thường xuyên. Điều này đặc biệt quan trọng đối với các hệ thống đa người dùng và thường xuyên cần thay đổi hệ thống tệp. Ta cũng không cần phải sao lưu mọi thứ mà chỉ cần sao lưu những tệp đã thay đổi kể từ lần sao lưu trước.

Các nhà quản trị hệ thống UNIX thích thực hiện sao lưu vào ban đêm vì lúc đó có ít người dùng đăng nhập, CPU không phải làm việc nặng nhọc, và hệ thống có ít tệp đang mở nhất. Nói chung, việc sao lưu ít khi được thực hiện bằng tay (trừ phi hệ thống không được dùng thường xuyên) mà được làm tự động nhờ chương trình *cron*. Do đó, có thể đặt thời gian thực hiện chính xác để làm ảnh hưởng ít nhất đến các tiến trình nền mà hệ thống đang chạy.

Không bao giờ chỉ giữ đúng một bản sao của hệ thống vì như vậy ta sẽ không thể quay lui đến bản sao trước được. Ví dụ ta thực hiện sao lưu hàng ngày nhưng lại muốn khôi phục lại một tệp đã xoá từ 1 tuần trước, khi đó điều này trở nên không thể được.

Lí tưởng nhất là chúng ta giữ các bản sao nhiều ngày, thậm trí hàng tuần trước khi dùng lại chúng. Trong các hệ thống nhiều người dùng điều này đặc biệt quan trọng vì đôi khi họ chỉ nhớ ra là cần một tệp sau khi đã xoá nó 2 tháng, khi đó ta đã quay vòng sử dụng băng vài lần. Kế hoạch lí tưởng phụ thuộc vào quan điểm của người quản trị đối với việc sao lưu, nhưng một hệ thống sao lưu toàn diện yêu cầu ít nhất các bản sao lưu tăng tiến hàng ngày (chỉ sao các phần thay đổi) của 2 tuần và một bản sao lưu toàn bộ mỗi tuần.

Sao lưu toàn bộ đòi hỏi phải sao tất cả các tệp của hệ thống do đó thiết bị lưu phải có kích thước bằng tổng kích thước hệ thống tệp. Có thể ta phải dùng nhiều thiết bị lưu cùng loại cho một lần sao lưu nếu dung tích của thiết bị nhỏ hơn hệ thống tệp cần lưu. Một số hệ thống sao lưu dùng các thuật toán nén do đó kích thước cần lưu thực sự sẽ nhỏ đi.

Sao lưu tăng tiến (còn gọi là sao lưu chênh lệch) chỉ lưu các tệp đã bị thay đổi hoặc được tạo sau lần sao lưu gần nhất. Để xác định các tệp này ta có thể căn cứ vào ngày sửa tệp.

Sao lưu thăng dư đôi khi khá khó khăn đối với Linux do đó người ta dùng một cách sao lưu khác là chỉ sao lưu tăng tiến một vùng của hệ thống tệp mà có vẻ như đã bị thay đổi. Sao lưu kiểu này gọi là sao lưu bộ phận. Ví dụ sao lưu hệ thống tệp /usr nếu mọi tệp của người dùng đều nằm ở đây.

Nên gán nhãn cho các băng sao lưu với các tên phản ánh cách ta dùng chúng. Ví dụ Ngày1, Ngày2 ... Ngày10. Sau đó khi phải quay vòng ta lại dùng lại Ngày1 (do đó Ngày1 luôn sau Ngày10).

Việc sao lưu bao nhiêu lâu một lần là tuỳ thuộc vào từng hệ thống. Nói chung với các hệ thống đa người dùng, hay thay đổi thì nên thực hiện hàng ngày. Nên thực hiện một sao lưu toàn bộ sau bốn hay năm lần sao lưu bộ phận.

Một kế hoạch sao lưu mở rộng của sao lưu hàng ngày được nhiều người quản trị thích dùng là chu kì sao lưu hàng ngày và hàng tuần. Ta chia hệ thống băng dùng để sao lưu thành 2 nhóm cho sử dụng hàng ngày và hàng tuần. Ví dụ, có 14 băng, ta dùng 10 băng cho sao lưu hàng ngày trong 2 tuần (trừ đi 4 ngày nghỉ cuối tuần của 2 tuần). Gán cho chúng các nhãn từ Ngày1 đến Ngày 10. Dùng 4 băng còn lại cho chu kì 2 tuần một và gọi chúng là Tuần 1,... Tuần 4. Như vậy hàng ngày sẽ lần lượt dùng các băng Ngày, đến hết một vòng sử dụng các băng ngày thì dùng một băng Tuần. Sau đó lại quay vòng dùng các băng Ngày rồi đến băng Tuần...

Chu kì sao lưu này có những ưu điểm hơn hẳn chu kì sao lưu hàng ngày đơn giản. Với 10 băng sao Ngày ta có thể khôi phục lại thông tin trong vòng 2 tuần trước. Với 4 băng sao lưu hai tuần một (băng Tuần) ta có thể khôi phục một tệp từ cách đó 8 tuần tức là một tệp hoặc một nhóm tệp từ 2 tháng trước chứ không phải chỉ hai tuần. Với nhiều băng hơn, ta có thể mở rộng chu kì sao lưu hoặc thêm chu kì sao lưu hàng tháng.

4. Giữ các thông tin về sao lưu

Sau khi đã sao lưu dữ liệu vài lần, đến lúc cần khôi phục một tệp, lúc đó chúng ta không biết rằng tệp đó nằm trong băng lưu nào hay băng đó lưu lại ngày nào. Để giải quyết vấn đề này, một số người ghi ngày, nội dung của băng lên một mảnh giấy gắn vào băng. Cách này không hay lắm vì ta phải lục tất cả các băng xem tệp ta cần ở đâu. Do đó chúng ta cần giữ các thông tin về sao lưu.

Mỗi khi sao lưu, ta lại cập nhật thông tin sao lưu. Đó là một tờ giấy hoặc một quyển sổ có các cột chứa các thông tin cơ bản sau :

- Ngày sao lưu
- Tên băng sao lưu (ví dụ Ngày 1)
- Hệ thống tệp được sao lưu
- Sao lưu toàn bộ hay bộ phận, nếu sao lưu bộ phận thì sao lưu thư mục nào.

Các thông tin này chỉ cần ghi lại trong vài giây mà thôi. Với các hệ thống lớn, ta có thể thêm một số thông tin khác cho đầy đủ hơn.

- Ai sao lưu
- Sao lưu bằng tay hay tự động
- Nơi cất băng lưu.

Ngày sao lưu giúp ta theo dõi thời điểm sao lưu lần cuối, đồng thời cũng là chỉ số để khôi phục tệp. Nếu một người dùng của hệ thống muốn khôi phục một tệp mà anh ta biết đã xoá nhầm 2 tuần trước, ta có thể căn cứ vào ngày để tìm ra băng chứa tệp đó nhờ vào ngày của thông tin sao lưu.

Để thuận tiện nên giữ thông tin sao lưu ở gần hệ thống, có người thích để nó ở cùng với các băng sao lưu.

Như đã nói, để có thể sao lưu tự động định kì, chúng ta dùng chương trình *cron*. Bằng cách thêm một dòng đặc tả thời các điểm sao lưu, chương trình thực hiện sao lưu vào trong file *crontab* là chương trình *cron* sẽ thực hiện việc sao lưu tự động cho chúng ta vào đúng thời điểm mà ta muốn. (Về chương trình *cron* chúng ta có thể xem chi tiết hơn ở phần sau). Như vậy vấn đề còn lại chỉ là chương trình thực hiện sao lưu, sau đây chung ta sẽ xem xét một vài chương trình sao lưu.

5. Sao lưu bằng lệnh tar

Chương trình *tar* (tape archipver : lưu trữ băng) là lệnh lưu các tệp và thư mục vào một thiết bị lưu trữ và khôi phục chúng sau đó. Lệnh *tar* tạo một tệp lưu trữ, là tệp duy nhất chứa nhiều tệp trong nó (giống PKZIP làm trong DOS). Lệnh *tar* chỉ làm việc với các tệp mà nó tạo ra.

Khuôn dạng của lệnh tar :

```
$ tar switch modifiers files
```

Phần `files` của lệnh chỉ các tệp hay thư mục người sử dụng muốn lưu hay khôi phục. Người sử dụng có thể lưu toàn bộ hệ thống tệp */usr* và khi khôi phục người sử dụng có thể chỉ cần khôi phục một tệp, ví dụ */usr/tienmc/big file*

Tham số `switch` chỉ cách thức mà lệnh *tar* ghi và đọc từ thiết bị lưu. Người sử dụng chỉ có thể dùng một tuỳ chọn `switch` với tar tại một thời điểm. Các giá trị đó có thể là:

Bảng 2.15. Các tuỳ chọn của tar

c	Tạo một tệp lưu mới
r	Ghi vào cuối tệp lưu hiện tại
t	Liệt kê danh sách các tệp trong tệp lưu
u	Thêm tệp chưa bị sửa hoặc chưa được lưu
x	Trích ra từ tệp lưu

Cũng có thể thêm một số các `modifiers` để điều khiển tệp lưu và cách *tar* dùng nó. Sau đây là các giá trị của `modifiers` :

Bảng 2.16. Các giá trị của modifier

A	Bỏ tên tệp tuyệt đối (gồm cả đường dẫn)
b	Đưa ra tham số khối (0-20)
c	Chống cắt tệp để ghi trên các thiết bị lưu khác nhau
f	Chỉ ra tên thiết bị lưu trữ
F	Specifies the name of a tệp for tar arguments
k	Đưa ra kích thước của thiết bị lưu (kilobytes)
l	Đưa ra thông báo lỗi nếu không thể liên kết được
m	Không khôi phục thời điểm sửa
u	Chỉ ra thiết bị lưu không phải là băng
p	Trích ra các tệp với các quyền của nó như ban đầu
v	Liệt kê danh sách tệp
w	Hiển thị hành động lưu và chờ người dùng khẳng định lại.

Lệnh *tar* dùng đường dẫn tuyệt đối cho hầu hết các hoạt động trừ khi có `modifier` là A.

Mặc định tệp dùng để lưu trữ kà đầu ra chuẩn (tức là màn hình) vì vậy nếu không chỉ rõ tệp ra thì kết quả của việc lưu trữ sẽ được in ra màn hình. Để chỉ rõ sẽ lưu vào tệp nào ta sử dụng cách định hướng dòng ra bằng kí hiệu >. Ví dụ để lưu 2 tệp `anh1.jpg` và `anh2.jpg` vào tệp mới `anh.bk` ta dùng :

```
$tar -c anh1.jpg anh2.jpg > anh.bk
```

Sau đây là một số ví dụ.

Nếu dùng một ổ băng là */dev/tape* và lưu toàn bộ hệ thống tệp, kích thước hệ thống tệp nhỏ hơn dung lượng băng. Ta dùng lệnh :

```
$ tar cf /dev/tape /
```

Tuỳ chọn f cho phép ta chỉ ra tên thiết bị */dev/tape*. Toàn bộ hệ thống tệp được lưu trong tệp lưu mới tạo (chỉ ra bởi tuỳ chọn c). Mọi thứ đang có trong băng sẽ bị ghi đè khi tệp lưu mới được tạo. Nếu thêm tuỳ chọn v trong lệnh, *tar* sẽ hiển thị tên tệp và các kích thước của chúng lên màn hình sau khi đã lưu.

Để khôi phục toàn bộ hệ thống tệp từ một băng đã dùng trong ví dụ trước, dùng lệnh :

```
$ tar xf /dev/tape
```

Để khôi phục một tệp trong băng có thể chỉ rõ tệp đó :

```
$ tar xf /dev/tape /usr/tiennc/bigfile
```

Lệnh này chỉ khôi phục tệp */usr/tiennc/bigfile*.

Để lấy danh sách tất cả các tệp trong băng lưu, ta dùng lệnh :

```
$ tar tvf /dev/tape
```

Tuỳ chọn v cho phép hiển thị danh sách lên màn hình.

Hầu hết các băng đều đòi hỏi phải đưa ra hệ số khối khi tạo một tệp lưu, nhưng ta không cần phải chỉ ra số này khi đọc băng vì *tar* có thể tự nhận ra. Hệ số khối báo cho *tar* biết lượng dữ liệu viết vào một khúc trong băng. Khi lưu vào băng, người sử dụng chỉ ra hệ số này với `modifier b`. Ví dụ :

```
$ tar cvfb /dev/tape 20 /usr
```

Lệnh này tạo một tệp lưu mới trên */dev/tape* có hệ số khối 20 và chứa tất cả các tệp của */usr* (nói chung mọi băng đều hoạt động với hệ số khối 20). Chú ý rằng các tham số `modifier` phải đúng thứ tự với kiểu `modifier` đã chỉ ra, như ở đây `f` trước b do đó tên thiết bị phải được chỉ ra trước hệ số khối.

Một vấn đề khác là băng có thể không đủ chứa toàn bộ hệ thống tệp, do đó phải dùng nhiều băng. Tuỳ chọn k được sử dụng để báo cho *tar* biết kích thước của băng, nó sẽ tương ứng với tham số là dung lượng của băng theo kilobyte. Ví dụ :

```
$ tar cvbfk 20 /dev/tape 122880 /usr
```

Như vậy tar dùng hệ số khối là 20 cho thiết bị lưu */dev/tape*, dung lượng băng là 122880KB.

Nếu dùng đĩa mềm để lưu, ta không dùng tham số khối :

```
$ tar cnfk /dev/fd0 1200 /usr/tiennc
```

trong đó thiết bị lưu là */dev/fd0*, dung lượng 1200KB. Tham số n báo cho *tar* biết đây không phải là băng.

6. Sao lưu tăng tiến dùng dump

Chương trình *dump* và *restore* rất hay được dùng để thực hiện sao lưu. Tuy vậy một vài phiên bản ATT của UNIX cũng không có các lệnh này. Lệnh *dump* cho phép người quản trị hệ thống lưu toàn bộ hệ thống tệp hoặc bất kì phần nào của hệ thống tệp.

6.1. Lệnh dump

Lệnh *dump* xây dựng một *danh* sách tương tự như một cây thư mục các tệp đã bị thay đổi kể từ lần thực hiện lệnh *dump* trước và đóng gói những tệp này vào một tệp lớn rồi lưu vào một thiết bị lưu trữ ngoài.

Để có thể thực hiện sao lưu chỉ những tệp bị thay đổi từ lần thực hiện *dump* trước (và do đó làm cho việc sao lưu trở nên hiệu quả hơn), *dump* sử dụng số mức, đó là một con số từ 0 đến 9 để xác định các tệp nào cần được sao lưu. Thực hiện *dump* ở mức 0 có nghĩa là thực hiện *dump* toàn bộ hệ thống tệp . *dump* ở mức *n* có nghĩa là thực hiện lưu các file đã bị thay đổi từ lần *dump* gần nhất với mức *dump* nhỏ hơn hoặc bằng n. Các thời điểm thực hiện *dump* trước được lưu trong */etc/dumpdates*.

Hầu hết các phiên bản của *dump* không cập nhật */etc/dumpdate* trừ khi được yêu cầu. Trong khi đó, nếu không có cập nhật này thì không có thể thực hiện sao lưu tăng tiến bằng *dump* một cách đúng đắn được vì không thể xác định được thời điểm lần cuối cùng thực hiện *dump* mức n. Cờ u sẽ khiến lệnh *dump* thực hiện cập nhật */etc/dumpdates*. Tệp */etc/dumpdates* là một tệp văn bản nên có thể sử đổi được. Nếu tên của một hệ thống tệp bị thay đổi, */etc/dumpdates* phải được sử đổi để phản ánh sự thay đổi này nhằm tránh *dump* bị nhầm lẫn

dump sẽ gửi đầu ra ra một số thiết bị lưu trữ mặc định thông thường là ổ băng từ đầu tiên trong hệ thống lớn và ổ đĩa mềm đầu tiên trong hệ thống nhỏ. Nếu muốn *dump* sao lưu ra một thiết bị đầu ra khác thì phải dùng cờ `f`.

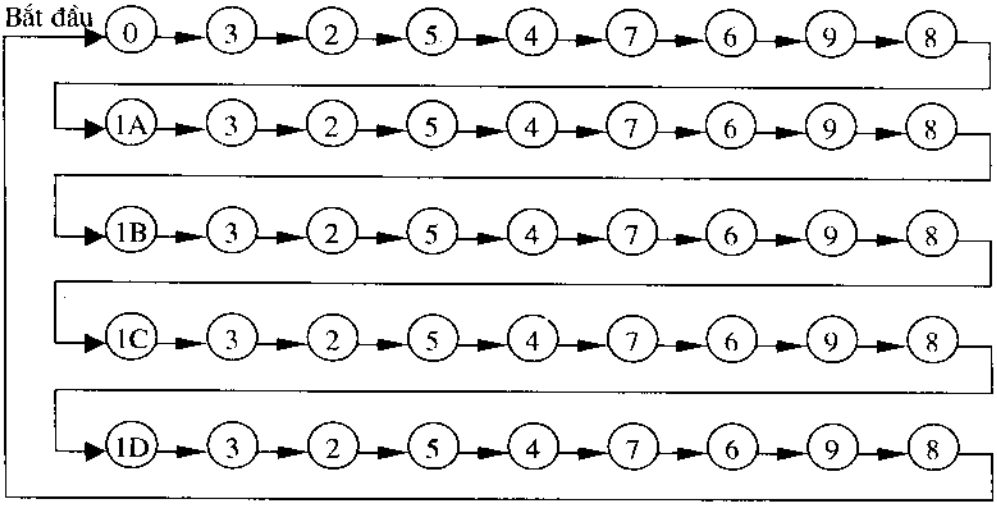
Ngoài ra khi dùng *dump* chúng ta có thể chỉ ra một số tuỳ chọn khác như kiểu của băng từ được sử dụng để lưu trữ, độ dài của băng, mật độ của băng. Ví dụ thực hiện sao lưu ở mức 5 cho hệ thống tệp được gắn kết vào thư mục */work* vào một băng dài 1700 foot, mật độ 1000bpi băng này tương ứng với tệp */dev/rst0*, ta thực hiện lệnh :

```
dump 5usdf 1700 1000 /dev/rst0/work
```

6.3. Chuỗi dump Tháp Hà Nội

Ta có thể thực hiện *dump* theo nhiều thứ tự khác nhau, một trong số đó thực hiện dựa trên bài toán Tháp Hà Nội. Thứ tự này là kết quả của sự thoả hiệp giữa mong muốn lưu được càng nhiều thông tin càng tốt và càng lâu càng tốt và hạn chế

về thời gian của người quản trị, CPU để thực hiện sao lưu và số lượng băng từ có hạn. Chuỗi thực hiện *dump* theo mô hình Tháp Hà Nội như sau :



Hình 2.1. Chuỗi dump tháp Hà Nội

Khi chuỗi này kết thúc nó lại lặp lại, tuy vậy với mỗi mức 0 lại sử dụng một băng mới. Cứ sau 9 lần thực hiện *dump* các băng (hoặc các bộ băng nếu như số băng cho mỗi lần thực nhiện *dump* là lớn hơn 1) cho mức 2 và 9 được sử dụng lại, trong khi đó các băng mức 1 (A,B,C,D) chỉ được sử dụng lại sau 45 lần thực hiện *dump*. Có 4 bộ băng mức 1 cho mỗi hệ thống tệp, được dán nhãn 1A, 1B, 1C, 1D. Mức 1 được thực hiện trên mỗi bộ băng này và chỉ số A, B, C, D chỉ để phân biệt giữa các bộ băng. Mỗi hệ thống tệp cần có một bộ băng cho các mức từ 1 đến 9.

6.3. Khôi phục từ các sao lưu bằng dump

Có nhiều biến thể của chương trình khôi phục sao lưu từ *dump*, một số phiên bản gọi là *restore* một số khác lại gọi là *restor*.

Để khôi phục một tệp, việc đầu tiên là phải gắn kết băng có chứa tệp đó vào hệ thống rồi chuyển đến thư mục mà ta muốn khôi phục tệp vào đó. Sau đó là thực hiện lệnh *restore -x* để khôi phục tệp. Ví dụ để khôi phục tệp */usr/chinq/lostfile* vào thư mục /tmp:

[gắn kết băng]

```
$ cd /tmp
$restore x /usr/chinq/lostfile
$ls /tmp/usr/chinq
lostfile
$ ls /usr/chinq
afile   bfile  cfile
$cp /tmp/usr/chinq/lostfile /usr/chinq/lostfile
$chown chinq /usr/chinq/lostfile
$chgroup svtt /usr/chinq/lostfile
```

Để khôi phục toàn bộ hệ thống tệp, việc đầu tiên là phải gắn kết một hệ thống tệp mới làm đích cho các tệp được khôi phục, chuyển đến thư mục gốc của hệ thống tệp đó. Tiếp theo là gắn kết băng lưu gần nhất có mức 0 (tức là mức sao lưu toàn bộ) và dùng lệnh *restore -r*, lệnh *restore* sẽ nhắc ta việc đưa vào các băng. Sau khi thực hiện xong việc khôi phục từ băng mức 0, ta cần phải tiếp tục thực hiện khôi phục với các băng có mức dump cao hơn cho đến mức *dump* cuối cùng thực hiện được. Ví dụ, lần *dump* gần nhất có mức 5 theo sơ đồ *dump* Tháp Hà nội, khôi phục lại hệ thống tệp */usr* nằm trên */dev/rhp0g* :

```
$ /etc/newfs /dev/rhp0g eagle  (newfs là phiên bản thân
thiện hơn của mkfs)
$ /etc/mount /dev/hp0g /usr
$ cd /usr
[ gắn kết băng mức 0 đầu tiên vào /usr]
$ restore -r
[ gắn kết các băng tiếp theo theo yêu cầu của lệnh
restore]
[ gắn kết băng đầu tiên của băng dump mức 1 gần
nhất của /usr nếu băng này mới hơn băng mức 0]
$ restore -r
[ gắn kết các băng tiếp theo theo yêu cầu của lệnh
restore]
[ gắn kết băng mức 3 của /usr]
```

```
$ restore -r
[gắn kết băng mức 2 của /usr]
$ restore -r
[gắn kết băng mức 5 của /usr]
$ restore -r
```

6.4. Khôi phục có tương tác với người sử dụng

Để thực hiện khôi phục có tương tác với người sử dụng, ta dùng lệnh *restore* với cờ `-i`. Tuỳ chọn này hiển thị các tệp trên băng và cho phép duyệt qua hệ thống tệp như các cây thư mục bình thường bằng các lệnh *ls*, *cd*, *pwd*. Các tệp muốn khôi phục được đưa vào một danh sách bằng lệnh *add*, và khi đã chọn xong các tệp muốn khôi phục, lệnh *extract* sẽ cho phép trích các tệp này ra từ băng lưu. Sử dụng lệnh *restore* với tuỳ chọn `-i` rất tiện vì thế hãy sử dụng chúng khi cần khôi phục các tệp.

7. Lệnh cpio

Lệnh này hoạt động cũng giống như *tar*. Tất cả các phiên bản ATT của UNIX đều có trong khi một số bản lại không có lệnh *tar* vì thế dùng *cpio* là giải pháp tốt để sao lưu phục vụ cho việc chuyển giữa các hệ thống dùng phiên bản ATT. *cpio* có thể dùng để chuyển các cây thư mục, ví dụ, lệnh :

```
find adir -depth -print | cpio -pd /tmp
```

sẽ chép cây thư mục `adir` vào */tmp*. Giống như *tar*, *cpio* không cho phép dùng nhiều băng lưu. Một vài phiên bản của *cpio* còn không thực hiện tốt với cách nối lệnh theo kiểu ống nước (dùng các gạch đứng như trên), và chỉ có root mới có quyền sao các tệp đặc biệt.

2. Lệnh dd

dd là chương trình sao chép và chuyển đổi. Trừ phi yêu cầu *dd* thực hiện chuyển đổi, *dd* chỉ thực hiện sao chép từ tệp đầu vào ra tệp đầu ra. *dd* thường được dùng để đọc các băng không được ghi dưới khuôn dạng UNIX, ngoài ra *dd* được dùng để tạo bản sao của toàn bộ hệ thống tệp một cách dễ dàng. *dd* cũng được

dùng để tạo bản sao các băng từ một cách nhanh chóng. Ví dụ :

Nếu bạn có 2 ổ băng từ (*/dev/rmt8* và */dev/rmt9*) và thực hiện sao từ ổ này sang ổ kia : :

```
%dd if=/dev/rmt8 of=/dev/rmt9 cbs=16b
```

Nếu bạn chỉ có một ổ băng từ (*/dev/rmt8*) thì việc tạo bản sao của băng phải thực hiện gồm 2 công đoạn, thứ nhất là sao nội dung của băng vào đĩa cứng, sau đó sao nội dung đã ghi trên đĩa cứng ra băng. Tất nhiên là khi đó phải đảm bảo rằng bạn có đủ không gian đĩa để lưu tạm nội dung của băng.

```
%dd if =/dev/rmt8 of=tfile cbs=16b
```

[đổi băng]

```
%dd if=tfile of=/dev/rmt8 cbs=16b
%rm file
```

V. THỰC HIỆN TỰ ĐỘNG CHƯƠNG TRÌNH

Chương trình *cron* và *at* là các chương trình cho phép người sử dụng thực hiện tự động các chương trình cần thiết tại một thời điểm xác định.

1. Chương trình cron

Chương trình *cron* (viết tắt của chronograph là) được thiết kế để cho phép thực hiện các lệnh tại một thời điểm xác định mà không cần phải trực tiếp kích hoạt chúng. Linux chạy *cron* dưới dạng một daemon đồng hồ khi bật hệ thống. Thông thường, lệnh chạy *cron* được chỉ ra trong tệp rc. Khi hoạt động *cron* đọc ngày và thời gian được đặt để thực hiện một lệnh trong tệp *crontab*. Mỗi dòng trong tệp này chỉ ra khi nào và làm thế nào để thực hiện lệnh tương ứng. Khi nhận ra ngày, giờ của hệ thống trùng với ngày giờ của một lệnh trong tệp *crontab*, daemon *cron* thực hiện lệnh đó. Lệnh đó không chỉ được thực hiện một lần mà cứ mỗi lần có sự phù hợp về ngày giờ lệnh đó lại được thực hiện. Chính nhờ tính chất thực hiện tự động này mà cron trở nên lí tưởng đối với người quản trị hệ thống để thực hiện các công việc như sao lưu hệ thống, tổ chức lại cơ sở dữ liệu...

Trong hầu hết các hệ thống, truy nhập *cron* chỉ giới hạn đối với người quản

trị hệ thống, tuy vậy cũng có thể trao quyền sử dụng nó cho một vài hoặc tất cả người dùng của hệ thống. Người quản trị hệ thống điều khiển việc ai có thể gửi các tiến trình sẽ được cron thực hiện thông qua chỉ một trong hai tệp là */usr/lib/cron/cron.allow* hay */usr/lib/cron/cron.deny* (nhiều hệ thống Linux dùng các tên */etc/cron.d/cron.allow* và */etc/cron.d/cron.deny*). Trong các tệp này liệt kê các tên người dùng, mỗi tên một dòng.

Tệp */usr/lib/cron/cron.allow* (hay */etc/cron.d/cron.allow*) chứa danh sách tất cả tên người dùng được phép dùng *cron* , ví dụ:

```
tiennc
chinq
nvviet
```

chỉ cho các đăng nhập `tiennc`, `chinq`, và `nvviet` (tất nhiên là cả siêu người dùng) có thể chuyển bất cứ công việc nào cho cron.

Tệp */usr/lib/cron/cron.deny* chứa danh sách tên người dùng không được phép dùng *cron*. Ví dụ :

```
sonnt
andt
```

cho phép mọi người trừ `sonnt` và `andt` dùng *cron*.

Nếu không có tệp nào trong hai tệp trên tồn tại thì chỉ có root mới có thể chuyển các tiến trình cho *cron*. Để cho phép mọi người dùng dùng *cron*, hãy tạo ra một tệp *cron.deny* rỗng.

1.1. Tệp crontab

Tiện ích *crontab* được sử dụng để ra lệnh cho *cron* thực hiện một lệnh tại một thời điểm nhất định. Chương trình *crontab* đọc một tệp trong đó ghi rõ những gì người dùng muốn *cron* thực hiện theo khuôn dạng *crontab*. Chương trình *crontab* còn thực hiện một số nhiệm vụ quản trị khác như hiển thị danh sách các công việc hiện tại của *cron* , xoá danh sách, thêm công việc mới vào danh sách.

Tệp chứa dữ liệu được chương trình *crontab* đọc để xác định những công việc mà *cron* phải thực hiện thường có tên là *crontab*. Tiện ích *crontab* có một tuỳ chọn cho phép người sử dụng xác định tên tệp chứa dữ liệu này nhưng tên tệp mặc định sẽ là *crontab*.

Cấu trúc tệp *crontab* là các dòng, mỗi dòng gồm các thông tin về thời điểm và công việc phải thực hiện tại thời điểm đó.

```
phut gio ngay-trong-thang thang-trong-nam ngay-trong-tuan lenh
```

Các trường trong tệp ngăn cách bởi dấu cách và có ý nghĩa sau :

- Phút trong giờ (0-59)
- Giờ trong ngày (0-23)
- Ngày trong tháng (1-31)
- Tháng (1-12)
- Ngày trong tuần (Sun=0, Mon=1, ... Sat=6)

Chương trình sẽ được thực hiện tại thời điểm đã chỉ ra. Có thể một lệnh hoặc là đường dẫn đầy đủ chỉ đến một chương trình script (kể cả khi đường dẫn đã có trong PATH vì *cron* không thừa kế các biến môi trường do đó nó không biết tìm chương trình cần thực hiện ở đâu). Tất nhiên, người dùng phải có quyền thực thi lệnh hay script này.

Các cột thời gian và ngày trong tệp *crontab* có thể chứa một số duy nhất hoặc một khoảng (được biểu diễn bằng một dấu "-" giữa hai số chỉ giá trị đầu và cuối, ví dụ 1-5 chỉ một khoảng từ 1 đến 5), hoặc một danh sách các gía trị phân cách bởi dấu phảy, hoặc dấu "*" để chỉ giá trị bất kì.

Ví dụ, một vài dòng trong tệp *crontab*

```
20 1 * * * /usr/bin/calendar -
0 2 1 * 0 /bin/organize_data
10,30,50 9-18 * * * /bin/setperms
```

Ví dụ này chỉ ra 3 tiến trình khác nhau. Dòng đầu chỉ ra lệnh `/usr/bin/calendar -` (dấu trừ là một phần của lệnh), phải được thực hiện vào 1h20 sáng hàng ngày trong tuần và hàng ngày trong năm.

Dòng thứ hai chỉ, vào 2h00 vào ngày đầu tiên của mọi tháng và mọi ngày chủ nhật tệp script */bin/organize_data* phải được thực hiện. Tất nhiên nếu hai ngày này trùng nhau thì script cũng chỉ được thực hiện một lần.

Thứ tự các dòng trong tệp *crontab* không quan trọng. Nếu có lỗi trong tệp crontab, chương trình *cron* sẽ gửi cho người dùng một thông điệp chỉ ra lỗi mà nó gặp phải khi thực hiện lệnh.

1.2. Chuyển và quản lí các tệp crontab

Sau khi đã tạo tệp *crontab* của riêng mình, người dùng có thể đưa nó cho *cron* thực hiện. Khi chuyển nó cho *cron*, tệp này được sao vào thư mục *cron*, thường là trong thư mục */usr/spool/cron/crontabs*, tệp sẽ có tên của người dùng đã chuyển nó cho *cron*. Ví dụ người dùng có tên chinq thì tệp sẽ để trong thư mục */usr/spool/cron/crontabs/chinq*, mọi tệp *crontab* của siêu người dùng thường có tên root.

Để chuyển tệp *crontab* cho *cron*, người sử dụng dùng lệnh *crontab* với cú pháp như sau:

```
$ crontab tep_crontab
```

với *tệp_crontab* là tên tệp mà người sử dụng muốn *crontab* sẽ đọc. Nếu không chỉ ra tên *tệp_crontab* thì mặc định tên tệp sẽ là *cron_tab*. Nếu người sử dụng đã chuyển một tệp *crontab* cho *cron* rồi thì tệp cũ sẽ bị thay thế bằng tệp mới.

Bất cứ thay đổi nào đối với *cron* đều phải thực hiện thông qua việc chuyển tệp *crontab* mới cho *cron*. Người dùng không bao giờ được sửa trực tiếp tệp trong thư mục */usr/spool/cron/crontabs*, *cron* sẽ không thể đọc được các thay đổi này.

Với tuỳ chọn -l của lệnh *crontab* ta có thể thấy tất cả các nhiệm vụ mà người đưa ra lệnh này đã chuyển cho *cron*, cũng có nghĩa là cho thấy nội dung tệp *crontab* của người đó.

```
$ crontab -l
```

Để xoá tệp *crontab* của mình, người sử dụng dùng tuỳ chọn -r (remove). Tuỳ chọn này xóa tệp với tên của người dùng trong thư mục */usr/spool/cron/crontabs*. Cú pháp của lệnh này là:

```
$ crontab -r
```

Để sửa tệp *crontab* hiện tại của mình người sử dụng có thể dùng lựa chọn -e (editor), nó sẽ kích hoạt luôn trình soạn thảo mặc định của hệ thống.

```
$ crontab -e
```

Sau khi đã được sử và ghi lại, tệp *crontab* của ta lại được chuyển cho *cron*.

Cron đọc nội dung thư mục */usr/spool/cron/crontab* ít nhất 5 phút một lần

(với hầu hết hệ thống Linux là đọc mỗi phút một lần). Vì vậy mọi thay đổi trong tệp *crontab* có thể có chỉ có tác dụng sau vài phút, nếu hệ thống càng phải hoạt động nặng thì thời gian trễ càng lớn.

1.3. Dùng các lệnh cron phức tạp

Các lệnh trong tệp *crontab* có thể là các lệnh bất kì hay các script, miễn là dòng lệnh phải đúng, tức là chúng có thể thực hiện được từ dấu nhắc của shell.

Đôi khi các lệnh hay các script trong tệp *crontab* có thể sinh ra các thông tin đầu ra hoặc các thông báo lỗi. Để tránh các thông báo lỗi này, ta có thể định hướng đầu ra vào */dev/null*. Ví dụ

```
0 * * * * date > /tmp/test1 2>/dev/null
```

chuyển đầu ra vào tệp */tmp/test1* và thông báo lỗi vào */dev/null* (tức là bỏ các thông báo lỗi).

Hay

```
30    1    *    *    *    cat    /usr/tiennc/chapt*    >
/usr/tiennc/archive/backup
```

Ghép các tệp bắt đầu bằng chapt trong */usr/tiennc* vào một tệp lớn là */usr/tiennc/archive/backup*.

Người dùng cũng có thể dùng cơ chế nối tiếp (pipeline) các lệnh trong tệp *crontab*. Ví dụ, ta có danh sách người dùng đã đăng nhập hệ thống trong ngày trong tệp */tmp/userlist*, sắp xếp chúng và gửi cho root.

```
0 1 * * * sort -u /tmp/userlist | mail -s"users for today" root
```

Một điểm quan trọng của cron là tất cả các lệnh được thực hiện trên Bourne shell (hoặc bash) nhưng không thực hiện đúng với C shell.

2. Chương trình at

Chương trình *at* cũng tương tự như *cron*, trừ một điểm là nó thực hiện lệnh chỉ một lần tại thời điểm đã chỉ ra chứ không phải thực hiện mãi như *cron*. Khuôn dạng của lệnh *at* :

```
at time date < file
```

Trong đó `time` và `date` lần lượt chỉ thời gian và ngày thực hiện lệnh, tệp là tên tệp đầu vào lệnh *at*.

Có thể truyền tham số cho lệnh *at* theo nhiều cách khác nhau, đó là điểm linh hoạt của lệnh *at*. Ví dụ có thể đưa ra giờ theo kiểu 18:40 (để chỉ 18 giờ 40 phút), hay 10 (để chỉ 10 giờ sáng) hay 10pm (để chỉ 10 giờ chiều), chương trình đều hiểu đúng. Ngoài ra nó còn chấp nhận một số từ đặc biệt thay vì phải chỉ ra giờ, ví dụ `noon`, `midnight`, `now`, `next` hay `zuzu`.

Tham số `date` có thể có hoặc không, thường được dùng khi `time` không được chỉ đủ rõ. Nếu không đưa ra `date` thì lệnh sẽ được thực hiện ngay lần tới gặp thời điểm như chỉ ra trong `time`. `Date` cũng có thể được đưa ra theo nhiều cách, có thể là tháng rồi đến ngày (May 15) hay một ngày trong tuần (theo dạng đầy đủ như Monday hay viết tắt 3 kí tự như Mon), người dùng cũng có thể chỉ ra năm, nhưng rất hiếm khi dùng. Lệnh *at* cũng nhận ra được một số từ đặc biệt cho `date` như `today` hay `tomorow`.

Tệp đầu vào của lệnh *at* có thể là tệp bất kì với các lệnh trong nó.

Ví dụ ta có tệp *reorg.data* với các lệnh bên trong :

```
/usr/tiennc/setperms
/usr/tiennc/sort_database
/usr/tiennc/index_database
/usr/tiennc/clean_up
```

Ta muốn nó được thực hiện vào 8h30 sáng, có thể đưa ra một trong các lệnh sau :

```
at 20:30 < reorg.data
at 8:30 pm < reorg/data
at 20:30 today < reorg.data
```

hay

```
at 0900 Monday next week < reorg.data
```

thực hiện lệnh vào thứ hai tuần sau.

Khi chuyển một chương trình cho *at*, sẽ nhận lại được một số hiệu công việc (job ID) xác định duy nhất lệnh *at* vừa đưa ra. Ví dụ

```
$ at 6 < do_it
job 827362.a at Wed Aug 31 06:00:00 EDT 1995
```

ở đây, số hiệu công việc là 827362.a, người sử dụng có thể dùng nó để thay đổi

công việc tương ứng.

Ta có thể liệt kê tất cả các công việc đã xếp vào *at* bằng tuỳ chọn -l. Đầu ra sẽ là thời điểm công việc được thực hiện nhưng không nói là thực hiện lệnh gì.

```
$ at -l
user - tiennc job 827362.a at Wed Aug 31 06:00:00 EDT 1995
user = tiennc job 829283.a at Wed Aug 31 09:30:00 EDT 1995
```

Để bỏ một công việc của at khỏi hệ thống, ta dùng tuỳ chọn -r (remove) với số hiệu công việc. Ví dụ :

```
$ at -r 2892732.a
```

Người dùng chỉ có thể bỏ đi một công việc của chính họ đã gửi cho *at*, nhưng root thì có thể bỏ bất kì công việc nào.

Mọi công việc được xếp vào *at* đều được lưu trong thư mục */usr/spool/cron/atjobs* với số hiệu công việc là tên tệp. Cũng như *cron* một trong hai tệp *at.allow* hoặc *at.deny* trong thư mục */usr/lob/cron* hoặc */etc/cron.d* điều khiển việc ai được phép dùng *at*.

Khi một công việc của *at* được thực hiện, tất cả đầu ra (đầu ra chuẩn và các thông báo lỗi) được gửi lại cho người dùng đã gửi công việc này cho *at* (tất nhiên là không kể khi đầu ra được định hướng lại). Lệnh *at* nhớ tất cả các biến môi trường và các thiết lập thư mục của người dùng. Nhìn vào danh sách các công việc trong */usr/spool/cron/atjobs* sẽ thấy tất cả các biến được định nghĩa trước khi lệnh được thực hiện.

Chương III

QUẢN LÝ CẤU HÌNH MẠNG

I. MẠNG TCP/IP

Trong chương này các vấn đề khi thiết lập một mạng các máy tính Linux sử dụng họ giao thức TCP/IP sẽ được xem xét, bao gồm địa chỉ IP, hệ thống tên miền, chọn đường... Đây là các vấn đề cơ bản cần thiết để hiểu được những gì mà công việc cài đặt và quản trị mạng yêu cầu.

1. Giao diện mạng

Để che dấu tính phức tạp và đa dạng của các thiết bị mạng, TCP/IP định nghĩa khái niệm giao diện (interface) để truy cập vào phần cứng. Giao diện này cung cấp tập hợp các thao tác như nhau cho tất các các dạng thiết bị, dựa trên các thao tác gửi và nhận dữ liệu.

Mỗi thiết bị mạng tương ứng với ít nhất một giao diện cài đặt trong hạt nhân của hệ điều hành. Ví dụ trong Linux giao diện Ethernet là *eth0*, *eth1*..., giao diện SLIP là *sl0*, *sl1*... Tên của các giao diện trên được sử dụng cho mục đích thiết lập cấu hình khi chỉ ra cho hạt nhân thiết bị phần cứng tương ứng.

Để hoạt động trong mạng TCP/IP, mỗi giao diện mạng phải được gán một địa chỉ IP để định vị khi truyền thông với các thực thể bên ngoài.

Mỗi một giao diện có rất nhiều tham số khác nhau cần được thiết lập. Một tham số quan trọng của giao diện mạng là kích thước cực đại của gói tin mà thiết

bị phần cứng có thể xử lý (MTU -Maximum Transfer Unit). Các thuộc tính của giao diện sẽ được trình bày dần trong các phần sau.

2. Địa chỉ IP

Giao thức liên mạng IP sử dụng loại địa chỉ 32 bit. Mỗi máy trạm phải được gán một địa chỉ trong liên mạng. Khi sử dụng mạng cục bộ không kết nối với các mạng khác, người sử dụng có thể tự gán địa chỉ IP tuỳ ý cho các máy trạm. Tuy nhiên, đối với các site Internet thì địa chỉ IP phải được cung cấp từ trung tâm phụ trách địa chỉ IP trên thế giới NIC - Network Information Center.

Mỗi địa chỉ IP được chia làm 4 phần, mỗi phần 1 byte. Ví dụ máy chủ Web của trường đại học Bách Khoa Hà Nội **www.hut.edu.vn** có địa chỉ là 0xC0A87F11 hay 192.168.127.17. Dạng sau được gọi là ký pháp thập phân có chấm (dotted quad notation).

Địa chỉ IP được chia thành 2 vùng: địa chỉ mạng (network address) và địa chỉ trạm (host address). Khi đề nghị NIC cung cấp địa chỉ IP thì ta sẽ không nhận được địa chỉ tương ứng của mỗi máy trạm. Thay vào đó là địa chỉ mạng và ta có quyền gán địa chỉ cho các máy trạm của mạng trong phạm vi địa chỉ đã được cung cấp.

Tuỳ thuộc vào kích cỡ của mạng mà kích thước các phần địa chỉ là lớn hay nhỏ. Không gian địa chỉ IP được chia thành 5 lớp A, B, C, D, E.

Lớp A	Bao gồm các địa chỉ từ 1.0.0.0 đến 127.0.0.0. Địa chỉ mạng được chứa trong byte đầu tiên. Như vậy mạng loại này có 24 bit cho địa chỉ host tương đương với 1.6 triệu máy trạm.
Lớp B	Bao gồm các địa chỉ từ 128.0.0.0 đến 191.255.0.0. Địa chỉ mạng là 2 byte đầu tiên. Lớp này cung cấp 16320 mạng, mỗi mạng có thể có 65024 máy trạm.
Lớp C	Bao gồm các địa chỉ từ 192.0.0.0 đến 223.255.0.0. Địa chỉ mạng là 3 byte đầu tiên. Lớp địa chỉ này cung cấp gần 2 triệu mạng với 254 máy trạm mỗi mạng.
Lớp D, E	Bao gồm các địa chỉ từ 224.0.0.0 đến 254.0.0.0 được dùng để dự phòng trong tương lai.

Trong ví dụ trên, giả sử địa chỉ 192.168.127.17 thuộc lớp C thì địa chỉ mạng của nó là 192.168.127.0 và địa chỉ trạm là 17.

Trong danh sách địa chỉ liệt kê ở trên, không phải mọi địa chỉ đều được gán cho các trạm. Các thành phần 0 và 255 được dùng cho các mục đích đặc biệt, địa chỉ với phần trạm có các bit bằng 0 được dùng để chỉ ra mạng tương ứng. Còn địa chỉ có thành phần trạm là các bit bằng 1 là địa chỉ quảng bá, là định vị cho tất các các trạm trên mạng đó. Như vậy địa chỉ 192.168.127.255 không có nghĩa cho một trạm mà có nghĩa là tất các các máy trạm trong mạng 192.168.127.0.

Địa chỉ 0.0.0.0 được dùng cho địa chỉ chọn đường mặc định (default route) dùng trong việc chọn đường cho các gói tin IP. Còn 127.0.0.0 gọi là địa chỉ quay ngược (loopback address). Tất cả các gói tin gửi đến địa chỉ 127.0.0.0 sẽ được gửi ngược trở lại máy tính. Thông thường địa chỉ này được gán cho một giao diện riêng trên máy tính gọi là giao diện loopback. Các gói tin IP mang địa chỉ này cho dù của giao thức TCP hay UDP đều bị quay trở về giống như chúng đến từ một trạm nào đó trên mạng. Cơ chế này cho phép sử dụng máy tính đơn lẻ để kiểm tra các hoạt động của các chương trình ứng dụng trên mạng hay sử dụng các ứng dụng này trên một máy tính cá nhân riêng biệt.

3. Ánh xạ địa chỉ vật lý

Trong thực tế mỗi bộ giao tiếp mạng được gán một địa chỉ gọi là địa chỉ MAC để truyền thông trên mạng. Nó có cấu trúc khác với cấu trúc của địa chỉ IP. Địa chỉ này là một số 6 bytes (48 bits). Vấn đề đặt ra là làm thế nào để chuyển đổi từ địa chỉ IP sang địa chỉ MAC.

Giao thức được thiết kế để làm nhiệm vụ trên là giao thức ARP - Address Resolution Protocol. Giao thức này không chỉ sử dụng trong các mạng Ethernet mà còn được sử dụng trong các mạng khác để chuyển đổi từ địa chỉ IP sang địa chỉ vật lý của bộ giao tiếp mạng. Giao thức này sử dụng phương pháp quảng bá. Ý tưởng cơ bản của nó là khi cần tìm ông X trong 100 người thì ta chỉ việc gọi to tên ông ta lên và ông X sẽ trả lời nếu ông ta có ở đó.

Khi ARP muốn tìm một địa chỉ MAC tương ứng với địa chỉ IP, nó sẽ sử dụng một gói tin ARP và quảng bá tới tất cả các máy tính khác trên mạng. Trên gói tin này có chứa địa chỉ IP cần chuyển đổi. Các máy khác sẽ so sánh địa chỉ này với địa chỉ IP của chúng, nếu trùng nhau nó sẽ gửi địa chỉ MAC của nó trở lại cho trạm có yêu cầu.

Tuy nhiên để truyền thông trong liên mạng thì không thể dùng giao thức trên để ánh xạ địa chỉ. Thực tế ARP chỉ dùng trong một mạng con, còn để truyền thông trong liên mạng với hàng triệu máy trạm thì giải pháp được dùng là cơ chế chọn

đường (IP routing) sẽ được đề cập đến ở sau.

Mỗi khi một trạm tìm được một ánh xạ địa chỉ, nó sẽ lưu lại trong ARP cache để dùng trong các lần sau. Tuy vậy, cũng không nên lưu trữ địa chỉ này mãi vì bộ giao tiếp mạng của máy trạm ở xa có thể bị thay đổi vì lý do nào đó. Do vậy, ánh xạ trên chỉ được lưu trữ trong thời gian nhất định, sau đó phải được cập nhật.

Trong một số trường hợp người ta lại cần tìm ánh xạ ngược: tìm một địa chỉ IP tương ứng với một địa chỉ MAC. Chẳng hạn trong trường hợp khởi động máy tính qua mạng. Máy tính cần khởi động chỉ có thông tin về địa chỉ MAC, nó sẽ gửi các gói tin quảng bá và một boot server sẽ tìm địa chỉ IP tương ứng cho nó. Giao thức này gọi là RARP - Reverse Address Resolution Protocol. Cùng với giao thức BOOTP, RAPP được sử dụng rộng rãi trong việc khởi động một máy tính không có ổ cứng thông qua mạng.

4. Chọn đường

4.1. Giới thiệu mạng IP

Trong thực tế, địa chỉ của một tổ chức hay cá nhân nào đó đều được tổ chức theo kiểu phân cấp: nước, tỉnh, thành phố, quận, phường ... Làm như vậy rất có lợi cho các nhà bưu điện khi muốn chuyển thư tín: Việc chuyển thư ở mỗi cấp chỉ việc quan tâm đến đường đi của thư trong phạm vi của mình.

Các mạng IP cũng được tổ chức theo cơ chế địa chỉ phân cấp như vậy. Mạng IP, hay còn gọi là Internet bao gồm rất nhiều các mạng riêng, mỗi mạng riêng lại được chia làm các mạng con nhỏ hơn, mỗi mạng riêng thực hiện các công việc về địa chỉ trong nội bộ mạng đó. Sự phân cấp này cho phép giảm lượng công việc chọn đường cho các gói tin IP trong toàn liên mạng.

4.2. Mạng con (subnetwork)

Địa chỉ IP được chia làm 2 phần như đã giới thiệu ở trước: Địa chỉ mạng và địa chỉ trạm. Điều này cho phép chia các mạng IP thành các mạng con. Theo mặc định thì mạng con được đặc trưng bởi phần địa chỉ mạng trong địa chỉ IP. Do vậy, các máy trạm trong cùng một mạng con sẽ có chung địa chỉ mạng và ngược lại các máy trạm có cùng địa chỉ mạng sẽ nằm trên cùng một mạng con.

Mỗi mạng con chịu trách nhiệm cho việc chọn đường cho các gói tin IP

trong mạng của mình, các gói tin này được nhận ra nhờ phần địa chỉ mạng của nó. Trong các mạng loại A, B, C thì phần địa chỉ này có độ dài cố định. Tuy nhiên để tạo sự linh hoạt trong việc phân chia mạng con thì địa chỉ mạng có thể mở rộng sang các bit của địa chỉ trạm. Đó là kỹ thuật sử dụng mặt nạ mạng con (subnet mask hay netmask). Mặt nạ là một số 32 bit trong đó các bit tương ứng với phần địa chỉ mạng bằng 1, các bit còn lại bằng 0.

Ví dụ: Giả sử trường đại học Bách Khoa Hà Nội có một mạng loại B với địa chỉ mạng là 192.169.127.0 và mặt nạ là 255.255.255.0 (địa chỉ mạng dài 24 bit).

Trường có 4 khoa có nhu cầu sử dụng mạng cục bộ riêng, do đó mạng trên sẽ được phân là 4 mạng con bằng việc lấy thêm 2 bit cho địa chỉ mạng (26 bit). Như vậy địa chỉ mạng của khoa Tin học là 192.169.50.0, của khoa Điện tử là 192.169.50. 64, của khoa Vật lý là 192.169.50.128, của khoa Năng lượng là 192.169.50.192. Mặt nạ của các mạng con này là 255.255.255.192.

	192	169	50	0
Bách Khoa	11000000	10100000	00011010	00000000
	192	169	50	128
Khoa Vật Lý	11000000	10100000	00011010	10000000
	192	169	50	168
Khoa Năng lượng	11000000	10100000	00011010	11000000

Hình 3.1.

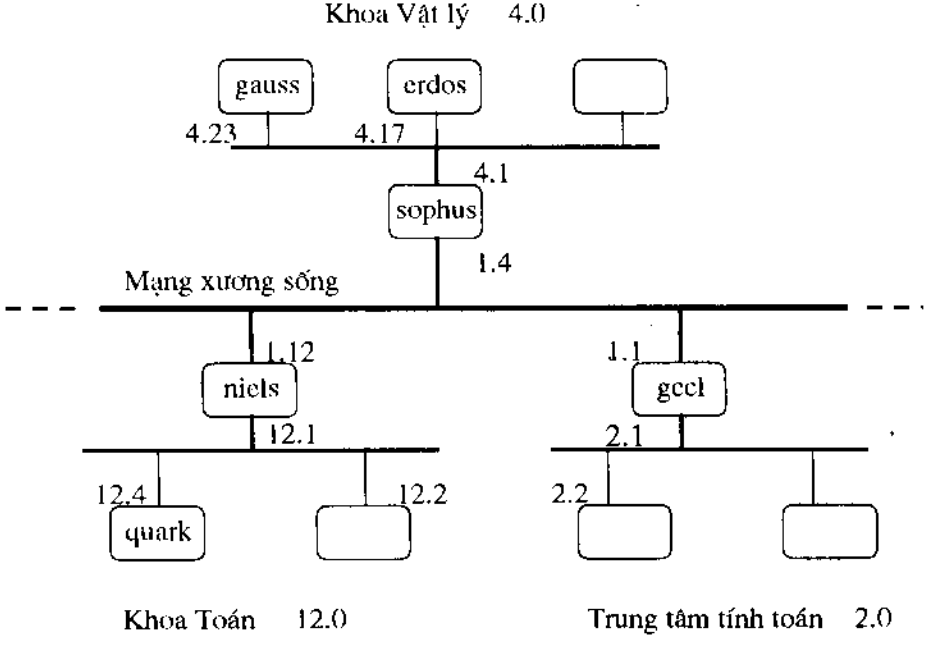
Sự phân chia mạng riêng thành các mạng con chỉ có ý nghĩa bên trong mạng riêng đó, nó là trong suốt đối với thế giới bên ngoài. Việc chia mạng được tiến hành bởi người quản trị hệ thống và thường dựa trên sự phân chia khác như ranh giới vật lý giữa các nhánh mạng, phân chia hành chính giữa các phòng ban hay vì mục tiêu an toàn bảo mật thông tin trên mạng.

4.3. Gateway

Các máy tính trong một mạng chỉ có thể liên lạc trực tiếp với những máy tính trong mạng đó. Việc trao đổi thông tin với những máy tính trên các mạng khác được thực hiện thông qua một trạm trung gian - gateway. Đây là một máy tính hoặc thiết bị chuyên dụng nối giữa hai hay nhiều mạng vật lý và được thiết lập cấu hình để chuyển các gói tin giữa các mạng đó.

Có thể xác định dễ dàng một trạm có thuộc một mạng hay không nhờ vào địa chỉ IP của trạm đó. Mỗi mạng được đặc trưng bởi một địa chỉ IP và các mạng khác nhau có các địa chỉ IP khác nhau. Trong ví dụ dưới đây mạng LAN của khoa Toán có địa chỉ mạng 149.76.4.0. Trạm edos muốn gửi một gói tin cho trạm quark có địa chỉ 149.76.12.4 sẽ biết được trạm này ở trên mạng khác và sẽ gửi gói tin này đến gateway mặc định của mạng : sophus.

Dưới đây là ví dụ một phần hình trạng mạng của trường đại học Groucho Maxk:



Hình 3.2 . Một phần hình trạng mạng của GMU

Trạm sophus là một máy tính nối với cả 2 mạng: mạng LAN của khoa toán và mạng xương sống của trường đại học. Nó sẽ có 2 giao diện mạng là *eth0* và *fddi0*. Tương ứng với các giao diện đó nó sẽ có 2 địa chỉ IP là 149.76. 4.1 và 149.76.1.4. Như vậy các gateway thường có ít nhất 2 địa chỉ IP và khi truy cập vào mạng nào nó sẽ dùng địa chỉ IP tương ứng.

Bảng 3.1. Các giao diện mạng của gateway

Interface	Address	Netmark
eth0	149.76.4.1	255.255.255.0
fddi0	149.76.1.4	255.255.255.0
lo	127.0.0.1	255.0.0.0

Cũng cần phân biệt sự khác nhau giữa địa chỉ IP của một máy trạm và địa chỉ IP của một giao diện. Thông thường trong mạng LAN, các trạm chỉ có một giao diện thì hai địa chỉ trên trên là trùng nhau. Tuy nhiên đối với các gateway có từ hai giao diện mạng thì chúng là khác nhau, vì mỗi trạm sẽ có nhiều địa chỉ IP ứng với mỗi giao diện.

4.4. Bảng chọn đường

Phần này sẽ trình bày việc sử dụng địa chỉ IP khi một trạm chọn một gateway để gửi các gói tin cho một trạm ở xa.

Trong ví dụ ở phần trước, khi trạm **erdos** muốn gửi một gói tin cho trạm quark, nó xem xét địa chỉ của **quark** và nhận thấy trạm đích không thuộc mạng cục bộ. Do đó **erdos** sẽ gửi gói tin này đến gateway mặc định, **sophus**. Đến lượt mình **sophus** sẽ tìm đường cho gói tin và nhận thấy cần phải gửi nó đến gateway nels của khoa vật lý. Để làm được như vậy **sophus** phải có những thông tin về địa chỉ trạm đích và các gateway tương ứng: thông tin đó được tổ chức trong bảng chọn đường.

Bảng chọn đường dùng trong giao thức IP là một bảng cho biết liên hệ giữa gateway và những mạng mà nó nối vào. Bảng chọn đường của trạm **sophus** như sau:

Bảng 3.2. Hình ảnh của một bảng chọn đường

Network	Gateway	Interface
149.76.1.0	-	fddi0
149.76.2.0	149.76.1.2	fddi0
149.76.3.0	149.76.1.3	fddi0
149.76.4.0	-	eth0
149.76.5.0	149.76.1.5	fddi0
. . .	. . .	. . .
0.0.0.0	149.76.1.2	fddi0

Thành phần 0.0.0.0 gọi là thành phần mặc định, tất cả các gói tin có địa chỉ đích thuộc mạng không được chỉ ra trong bảng chọn đường sẽ được gửi đến gateway mặc định.

Việc chọn đường tới các trạm trên mạng mà **sophus** nối trực tiếp thì sẽ không cần gateway. Do đó trên bảng chọn đường ta ký hiệu là ký tự " - ".

Có nhiều cách khác nhau để xây dựng bảng chọn đường. Đối với các mạng nhỏ như mạng cục bộ thì ta có thể làm thủ công bằng cách thêm từng thành phần vào trong bảng chọn đường. Đối với các mạng cỡ lớn thì việc xây dựng bảng chọn đường được thực hiện nhờ các chương trình gọi là *routing daemon*. Cơ chế hoạt động của chương trình này phụ thuộc vào các chiến lược chọn đường được thiết kế cho mạng như chọn đường tập trung hay phân tán, thích nghi hay không thích nghi...

Tuỳ thuộc vào kích cỡ của mạng mà các giao thức chọn đường khác nhau sẽ được sử dụng. Đối với việc chọn đường trong các hệ tự trị (autonomous system) thì sử dụng các giao thức internal routing protocol. Giao thức thông dụng trong loại giao thức trên là RIP - Routing Information Protocol, giao thức này được cài đặt trong chương trình *BSD routed*. Việc chọn đường giữa các hệ tự trị sử dụng các giao thức như EGP (External Gateway Protocol) và BGP (Border Gateway Protocol). Các giao thức này được cài đặt trong *gated*, là một chương trình phát triển bởi trường đại học Cornel.

5. Hệ thống tên miền DNS

5.1. Ánh xạ địa chỉ

Việc định danh các phần tử trong liên mạng bằng các con số như địa chỉ IP rõ ràng là làm người dùng khó nhớ và dễ nhầm lẫn. Chính vì vậy, trong thực tế các máy trạm thường được định danh bằng tên và ta xây dựng ứng dụng để chuyển đổi từ tên máy sang địa chỉ IP tương ứng. Quá trình đó gọi là ánh xạ địa chỉ (hostname resolution).

Các hàm cho quá trình chuyển đổi trên được xây dựng sẵn và để trong thư viện. Một ứng dụng muốn tìm địa chỉ IP tương ứng của một trạm làm việc không nhất thiết phải xây dựng các hàm riêng mà sẽ sử dụng các thư viện này. Các hàm thường dùng là *gethosthyname()* và *gethosthyaddr()*. Như vậy, quá trình chuyển đổi tên là trong suốt đối với những chương trình ứng dụng.

Trong các mạng LAN Ethernet cỡ nhỏ, việc ánh xạ giữa tên máy trạm và địa chỉ IP được thực hiện rất đơn giản. Các thông tin này được lưu trong tệp */etc/hosts* theo cấu trúc tên máy-địa chỉ IP trên tất cả các trạm. Khi một máy tính được thêm vào hay loại bỏ khỏi mạng, các tệp này phải được sửa đổi trên tất cả các trạm còn lại. Công việc này sẽ trở nên bất tiện khi mạng có nhiều máy tính.

Một giải pháp cho vấn đề này là sử dụng dịch vụ thông tin mạng NIS - Network Information System. NIS còn gọi là YP - Yellow Page được phát triển bởi Sun Microsystem. NIS lưu trữ thông tin về tên và địa chỉ của các máy trạm (và nhiều thông tin khác nữa như tài khoản...) vào trong một cơ sở dữ liệu và đặt trong một trạm trung tâm. Các trạm khác sẽ truy cập tới cơ sở dữ liệu này để lấy thông tin. Giải pháp này được dùng cho các mạng cỡ vừa như mạng LAN vì trạm trung tâm sẽ trở nên quá tải do tất cả các trạm khác sẽ liên tục truy cập tới cơ sở dữ liệu trung tâm này.

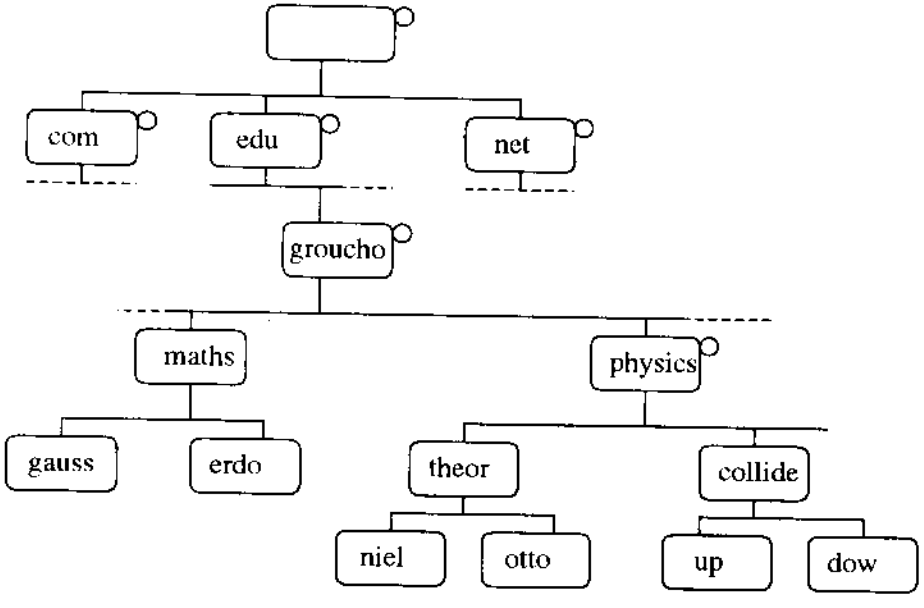
Trong mạng toàn cầu Internet, ban đầu một cơ sở dữ liệu địa chỉ như trên cũng được xây dựng trong một tệp HOSTS.TXT. Tệp này được đặt tại Trung tâm thông tin mạng NIC. Tất cả các thành viên tham gia Internet đều phải truy cập vào cơ sở dữ liệu này. Khi mạng Internet mở rộng ra thì giải pháp này trở nên không thích hợp nữa. Ngoài việc phải chi phí rất lớn cho việc quản trị tại NIC, máy chủ phục vụ dữ liệu này đã trở nên quá tải.

Vì những lý do đó mà một giải pháp khác đã ra đời. DNS - hệ thống tên

miền được phát triển từ năm 1984 bởi Paul Mockapetris. Hệ này đã giải quyết được hai vấn đề đặt ra là giảm chi phí quản lý tập trung và mỗi máy trạm được định danh duy nhất.

5.2. Hệ thống tên miền

DNS tổ chức việc định danh bằng tên theo hệ thống phân cấp các miền. Mỗi hệ thống trong một cấp được gọi là một miền. Miền là tập hợp các site có quan hệ với nhau theo một nghĩa nào đó. Các site này tạo nên một mạng riêng cho hệ thống đó, chẳng hạn các máy tính của một trường đại học, một tổ chức của chính phủ... Ví dụ các trường đại học được tập trung vào một miền gọi là miền **edu**, mỗi miền được chia làm các miền con (subdomain) và cứ thế đi xuống. Trường đại học GMU được gán tên miền **groucho.edu**, khoa Toán của trường là miền **maths.groucho.echo.** Các máy trạm tại khoa Toán được định danh là tên máy và tên miền, ví dụ **erdos.maths.groucho.edu**, hay còn gọi là tên miền đầy đủ (FQDN - fully qualified domain name). Tên miền đầy đủ cho phép định danh duy nhất một máy trạm trong mạng toàn cầu.



Hình 3.3. Tổ chức tên miền của GMU

Tuỳ thuộc vào vị trí của miền trong hệ thống phân cấp mà nó có thể là miền cấp 1, cấp 2 hay cấp 3. Thông thường hệ thống này chỉ phân đến cấp 3, ít khi phân đến cấp nhỏ hơn. Các tên miền thường gặp ở cấp 1 là:

edu	(Chủ yếu ở Mỹ) các trường đại học
com	Tổ chức thương mại, công ty kinh doanh
org	Các tổ chức phi thương mại
net	Các gateway, hoặc các máy trạm quản trị mạng
mil	Tổ chức quân sự Mỹ
gov	Chính phủ Mỹ

Thông thường bốn miền đầu thuộc về các site của Mỹ, song vẫn có thể có một số site không thuộc Mỹ nằm trong các miền này, chẳng hạn như miền **net**. Các miền **mil** và **gov** chỉ sử dụng cho các site của Mỹ.

Đối với các site của các nước khác thì miền cấp cao nhất được định danh bằng mã 2 chữ cái đặc trưng của tên nước đó, mã này được quy định trong tiêu chuẩn ISO-3166. Ví dụ vn -Việt Nam, fr - Pháp, fi - Phần Lan ... Các miền cấp 2 của Việt Nam được định danh giống như các miền cấp 1 của quốc tế như **edu.vn**, **com.vn**, ... Tuy nhiên một số nước lại không sử dụng quy ước trên cho các miền cấp dưới của mình mà sử dụng luôn tên dài đặc trưng cho tổ chức sở hữu nó, chẳng hạn tên của một site ở Đức là **ftp.infomatik.uni-erlangen.de**

Tên miền của máy trạm mang mã của một quốc gia không cho ta biết trạm này được định vị tại nước đó. Tên miền chỉ có ý nghĩa rằng máy trạm đó được đăng ký định danh tại trung tâm NIC của nước đó.Ví dụ, một công ty của Pháp có chi nhánh đặt tại Việt Nam nhưng máy trạm của chi nhánh đó có tên miền cấp cao nhất là **fr.**

Rõ ràng với hệ thống tổ chức phân cấp như trên thì mỗi máy trạm sẽ được định danh duy nhất trên thế giới và không thể có tình trạng hai máy trên mạng toàn cầu có cùng tên được. Ngoài ra việc chia thành các miền con cho phép chúng ta nhớ tên máy trạm một cách dễ dàng hơn. Không những thế DNS còn cho phép uỷ nhiệm quyền quản trị mạng của miền con cho người quản trị mạng đó. Ví dụ người quản lý mạng của trường đại học Groucho muốn chia mạng của trường thành các miền con cho mỗi khoa trong đó có khoa Vật lý và khoa Toán. Họ nhận thấy khoa Vật lý có quá nhiều máy trạm và rất khó có thể thực hiện công việc quản trị từ bên ngoài, do đó quyền quản trị miền **physics.groucho.edu** được giao cho người quản

trị của mạng này. Khi đó việc đặt tên cho các máy trạm trong miền con này cũng như cách thức gán địa chỉ IP cho chúng là tuỳ ý và không phụ thuộc vào các yếu tố bên ngoài khác.

Không gian tên còn được chia thành các vùng (zone). Vùng là tập hợp tất cả các máy trạm được quản lý trực tiếp bởi một miền con. Cần phân biệt sự khác nhau giữa vùng và miền. Ví dụ miền **groucho.edu** bao gồm tất cả các máy trạm thuộc trường đại học Groucho Mark, còn vùng **groucho.edu** bao gồm tất cả các máy trạm được quản lý trực tiếp bởi trường, còn các máy tính thuộc khoa vật lý thuộc một vùng khác tên là **physics.groucho.edu**. Trên hình 3.3 bắt đầu của một vùng được đánh dấu bởi vòng tròn bên cạnh tên miền.

5.3. Ánh xạ địa chỉ trong DNS

Thực tế, DNS là một cơ sở dữ liệu phân tán khổng lồ, nó được cài đặt trên các máy chủ gọi là Name Server trên đó có chứa các thông tin của một hay nhiều miền. Mỗi vùng phải có ít nhất hai Name Server lưu giữ các thông tin về những máy trạm thuộc vùng đó. Ví dụ để tìm địa chỉ IP của máy trạm **erdos** thuộc vùng **groucho.edu**, chỉ cần gửi yêu cầu đến Name Server của vùng này.

Khi một máy trạm muốn tìm địa chỉ IP của một trạm khác, chẳng hạn **erdos.maths.groucho.edu**, trước tiên nó sẽ gửi yêu cầu tới máy chủ Name Server cục bộ trong vùng của mình. Nếu máy chủ này không biết thông tin về **erdos**, nó sẽ gửi yêu cầu ánh xạ địa chỉ đến máy chủ ở miền gốc. Máy chủ này nhận ra **erdos.maths.grouch.edu** không thuộc vùng quản lý của mình nhưng thuộc miền con **edu**, nó sẽ gửi trở lại danh sách các Name Server của vùng **edu**. Máy chủ cục bộ sẽ gửi một yêu cầu ánh xạ địa chỉ đến một máy chủ trong danh sách trên, ví dụ **a.asi.edu**. Trạm này có thông tin về Name Server của **groucho.edu** và gửi địa chỉ IP của máy chủ này về máy chủ cục bộ. Cuối cùng Name Server cục bộ sẽ gửi yêu cầu tới Name Server của trường đại học Groucho để lấy địa chỉ IP của **erdos.maths.groucho.edu.**

Để giảm thời gian tìm kiếm trên mạng, các Name Server đều tổ chức lưu lại thông tin đã tìm kiếm trước đó vào trong cache. Mỗi khi có yêu cầu tìm kiếm địa chỉ của một trạm trong miền **groucho.edu**, Name Server cục bộ sẽ không phải tiến hành tìm kiếm trên mạng như trước mà sẽ truy vấn trực tiếp đến Name Server của miền này nhờ địa chỉ của nó đã lưu trong cache.

Tuy nhiên Name Server sẽ không duy trì mãi thông tin trên. Sau mỗi khoảng

thời gian nhất định thông tin đó sẽ bị loại bỏ khỏi cache. Khoảng thời gian đó gọi là thời gian sống (TTL - Time To Live). Mỗi thành phần trong cơ sở dữ liệu của DNS sẽ được người quản trị mạng gán cho một giá trị TTL nhất định.

5.4. Domain Name Servers

Trong nội bộ một vùng luôn có một số máy chủ nhất định để lưu trữ các thông tin về tên máy - địa chỉ IP của tất cả các máy trạm trong vùng đó. Các máy chủ này gọi là master server. Tất cả các yêu cầu tìm kiếm ánh xạ địa chỉ về các trạm trong vùng cuối cùng phải đến được các máy chủ này.

Để các thông tin do các Name Server cung cấp là chính xác và nhất quán thì chúng phải được đồng bộ hoá. Một trong chúng đóng vai trò máy chủ sơ cấp (primary server), các máy còn lại là máy chủ thứ cấp (secondary server). Máy chủ sơ cấp có nhiệm vụ nạp các thông tin trong vùng từ các tệp dữ liệu, còn máy chủ thứ cấp sẽ nhận các thông tin trên từ máy chủ sơ cấp trong những khoảng thời gian nhất định.

Việc tổ chức nhiều Name Server như vậy trong một vùng có tác dụng phân tán tải, đồng thời tăng mức dư cho mạng. Khi một máy có sự cố thì các máy khác sẽ có nhiệm vụ duy trì hoạt động đáp ứng dịch vụ.

Trong một số trường hợp chúng ta có thể tổ chức một Name Server cho riêng một mạng cục bộ. Các máy chủ này không có tác dụng đáp ứng các yêu cầu DNS cũng như tìm kiếm thông tin DNS từ Internet mà chỉ đáp ứng các yêu cầu DNS của các ứng dụng trong mạng nội bộ. Khi đó nó chỉ cần lưu trữ các thông tin trong mạng cục bộ mà thôi (cache-only name server).

5.5. Cơ sở dữ liệu DNS

DNS không chỉ có nhiệm vụ thực hiện ánh xạ tên trạm - địa chỉ mà còn thực hiện trao đổi thông tin giữa các Name Server. Trong thực tế cơ sở dữ liệu của DNS có rất nhiều thành phần với cấu trúc khác nhau.

Thành phần cơ bản của cơ sở dữ liệu DNS là bản ghi nguồn RR (Resource Record). Mỗi bản ghi có một kiểu dữ liệu (dạng bản ghi) và tương ứng với một loại mạng (lớp bản ghi). Lớp bản ghi đặc trưng cho dạng địa chỉ mà mạng tương ứng với lớp đó sử dụng như địa chỉ IP (lớp IN), địa chỉ mạng Hesiod (dùng tại MIT)... Dạng bản ghi thông dụng là bản ghi loại A, là một cặp tên miền định danh đầy đủ (FQDN) - địa chỉ IP.

Trong liên mạng, một trạm có thể có nhiều hơn một tên. Trong đó có một tên phải được khai báo là chính thức (official hay canonical host name), các tên còn lại chỉ là các bí danh của nó (aliases). Các tên chính thức được khai báo trong các bản ghi loại A, còn các tên bí danh được khai báo trong các bản ghi loại CNAME trỏ tới bản ghi loại A tương ứng.

Trong phần này chúng ta chỉ xem xét một số ví dụ về các loại bản ghi, các loại khác sẽ được nghiên cứu kỹ hơn trong phần cài đặt dịch vụ DNS. Dưới đây là ví dụ về một phần của cơ sở dữ liệu DNS chứa trên Name Server của miền **physics.groucho.edu.**

```
;
; Các thông tin chung của physics.groucho.edu
@               IN    SOA          (
                niels.physics.groucho.edu.
                hostmaster.niels.physics.groucho.edu.
                1034               ; serial no
                360000             ; refresh
                3600               ; retry
                3600000            ; expire
                3600               ; default ttl
                                  )
;
; Name servers
                IN    NS       niels
                IN    NS       gauss.maths.groucho.edu.
gauss.maths.groucho.edu. IN A 149.76.4.23
;
; Mạng của khoa Vật lý (subnet 12)
niels           IN    A        149.76.12.1
                IN    A        149.76.1.12
nameserver      IN    CNAME    niels
otto            IN    A        149.76.12.2
```

```
quark           IN      A       149.76.12.4
down            IN      A       149.76.12.5
strange         IN      A       149.76.12.6
...
 ; Mạng con Collider (subnet 14)
boson           IN      A       149.76.14.1
muon            IN      A       149.76.14.7
bogon           IN      A       149.76.14.12
...
```

Ngoài các bản ghi loại A và CNAME, trong ví dụ trên còn có bản ghi loại SOA (Start of Authority). Bản ghi này chứa các thông tin chung về vùng mà Name Server có thẩm quyền. Trong bản ghi này có chứa thời gian tồn tại mặc định TTL cho tất cả các bản ghi.

Các tên không có dấu chấm (.) ở cuối tên trong ví dụ trên là các tên tương đối trong miền **groucho.edu**. Ký tự đặc biệt @ cho biết đây là bản ghi này tự trỏ đến chính tên của miền đó.

Trong miền **groucho.edu** có một miền con cho khoa Vật lý là **physics.groucho.edu**. Để Name Server của miền **groucho.edu** có thể biết được thông tin về Name Server của miền con của nó, DNS tổ chức một cặp bản ghi loại A và loại NS. Bản ghi loại NS chứa thông tin về tên miền đầy đủ FQDN của Name Server, còn bản ghi loại A chứa thông tin về tên và địa chỉ IP của máy chủ đó. Cặp bản ghi này gọi là các bản ghi liên kết (glue record), chúng chỉ được dùng khi một vùng muốn chứa các thông tin về Name Server của vùng con của nó. Cặp bản ghi liên kết của Name Server cho vùng con **physics.groucho.edu** như sau:

```
;
; Vùng dữ liệu của groucho.edu zone.
@               IN      SOA             (
                vax12.gcc.groucho.edu.
                hostmaster.vax12.gcc.groucho.edu.
                233                     ; serial no
                360000                  ; refresh
                3600                    ; retry
```

```
                3600000         ; expire
                3600            ; default ttl
                )
....
;
 ; Bản ghi liên kết của vùng physics.groucho.edu
physics         IN    NS        niels.physics.groucho.edu.
                IN    NS        gauss.maths.groucho.edu.
niels.physics   IN     A        149.76.12.1
gauss.maths     IN     A        149.76.4.23
...
```

5.6. Ánh xạ địa chỉ ngược

Ánh xạ địa chỉ ngược chuyển đổi từ địa chỉ IP sang tên chính thức của một máy trạm. Khi sử dụng một tệp */etc/hosts* thì việc ánh xạ ngược chỉ đơn giản là tìm kiếm tệp cho đến khi tìm ra địa chỉ IP tương ứng. DNS không tiến hành tìm kiếm vết cạn không gian tên như trên. Thay vào đó nó tổ chức một miền đặc biệt gọi là *in-addr.arpa*, miền này chứa địa chỉ IP của tất cả các máy trạm theo ký pháp thập phân có chấm đảo ngược (reverted dotted-quad notation). Ví dụ địa chỉ 149.76.12.4 tương ứng với tên **4.12.76.149.in-addr.arpa.** Bản ghi nguồn liên kết tên dạng trên với tên chính thức của máy trạm là bản ghi loại PTR.

Khi tạo ra một vùng có thẩm quyền riêng, người quản trị mạng ở vùng đó có quyền chia vùng của mình ra nhiều mạng con. Ví dụ khoa Vật lý được chia ra thành các mạng con với địa chỉ mạng tương ứng là 149.76.8.0, 149.76.12.0 và 149.76.14.0. Với mỗi mạng con cần tạo ra một vùng in-addr.arpa, đó là : **8.76.149.in-addr.arpa, 12.76.149.in-addr.arpa, 14.76.149.in-addr.arpa.** Mỗi vùng sẽ chứa các bản ghi PTR đối với các trạm trong mạng con. Sau đây là ví dụ về vùng in-addr.arpa của mạng con 149.76.12:

```
;
 ; Miền 12.76.149.in-addr.arpa
@               IN      SOA     (
```

```
                niels.physics.groucho.edu.
                hostmaster.niels.physics.groucho.edu.
                233 360000 3600 3600000 3600
                )
2               IN      PTR     otto.physics.groucho.edu.
4               IN      PTR     quark.physics.groucho.edu.
5               IN      PTR     down.physics.groucho.edu.
6               IN      PTR     strange.physics.groucho.edu.
```

Tệp tin cơ sở dữ liệu trên máy chủ DNS ở vùng cha 149.76 có dạng như sau:

```
;
 ; Miền 76.149.in-addr.arpa
@               IN    SOA           (
                vax12.gcc.groucho.edu.
                hostmaster.vax12.gcc.groucho.edu.
                233 360000 3600 3600000 3600
                )
...
 ; subnet 4: Khoa Toán
1.4             IN      PTR     sophus.maths.groucho.edu.
17.4            IN      PTR     erdos.maths.groucho.edu.
23.4            IN      PTR     gauss.maths.groucho.edu.
...
 ; subnet 12: Khoa Vật lý
12              IN      NS      niels.physics.groucho.edu.
                IN      NS      gauss.maths.groucho.edu.
niels.physics.groucho.edu. IN  A 149.76.12.1
gauss.maths.groucho.edu. IN  A   149.76.4.23
...
```

Một hệ quả của việc sử dụng miền in-addr.arpa trong phép ánh xạ địa hỉ

ngược là: Chỉ có thể tạo ra một vùng ứng với một mạng con với số bit mặt nạ là bội của 8 (byte boundary). Trong ví dụ trên các mạng con của trường đại học Groucho Marx đều có mặt nạ là 255.255.255.0, do đó vùng in-addr.arpa mới có thể được tạo ra cho mỗi mạng con. Nếu mặt nạ là 255.255.255.128 chẳng hạn thì việc tạo vùng cho mạng con 149.76.12.128 là không thể được vì không có cách nào báo cho DNS biết được vùng **12.76.149.in-addr.arpa** đã được chia làm 2 miền con, với các máy trạm có khoản địa chỉ từ 1 đến 127 và từ 128 đến 255.

II. THIẾT LẬP CẤU HÌNH PHẦN CỨNG

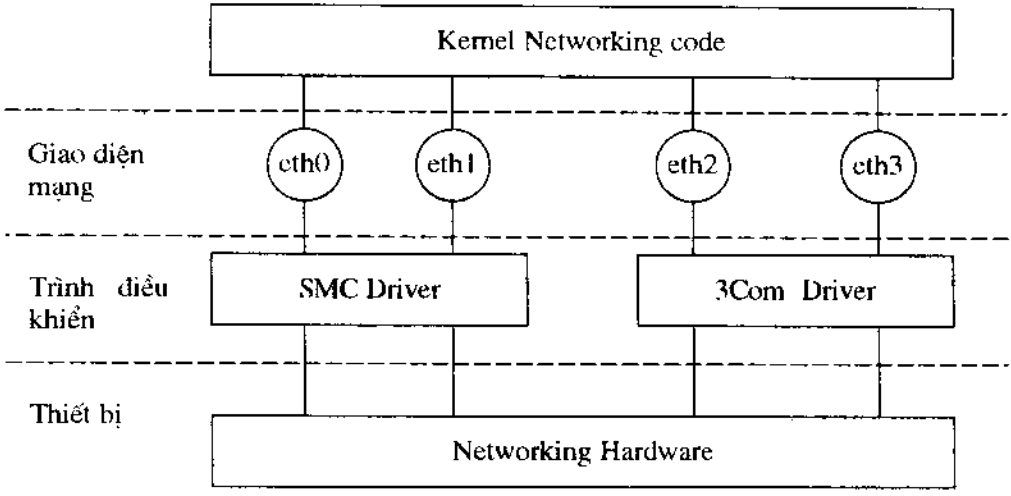
1. Thiết bị, trình điều khiển và giao diện

Trong phần trước chúng ta đã nói qua một số khái niệm về giao diện mạng cũng như một số vấn đề cơ bản của giao thức TCP/IP, nhưng chưa giải thích làm thế nào để các chương trình của hạt nhân truy cập vào thiết bị phần cứng. Điều này liên quan trực tiếp đến các khái niệm về giao diện và trình điều khiển.

Trước tiên là khái niệm về thiết bị phần cứng, ví dụ card mạng Ethernet. Đó là một tập hợp các thiết bị điện tử, các chip điều khiển... đặt trên một vỉ bằng nhựa tổng hợp và cắm vào máy tính qua một khe cắm.

Để có thể sử dụng được card mạng Ethernet, một số các hàm đặc biệt phải được cài đặt trong hạt nhân của Linux để hạt nhân có thể biết được cách thức truy cập vào thiết bị phần cứng, đó là trình điều khiển. Chẳng hạn đối với các thiết bị thuộc họ Ethernet, Linux có một số các trình điều khiển tương tự nhau. Chúng được xếp vào họ các trình điều khiển Becker (Becker Series Drivers). Trình điều khiển thiết bị cổng song song có tên là D-Link, được viết bởi tác giả là Donald Becker.

Các trình điều khiển hoạt động bằng cách gửi lệnh và dữ liệu đến thiết bị, đồng thời nhận các dữ liệu là kết quả hoạt động của thiết bị. Trên các máy tính PC, việc truyền thông giữa trình điều khiển và thiết bị được thực hiện qua một vùng nhớ vào ra (I/O). Vùng nhớ này thường được ánh xạ địa chỉ lên các thanh ghi vào ra. Các lệnh cũng như dữ liệu trao đổi giữa chúng đều được truyền qua các thanh ghi trên. Địa chỉ của vùng nhớ vào ra thường được bắt đầu bởi địa chỉ cơ sở, thông thường địa chỉ cơ sở của vỉ mạch Ethernet là 0x300 hoặc 0x360.



Hình 3. 4. Giao diện, trình điều khiển và thiết bị

Việc xác định địa chỉ cơ sở được hạt nhân thực hiện trong quá trình khởi động. Quá trình này được gọi là tự động thăm dò (autoprobing). Tuy nhiên một số vi mạch thuộc thế hệ mới hoặc không được hạt nhân hỗ trợ thì hạt nhân không thể phát hiện ra được. Một số phiên bản của Linux cũng không hỗ trợ việc tự động phát hiện hai card mạng Ethernet cùng lúc. Do đó nếu có hai card Ethernet thì ta phải tự khai báo thiết bị thứ hai.

Mỗi một thiết bị ngoại vi có một tham số quan trọng là số hiệu ngắt (IRQ - Interrupt Request Number). Thiết bị phần cứng thường sử dụng ngắt trong trường hợp chúng cần được điều khiển như có dữ liệu tới, lỗi phần cứng... Tuỳ thuộc vào từng thế hệ máy PC mà mỗi số hiệu ngắt được gán cho các thiết bị hay công việc khác nhau.

Như đã mô tả trong chương trước, hạt nhân truy cập vào các trình điều khiển thiết bị thông qua các giao diện. Các giao diện cung cấp các hàm vào ra giống nhau cho tất cả các dạng thiết bị phần cứng nào đó, chẳng hạn hàm nhận hay hàm truyền một gói tin.

Các giao diện được định danh bởi tên. Các tên này được định nghĩa bên trong hạt nhân. Giao diện Ethernet có tên là *eth0, eth1*, ... Việc gán tên cho giao diện tuỳ thuộc vào thứ tự mà giao diện đó được thiết lập cấu hình, chẳng hạn vì

Ethernet thứ nhất được cài đặt với giao diện *eth0*, vỉ tiếp theo với giao diện *eth1*... Ngoại lệ của quy tắc này là giao diện SLIP, nó được gán tên động. Mỗi khi có một kết nối SLIP được thiết lập thì một giao diện tương ứng sẽ được gán cho cổng nối tiếp.

Khi khởi động, hạt nhân sẽ thông báo các thiết bị mà nó phát hiện ra được, cùng với tên giao diện mà nó cài đặt. Ví dụ:

```
This processor honours the WP bit even when in supervisor
mode. Good.

Floppy drive(s): fd0 is 1.44M

Swansea University Computer Society NET3.010

IP Protocols: ICMP,UDP, TCP

PPP: version 0.2.1(4 channels) OPTIMIZE_FLAGS

TCP compresion code copyright 1989 Regents of University of
California

PPP line discipline registered

SLIP: version 0.7.5 (4 channels)

CSLIP: code copyright 1989 Regents of University of
California

dl0: D-link DE-600 pocket adapter, Ethernet Address:
00:80:C8:71:76:95

3c59 x.c:v0.99H 11/17/98 Donald Becker

eth0: 3Com 3c595 Vortex 100baseTx at
0x1020,00:60:97:ac:25:b3,IRQ 11

64K word-wide RAM 3:1 Rx:Tx split, 10baseT interface.

eth0: Overriding PCI latency timer (CFLT) setting of 64,
new value is 248.
```

Trong ví dụ trên, hạt nhân được biên dịch với tuỳ chọn hỗ trợ TCP/IP, bao gồm trình điều khiển cho SLIP, CSLIP, PPP. Dòng cuối cùng cho biết hạt nhân phát hiện ra một bộ điều hợp D-Link Do đó nó cài giao diện *dl0*. Nếu có một vỉ mạch Ethernet thì hạt nhân sẽ cho hiển thị dòng bắt đầu bằng *eth0* và dạng thiết bị phát hiện được. Trong trường hợp có một thiết bị Ethernet mà không có thông báo trên chứng tỏ hạt nhân không có khả năng phát hiện nó.

2. Cấu hình cho hạt nhân

Nói chung, các nhà cung cấp hệ điều hành Linux đều đưa ra hạt nhân hỗ trợ hầu hết các loại thiết bị PC thông dụng. Có nghĩa là hạt nhân đã có tương đối đầy đủ các trình điều khiển cần thiết. Song điều này sẽ làm lãng phí bộ nhớ vì các tất cả các thành phần của hạt nhân được nạp một lần lúc khởi động, chúng không thể sử dụng trong bộ nhớ hoán đổi. Vì vậy nên cấu hình hạt nhân với vừa đủ các trình điều khiển cần thiết để tăng tốc độ làm việc.

Xây dựng lại hạt nhân là công việc rất hay làm với hệ điều hành Linux. Các thao tác cơ bản có thể tham khảo chi tiết tại tài liệu "Installation and Getting Started" của Matt Welsh. Trong phần này chỉ đưa ra một số tham số có liên quan đến cấu hình hạt nhân ảnh hưởng tới hoạt động mạng.

Khi chúng ta sử dụng lệnh *make config* để cấu hình cho hạt nhân, trước tiên ta sẽ được hỏi về các tuỳ chọn chung. Sau khi lựa chọn các tuỳ chọn chung, ta sẽ được hỏi về các tuỳ chọn cụ thể. Với các phiên bản khác nhau của hạt nhân thì các tuỳ chọn có thể khác nhau một chút song về cơ bản thì không khác nhau. Sau đây là các tuỳ chọn để hạt nhân hỗ trợ hoạt động mạng đối với hạt nhân phiên bản 1.1.14 và các phiên bản sau này.

```
*
* Network options - //Các tuỳ chọn cho mạng
*
TCP/IP networking (CONFIG_INET) [ y]
```

Để sử dụng mạng TCP/IP, ta phải chọn Y. Nếu chọn N thì ta sẽ không thể sử dụng mạng với giao thức TCP/IP được. Tuy vậy ta vẫn có thể sử dụng mạng bằng giao thức IPX.

```
IP forrwarding/gatewaying (CONFIG_IP_FORWARD) [ n]
```

Tuỳ chọn này trả lời là Y nếu máy tính hoạt động như một gateway giữa hai mạng, hoặc có hỗ trợ kết nối cho trạm khác bằng liên kết SLIP... Để thiết lập một máy trạm hoạt động như một bức tường lửa (firewall), ta nên chọn N. Bức tường lửa là một máy trạm kết nối giữa hai hay nhiều mạng nhưng không làm nhiệm cụ chọn đường cho các gói tin giữa chúng. Chúng thường được sử dụng để những người dùng trong một mạng riêng của một công ty hay một tổ chức truy cập vào Internet và đảm bảo an toàn thông tin cao nhất cho mạng nội bộ. Người dùng được cung cấp quyền truy cập Internet thông qua bức tường lửa nhưng không một trạm nào bên ngoài được quyền truy cập vào mạng nội bộ.

```
*
* it is safe to leave these untouche //Có thể bỏ qua tuỳ chọn
này
*
PC/TCP compatibility mode (CONFIG_INET_PCTCP) [ n]
```

Tuỳ chọn trên cho phép khắc phục một số điểm không tương thích giữa một vài phiên bản của PC/TCP, một phiên bản thương mại của TCP cho các máy tính cá nhân dùng hệ điều hành MS-DOS. Nếu chúng ta chọn Y thì máy tính vẫn có khả năng hoạt động cùng với các máy Linux khác song hiệu năng của hệ thống bị giảm đối với các liên kết tốc độ chậm.

```
Reverse ARP (CONFIG_INET_RARP) [ n]
```

Tuỳ chọn trên cho phép hỗ trợ giao thức ánh xạ địa chỉ ngược RARP - Reverse Address Resolution Protocol). RARP được sử dụng để các máy trạm không có ổ đĩa và các X terminal yêu cầu địa chỉ IP trong quá trình khởi động. Nếu mạng có những máy trạm như trên thì ta nên lựa chọn Y. Phiên bản mới nhất của các tiện ích mạng (net-032d) có chứa một tiện ích rarp cho phép thêm thông tin của các hệ thống vào trong cache của RARP.

```
Assume subnets are local (CONFIG_INET_SNARL) [ y]
```

Thông thường, một đơn vị dữ liệu của TCP sẽ được chia ra làm nhiều phần nhỏ hơn tại tầng IP (gói tin IP). Khi gửi cho các trạm trong mạng cục bộ thì kích thước gói tin IP là tương đối lớn. Đối với các trạm ở xa kích thước gói tin sẽ nhỏ hơn để tránh sự phân mảnh dữ liệu do một số liên kết ở xa có kích thước gói tin cực đại là nhỏ. Nếu chọn N đối với SNARL, hạt nhân chỉ xem một mạng con là cục bộ khi nó có liên kết trực tiếp với nó. Còn nếu chọn Y cho SNARL thì hạt nhân sẽ xem tất cả các mạng con là cục bộ và sử dụng gói tin có kích thước lớn khi truyền tin với các trạm trong mạng đó.

```
Disable NAGLE algorithm (normally enable) (CONFIG_TCP_NAGLE_OFF) [ n]
```

Thuật toán Nagle là một thuật toán heuristic được sử dụng để tránh việc phải gửi các gói tin IP có kích thước nhỏ (Tinygram). Tinygram được dùng trong các ứng dụng tương tác chỉ chuyển dữ liệu về một phím bấm như telnet hay rsh. Tinygram sẽ làm giảm hiệu suất hoạt động của mạng đối với các liên kết có băng thông nhỏ như SLIP. Thuật toán Nagle sẽ ngăn chặn việc gửi dữ liệu TCP trong một số tình huống nhất định. Nên chọn loại bỏ thuật toán trên khi gặp có khá nhiều gói tin bị mất.

```
The IPX protocol (CONFIG_IPX) [ n]
```

IPX là giao thức tầng giao vận được sử dụng trong mạng Novell Netware. Giao thức này vẫn đang được phát triển và sử dụng rộng rãi. Nó cho phép trao đổi dữ liệu với các tiện ích của DOS sử dụng IPX và truyền thông với các trạm trong mạng Novell thông qua liên kết PPP.

```
Dummy net driver support (CONFIG_DUMMY) [ y]
```

Bắt đầu từ phiên bản 1.1.16, Linux hỗ trợ một trình điều khiển đặc biệt, đó là dummy interface. Giao diện này mặc dù ít được sử dụng song nó rất có ích cho các máy trạm riêng lẻ hay kết nối với mạng qua liên kết SLIP. Nó được xây dựng dựa trên giao diện loopback. Khi một máy trạm có sử dụng liên kết SLIP nhưng không có bộ giao tiếp mạng Ethernet, ta cần sử dụng giao diện dummy để trạm luôn có một địa chỉ IP. Chi tiết về giao diện trên sẽ được trình bày kỹ hơn trong phần "Giao diện dummy" ở chương sau.

3. Một số giao diện mạng trong Linux

Hạt nhân hệ điều hành Linux hỗ trợ nhiều loại trình điều khiển cho các thiết bị phần cứng khác nhau. Trong phần này sẽ liệt kê một số trình điều khiển thường gặp và các giao diện tương ứng với chúng.

Có nhiều tên chuẩn cho các giao diện trong Linux. Hầu hết các trình điều khiển đều hỗ trợ nhiều giao diện, ví dụ *eth0, eth1...*

lo — Giao diện loopback. Giao diện này sử dụng cho các mục đích thử nhiệm. Nó hoạt động bằng cách gửi trả về tầng mạng bất kỳ một gói tin nào gửi qua nó. Trong hạt nhân luôn luôn có một trình điều khiển cho giao diện này.

ethn — Là giao diện cho card mạng Ethernet thứ n-1. Đây là tên chung cho tất cả các card mạng Ethernet.

dln — Giao diện cho bộ điều hợp D-Link DE-600, một dạng khác của thiết bị Ethernet. Sự khác nhau duy nhất là DE-600 được điều khiển thông qua cổng song song chứ không phải qua các khe cắm trên khe ISA hay PCI của máy tính.

sln — Giao diện SLIP, giao diện này được liên kết với một cổng nối tiếp, ví dụ *sl0, sl1...* hạt nhân Linux hỗ trợ tới 4 giao diện SLIP.

pppn Giao diện PPP. Giống như giao diện SLIP, một giao diện PPP được liên kết với một cổng nối tiếp khi cổng này chuyển sang chế độ PPP. Hiện tại có 4 giao diện PPP được hỗ trợ.

plipn Giao diện PLIP. Giao diện này thực hiện truyền các gói tin IP qua cổng song song. Hạt nhân Linux hỗ trợ 3 giao diện PLIP. Giao diện trên được cài đặt nhờ trình điều khiển PLIP khi khởi động và được ánh xạ vào một cổng song song.

4. Cài đặt các thiết bị mạng Ethernet

Hiện tại Linux hỗ trợ nhiều chủng loại bộ giao tiếp mạng Ethernet. Hầu hết các trình điều khiển cho chúng đều được viết bởi Donald Becker (**becker@cesdis.gsfc.nasa.gov**). Ông là tác giả của họ các trình điều khiển cho các bộ giao tiếp dựa trên National Semiconductor 8390 chip, chúng được biết đến với tên Becker Series Drivers. Ngoài ra còn có các trình điều khiển khác cho các sản phẩm của D-Link, trong đó bộ điều hợp D-Link dùng để truy cập Internet qua cổng song song. Các trình điều khiển này được viết bởi Bjorn Ekwall (**bj0rrn@bloxe.se**). Các trình điều khiển DEPCA được viết bởi David C. Davis (**davies@waton.lkg.dec.com**).

4.1. Các loại bộ giao tiếp mạng được hỗ trợ

Danh sách các phần cứng được Linux hỗ trợ được để tại **http://www.linux.org**. Một danh sách đầy đủ các bộ giao tiếp mạng được hỗ trợ được phát hành hàng tháng tại nhóm tin **comp.os.linux.announce** trong mục Ethernet HOWTO do Paul Gortmarker phát hành (**gpg109@rsphysse.anu.edu.au**).

Dưới đây là danh sách tóm lược các card mạng thông dụng được hỗ trợ bởi Linux. Danh sách đầy đủ phải tham khảo trong HOWTO. Ngay cả trong trường hợp card mạng được liệt kê ra ở đây, ta cũng nên kiểm tra lại HOWTO để biết một số chi tiết đặc biệt trong hoạt động của card. Trường hợp đặc biệt là một số vi mạch Ethernet có chế độ vào ra kiểu DMA (vào ra bằng cách truy cập trực tiếp bộ nhớ) sử dụng cùng kênh DMA với bộ điều khiển Adaptec 1542 SCSI. Nếu không thay đổi sang một kênh DMA khác, Ethernet sẽ ghi dữ liệu nhận được lên một vùng bất kỳ của ổ cứng, gây ra lỗi hệ thống.

3Com EtherLink

Các loại 3c503, 3c503/16, 3c507, 3c509, 3c59x đều được hỗ trợ. Loại 3c501 cũng được hỗ trợ, nhưng tốc độ của nó chậm khi hoạt động trong Linux.

Novell Eagle

NE1000, NE2000, và một số loại thuộc họ này. NE1500 và NE2100 cũng được hỗ trợ.

Western Digital/SMC

WD8003, WD8013 cũng như SMC Elite, SMC Elite Plus và SMC Elite 16 Ultra đều được hỗ trợ.

Hewlett Packard

HP 27252, HP 27247B, và HP J2405A.

D-Link

Bộ điều hợp DE-600, DE-100, DE-200, và DE-220-T. Cũng có một công cụ bổ xung cho DE-650-T (là một loại bộ giao tiếp PCMCIA).

DEC

DE200 (32K/64K), DE202, DE100, và DEPCA rev E.

Allied Teliesis

AT1500 và AT1700.

Để sử dụng một trong những thiết bị trên, ta có thể dùng các hạt nhân đã được dịch sẵn của các nhà cung cấp Linux. Tất cả các trình điều khiển cho các thiết bị trên đã được xây dựng sẵn. Tuy nhiên, khi đã quen với công việc biên dịch hạt nhân , ta nên xây dựng lại hạt nhân cùng với các trình điều khiển cần thiết để đảm bảo hiệu quả hoạt động của Linux.

4.2. Tự động phát hiện bộ giao tiếp mạng Ethernet

Khi khởi động, Linux sẽ cố gắng phát hiện loại card Ethernet trên máy và định vị cho nó (định vị có nghĩa là ánh xạ cho thiết bị ngoại vi một vùng nhớ vào ra I/O). Thứ tự dò tìm và địa chỉ vào ra tương ứng với các loại card như sau:

Bảng 3.3. Một số bộ giao tiếp mạng được hỗ trợ

Vỉ mạch	Địa chỉ dò tìm
WD/SMC	0x300, 0x280, 0x380, 0x240
SMC 16 Ultra	0x300, 0x280
3c501	0x280
3c503	0x300, 0x310, 0x330, 0x350, 0x250, 0x280, 0x2a0, 0x2e0
NEx000	0x300, 0x280, 0x320, 0x340, 0x360
HP	0x300, 0x320, 0x340, 0x280, 0x2C0, 0x200, 0x240
DEPCA	0x300, 0x320, 0x340, 0x360

Việc tự động dò tìm có hai nhược điểm. Thứ nhất là Linux có thể không nhận đúng các tham số đối với một số card mạng. Nhóm này bao gồm các thiết bị dạng "nhái", thậm chí các thiết bị loại "xịn" như WD80x3. Thứ hai là hạt nhân không tự động nhận ra được nhiều hơn hai bộ giao tiếp mạng. Đây là một đặc tính của Linux cho phép ta được quyền gán giao diện tương ứng cho bộ giao tiếp mạng.

Khi việc dò tìm card mạng không thành công, ta cần chỉ rõ cho hạt nhân biết tên của nó và địa chỉ cơ sở tương ứng. Ở trong Net-3, có hai cách khác nhau để thực hiện.

Cách thứ nhất là sửa đổi mã nguồn của hạt nhân và biên dịch lại nó. Tất cả các thông tin về trình điều khiển thiết bị mạng được đặt trong tệp *driver/net/Space.c*. Cách này đòi hỏi người thực hiện phải có kinh nghiệm với mã nguồn hạt nhân của hệ thống.

Cách thứ hai là sử dụng các tham số hạt nhân (kernel parameters). Đây là các thông tin đưa thêm cho hạt nhân lúc khởi động. Nếu sử dụng lilo để khởi động thì các tham số sẽ được đưa vào qua mục append trong tệp */etc/lilo.conf*. Để đưa cho hạt nhân các thông tin một thiết bị Ethernet, ta dùng các tham số sau:

```
ether=irq,base addr,param1,param2,name
```

Bốn tham số đầu tiên là dạng số. Tham số cuối cùng là tên thiết bị. Tất cả các tham số dạng số là tuỳ chọn. Nếu chúng bị bỏ qua thì hạt nhân sẽ tự động dò tìm, nếu chúng được đặt là 0 thì hạt nhân sẽ sử dụng giá trị mặc định. Ý nghĩa của chúng như sau:

irq

Số hiệu ngắt của thiết bị. Theo mặc định thì hạt nhân sẽ tự động dò tìm IRQ của thiết bị. Loại card 3c503 có tính chất đặc biệt là sẽ chọn tự do IRQ trong danh sách 5,9,3,4.

base_addr

Địa chỉ vào ra cơ sở của thiết bị. Nếu được đặt về 0 thì hạt nhân sẽ tự động dò tìm địa chỉ này.

param1, param2

Các tham số này được sử dụng khác nhau đối với các trình điều khiển khác nhau. Với thiết bị sử dụng bộ nhớ chung như WD80x3, đây là địa chỉ bắt đầu và kết thúc của vùng nhớ chung. Một số thiết bị khác sử dụng param1 để chỉ mức độ gỡ rối thông tin. Param1 = 8 thì bỏ chế độ gỡ rối. Param1 = 0 là chế độ mặc định. Các giá trị từ 1 đến 7 chỉ mức độ gỡ rối tăng dần. Loại card 3c503 dùng param2 để chọn bộ thu phát trong (mặc định) hay bộ thu phát ngoài (param = 1).

Nếu có 2 card mạng, ta có thể để cho Linux tự động phát hiện một trong chúng, đồng thời đưa các tham số của cái còn lại trong lilo. Tuy vậy phải chắc chắn rằng Linux phải phát hiện card thứ nhất trước!. Điều này được thực hiện bằng cách đưa tham số *reserve* trong lilo, tham số này chỉ dẫn hạt nhân không dò tìm không gian nhớ vào ra đã được người dùng giành cho card Ethernet thứ hai.

Ví dụ: Để Linux cài đặt một bộ giao tiếp mạng Ethernet thứ hai, cần thêm các tham số hạt nhân sau vào tệp */etc/lilo.conf*:

```
reserve=0x300,32 ether=0,0x300,eth1
```

Tuỳ chọn reserve đảm bảo không một trình điều khiển nào có thể truy cập vào không gian nhớ vào/ra khi dò tìm thiết bị. Ta cũng có thể sử dụng tham số hạt nhân để ghi đè lên cấu hình của eth0:

```
reserve=0x340,32 ether=0,0x340,eth0
```

Để tắt chế độ dò tìm đối với eth0, ta gán giá trị -1 cho địa chỉ cơ sở:

```
ether=0,-1,eth0
```

5. Trình điều khiển PLIP

PLIP - Parallel Line IP, là một giao thức đơn giản để có một mạng với 2 máy tính. Nó sử dụng liên kết mạng giữa hai máy tính qua cổng song song và cáp dữ liệu đặc biệt, cho phép đạt tốc độ truyền từ 10kB*ps* đến 20kB*ps*.

PLIP được phát triển bởi Crynwr. Thiết kế PLIP của ông mang nhiều tính "thủ thuật" hơn là kỹ thuật: thường cổng song song được thiết kế với 8 đầu ra dữ liệu để gửi dữ liệu ra thiết bị ngoại vi mà không có chiều ngược lại. PLIP đã sử dụng 5 đầu vào trạng thái để truyền dữ liệu theo chiều ngược lại, và thực hiện truyền dữ liệu với 4 bit (1/2 byte) vào một thời điểm. Chế độ hoạt động như trên là chế độ 0. Ngày nay người ta đã mở rộng PLIP sang chế độ 1, tức là thực hiện truyền dữ liệu hai chiều với giao diện đầy đủ 8 bit.

Hiện nay Linux có hỗ trợ giao thức PLIP ở chế độ 0. Để kết nối hai máy tính dùng PLIP ta cần có cáp nối chúng thông qua cổng song song. Các loại cáp thường dùng là "Null Printer" hay "Turbo Laplink". Ta cũng có thể dễ dàng tự làm lấy một cái cáp như vậy với dụng cụ là một mỏ hàn, vật liệu là dây dẫn bằng đồng và 2 đầu cắm.

Trình điều khiển của PLIP được thực hiện bởi rất nhiều tác giả. Phiên bản hiện nay thường dùng là của Niibe Yutaka. Khi biên dịch vào trong hạt nhân, nó sẽ thiết lập một giao diện mạng ứng với mỗi cổng máy in trên máy, ví dụ giao diện *plip1* cho cổng *lp1*, *plip2* cho cổng *lp2*,... Ánh xạ giao diện tới cổng như sau:

Bảng 3.4. Các giao diện plip

Giao diện	Cổng vào/ra	IRQ
Plip0	0x3BC	7
Plip1	0x378	7
Plip2	0x278	5

Nếu muốn có một cấu hình khác cho cổng máy in thì ta có thể thay đổi các giá trị này trong tệp mã nguồn */drivers/net/Space.c* và biên dịch lại hạt nhân.

Mặc dù đã thiết lập giao diện PLIP cho các cổng song song nhưng ta vẫn được phép sử dụng các cổng này bình thường cho các công việc khác. Chúng chỉ được trưng dụng cho giao diện PLIP khi ta thiết lập cờ UP cho giao diện này.

6. Thiết lập cấu hình cho cổng nối tiếp

Các giao thức SLIP (Serial Line IP) và PPP (Point-to-Point Protocol) được sử dụng rộng rãi để truyền các gói tin IP qua cổng nối tiếp. Rất nhiều các nhà cung cấp dịch vụ cho phép thực hiện quay số qua modem và sử dụng các giao thức trên để truy cập dịch vụ Internet và liên kết IP cho mỗi cá nhân. Để sử dụng SLIP hoặc PPP trước tiên ta cần cấu hình cho các cổng nối tiếp.

6.1. Các chương trình dùng cho Modem

Có rất nhiều chương trình đầu cuối (terminal) để phục vụ việc kết nối bằng cách quay số qua modem trong Linux. Một chương trình được sử dụng truyền thống là *kermit*. Hiện nay có nhiều chương trình cung cấp nhiều tính năng hơn như hỗ trợ danh bạ điện thoại, hỗ trợ các script để quay số và truy cập vào trạm ở xa..., *minicom* là một chương trình như vậy. Ngoài ra còn có các chương trình với giao diện đồ hoạ trong môi trường X-Window như *seyon*.

Ngoài các chương trình đầu cuối như trên còn có các phần mềm sử dụng cổng nối tiếp để truyền dữ liệu giữa các trạm. Một trong những phần mềm như vậy là các chương trình truyền thông sử dụng giao thức UUCP. Chúng được sử dụng nhiều trong các ứng dụng truyền tệp, thực hiện gọi thủ tục ở xa (RPC), trao đổi thư điện tử trong mạng nội bộ.

SLIP là giao thức liên mạng sử dụng đường truyền qua cổng nối tiếp. Giao thức này cho phép hoạt động ở cả chế độ tương tác và không tương tác. Có thể dùng SLIP để thực hiện quay số vào mạng ở xa hoặc dùng trong các dịch vụ truyền tệp FTP. Ngoài ra nó cũng được sử dụng để thực hiện một kết nối cố định hoặc bán cố định giữa hai mạng LAN, sử dụng trong mạng ISDN...

6.2. Các thiết bị truyền tin nối tiếp

Các chương trình mà Linux dùng để quản lý thiết bị nối tiếp có tên là *ttys*. Đây là chữ viết tắt của Teletype™, một trong những nhà sản xuất thiết bị đầu cuối nổi tiếng thời bấy giờ. Ngày nay tên này được dùng cho tất cả các thiết bị đầu cuối thực hiện trao đổi dữ liệu kiểu hướng ký tự.

Trong Linux có 3 loại thiết bị ttys khác nhau: màn hình ảo (virtual console), thiết bị đầu cuối giả (*p*seudo-terminal), hoạt động gần giống như một ống dẫn hai chiều (two way pipe) và thiết bị nối tiếp. Các thiết bị nối tiếp cũng được xem như là ttys, bởi vì chúng cho phép thực hiện các phiên làm việc tương tác thông qua

một liên kết với một thiết bị đầu cuối hoặc với một trạm khác ở xa qua đường điện thoại.

Các tham số quan trọng của một liên kết nối tiếp là tốc độ đường truyền và bit chẵn lẻ (parity). Khi nhắc tới tốc độ đường truyền cần quan tâm tới các khái niệm Bit rate và Baud rate. Bit rate là tốc độ đường truyền tín hiệu được tính bằng đơn vị bit/s (b*ps*). Baud rate thực chất là tốc độ thay đổi tín hiệu tại nguồn phát tín hiệu- ở đây là thiết bị nối tiếp. Thực chất Bit rate cho ta biết một cách trực quan hơn tốc độ truyền dữ liệu giữa hai điểm của liên kết nối tiếp. Các giá trị của Baud rate và Bit rate là khác nhau mặc dù chúng đều được dùng để chỉ thông lượng trên đường truyền tín hiệu.

6.3. Truy cập vào các thiết bị nối tiếp

Cũng giống như tất cả các hệ thống Unix khác, các thiết bị nối tiếp được truy cập thông qua tệp thiết bị trong thư mục */dev*. Có hai dạng tệp thiết bị tương ứng với một cổng nối tiếp, tuỳ thuộc vào tệp thiết bị mà hoạt động của cổng sẽ được điều khiển tương ứng.

Dạng tệp thứ nhất được sử dụng khi cổng nối tiếp phục vụ cho việc quay số vào (dial-in). Tên của chúng là ttyS0, ttyS1, ..., Dạng tệp thứ hai được sử dụng khi cổng nối tiếp phục vụ cho việc quay số ra (dial-out) với các tên cua0, cua1,...

Như chúng ta đã biết, các tệp thiết bị trong */dev* có hai tham số quan trọng là số hiệu chính (major number) và số hiệu phụ (minor number). Số hiệu chính đặc trưng cho dạng tệp thiết bị, còn số hiệu phụ để phân biệt các tệp khác nhau của cùng một dạng thiết bị đó. Số hiệu chính của */dev/cua* là 5, còn số hiệu chính của */dev/ttyS* là 4. Số hiệu phụ của hai dạng tệp trên là như nhau và bắt đầu từ giá trị 64:

```
$ ls -l /dev/cua*
/dev/cua0      crw-rw-rw-  1 root  root  5, 64 Nov 30 19:31
/dev/cua1      crw-rw-rw-  1 root  root  5, 65 Nov 30 19:31
/dev/cua2      crw-rw-rw-  1 root  root  5, 66 Nov 30 19:31
/dev/cua3     crw-rw-rw-  1 root  root  5, 67 Nov 30 19:31

$ ls -l /dev/ttyS*
/dev/ttyS0     crw-rw-rw-  1 root  root  4, 64 Nov 30 19:31
```

```
/dev/ttyS1     crw-rw-rw-  1 root  root  4, 65 Nov 30 19:31
/dev/ttyS2     crw-rw-rw-  1 root  root  4, 66 Nov 30 19:31
/dev/ttyS3   crw-rw-rw-  1 root  root  4, 67 Nov 30 19:31
```

Nếu chưa có tệp thiết bị nào như trên thì ta có thể tạo ra một tệp như vậy. Khi đăng nhập với tư cách là root:

```
# mknod -m 666 /dev/cual c 5 65
# chown root.root /dev/cual
```

Một số ứng dụng có sử dụng tệp liên kết */dev/modem* tới */dev/cua1* để tăng thêm tính trực quan. Tuy nhiên, ta không thể sử dụng cả hai tệp trong hai chương trình khác nhau cùng một lúc được do thực chất chúng đều chỉ đến một thiết bị. Khi một chương trình sử dụng tệp thiết bị, nó sẽ sử dụng một tệp khoá (lock file) để báo cho các chương trình khác biết rằng thiết bị đang được dùng. Khi đó nếu hai chương trình cùng sử dụng một thiết bị sẽ xảy ra tình trạng đọc tệp khoá của nhau, làm cho không chương trình nào hoạt động được.

III. THIẾT LẬP CẤU HÌNH MẠNG TCP/IP

Trong phần này ta sẽ từng bước thiết lập cấu hình cho mạng các máy tính Linux sử dụng giao thức TCP/IP. Các vấn đề bao gồm gán địa chỉ IP, cấu hình hoạt động cho giao thức TCP/IP và cuối cùng là một số công cụ cần thiết được sử dụng để giải quyết các vấn đề gặp phải trong quá trình cài đặt.

Hầu hết các công việc làm trong phần này chỉ phải làm một lần trong quá trình thiết lập cấu hình mạng. Chúng chỉ được làm lại khi thêm một trạm mới vào mạng hoặc cấu hình mạng lại từ đầu. Một số lệnh dùng để cấu hình giao thức TCP/IP phải được thực hiện khi hệ thống khởi động bằng cách đặt chúng vào trong các scripts khởi động của hệ thống trong thư mục */etc/rc*.

Các scripts liên quan đến cấu hình mạng thường là rc.net hoặc rc.inet, một số trường hợp có thể có thêm *rc.net1* và *rc.net2*. Trong đó *rc.net1* chứa các lệnh khởi tạo hạt nhân hệ thống để làm việc với mạng máy tính, *rc.net2* khởi tạo các dịch vụ và một số ứng dụng cơ bản trên mạng. Trong phần này sẽ đề cập đến các

công việc liên quan đến *rc.net1*. Các dịch vụ và ứng dụng sẽ được đề cập trong chương sau.

1. Thiết lập hệ thống tệp proc

Một số công cụ trong việc thiết lập cấu hình mạng dựa vào *proc* để giao tiếp với hạt nhân hệ điều hành. *Proc* là một giao diện cho phép truy cập vào hạt nhân trong thời gian chạy theo cơ chế của một hệ thống tệp. Khi được tạo dựng, các tệp của nó có thể được liệt kê hoặc xem nội dung như mọi hệ thống tệp bình thường khác. Ví dụ tệp *loadavg* cho biết độ nạp trung bình của hệ thống, *meminfo* cho biết thông tin về bộ nhớ trong và bộ nhớ hoán đổi.

Các đoạn mã liên quan tới mạng được để trong thư mục net. Thư mục này chứa các tệp tin liên quan đến bảng ARP của hạt nhân, trạng thái của các liên kết TCP/IP và thông tin về bảng chọn đường. Hầu hết các công cụ quản trị mạng đều lấy thông tin từ các tệp tin này.

Hệ thống tệp proc được tạo dựng trên thư mục */proc* trong quá trình khởi động. Cách tốt nhất để tạo dựng tự động một hệ thống tệp và thêm nó vào trong tệp */etc/fstab* và cho thực hiện lệnh *"mount proc"* trong script khởi động */etc/rc*

```
#procfs mount point:
none    /proc proc  defaults
```

Ngày nay hệ thống tệp proc được hạt nhân của hệ điều hành Linux hỗ trợ tự động. Quá trình tạo dựng được thực hiện tự động mà không cần có sự can thiệp của người quản trị hệ thống. Tuy nhiên trong trường hợp hạt nhân không tự động hỗ trợ, thông báo lỗi sau sẽ xuất hiện:

```
"mount: fs type procfs not supported by kernel"
```

Khi đó cần phải dịch lại hạt nhân hệ điều hành với tuỳ chọn hỗ trợ hệ thống tệp proc.

2. Cài đặt các tệp nhị phân

Linux là một hệ điều hành có mã nguồn mở, chính vì vậy mà ta có thể nhận được các ứng dụng trên Linux ở dạng mã nguồn (source), mã nhị phân hay mã nguồn đã được biên dịch (binaries) hoặc ứng dụng đóng gói hoàn chỉnh (Package).

Với các phiên bản được cung cấp dưới dạng đóng gói hoàn chỉnh, nó đã chứa hầu như đầy đủ các ứng dụng, tiện ích và các tệp tin phục vụ cho mạng máy tính.

Khi cần cài đặt một phiên bản mới của hạt nhân và các tiện ích liên quan, do các phần mềm đóng gói chưa thay đổi theo tương ứng nên ta cần phải cài các tiện ích đó từ mã nguồn hoặc các tệp nhị phân mới nhất. Chúng thường được cung cấp kèm theo hạt nhân dưới dạng *net-XXX.tar.gz* trong đó *XXX* là số hiệu phiên bản.

Nếu muốn biên dịch và cài đặt các ứng dụng TCP/IP chuẩn cho mạng máy tính, có thể lấy mã nguồn từ các địa chỉ truyền tệp của Linux. Địa chỉ FTP chính thức của Net-3 là **sunacm.swan.ac.uk,** địa chỉ mirror tương ứng là **sunsite.unc.edu** ở trong thư mục **system/Network/sunacm**. Phiên bản mới nhất của Net-2e có thể tải về từ **ftp.aris.com**. Mã chương trình và ứng dụng mạng của Mathias Urlichs' BSD có thể lấy từ địa chỉ **ftp.ira.ka.de** trong thư mục **/pub/system/linux/netbsd**.

3. Thiết lập tên máy trạm

Tất cả các ứng dụng trên mạng tại một trạm đều có sử dụng tên của trạm đó. Tên máy trạm được thiết lập trong quá trình khởi động bằng lệnh:

```
$ hostname      name
```

Một máy trạm thường sử dụng tên mà không có tên miền tương ứng với nó. Ví dụ, một máy trạm tại miền **vbrew.com** của mạng Virtual Brewery có tên định danh đầy đủ là **vale.vbrew.com** thì tên cục bộ thường sử dụng là vale. Tuy nhiên, để đảm bảo có thể thực hiện tìm kiếm địa chỉ IP cho trạm này thì tên cục bộ đó phải được đưa vào trong tệp tin */etc/hosts* (Sẽ đề cập dưới đây).

Một cách để thêm phần tên miền vào tên trạm để có được định danh đầy đủ FQDN là sử dụng lệnh domainname. Tuy nhiên, trong trường hợp máy trạm có sử dụng dịch vụ thông tin mạng NIS thì lệnh *domainname* được dùng để thiết lập tên cho miền NIS (sẽ giải thích sau), do đó xung đột có thể xảy ra vì tên miền NIS và tên miền DNS có thể khác nhau. Chính vì vậy, cách này không nên sử dụng khi dùng đồng thời hai dịch vụ trên.

4. Gán địa chỉ IP

Nếu chỉ muốn thiết lập cấu hình máy tính để chạy các ứng dụng mạng trên một máy duy nhất thì ta có thể bỏ qua phần này. Máy tính chỉ cần có giao diện loopback với địa chỉ IP 127.0.0.1 là đủ.

Trong trường hợp máy tính làm việc trong môi trường mạng thực sự thì nó phải được cấp một địa chỉ IP. Khi muốn kết nối máy tính vào mạng có sẵn, phải đề

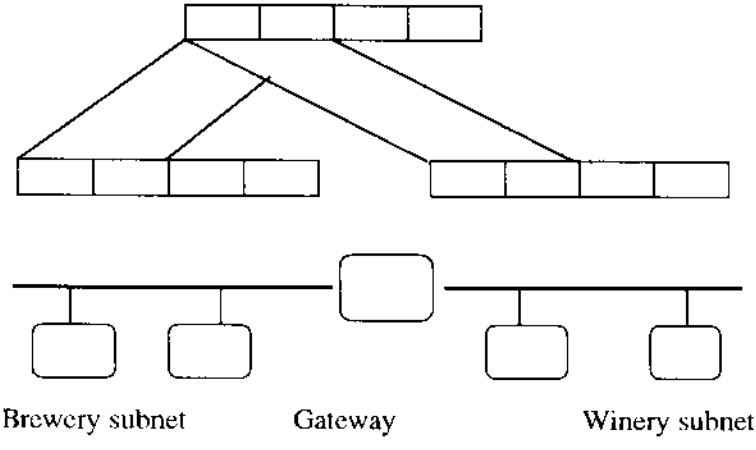
nghị người quản trị mạng cấp cho một địa chỉ IP. Trường hợp tự thiết lập một mạng riêng thì quá trình gán địa chỉ IP được tiến hành theo các bước mô tả dưới đây.

Các trạm trong một mạng cục bộ phải có cùng phần địa chỉ mạng. Do đó trước tiên cần phải chọn cho mạng một địa chỉ IP. Nếu mạng được chia làm các mạng con thì phải dùng kỹ thuật chia mạng (subnetting) để phân chia địa chỉ IP cho từng mạng con tương ứng.

Với những mạng không kết nối vào mạng Internet, việc chọn địa chỉ IP là tuỳ ý. Tuy vậy mạng phải thuộc một trong 3 lớp A, B, C vì nếu không mạng sẽ không hoạt động đúng. Nếu có ý định kết nối vào Internet thì mạng phải được cung cấp địa chỉ IP một cách chính thức, cách tốt nhất là đề nghị các nhà cung cấp dịch vụ ISP giúp đỡ.

Việc chia mạng chỉ phải tiến hành khi chúng ta có nhiều hơn một mạng quảng bá. Với nhiều mạng điểm -điểm thì không nhất thiết phải chia mạng. Ví dụ, nếu chỉ có một mạng LAN Ethernet và có một hay nhiều liên kết SLIP, việc chia mạng là không cần thiết.

Ví dụ: Nhà quản trị mạng Brewery đề nghị trung tâm NIC cho một mạng loại B và được cấp địa chỉ mạng 191.72.0.0. Để chia thành hai mạng con, 8 bits của phần địa chỉ trạm được thêm vào cho các mạng con. Mạng con thứ nhất được gán địa chỉ mạng 1, mạng kia là 2. Như vậy mỗi mạng con có thể có tối đa 256 máy trạm và địa chỉ mạng của chúng là 190.72.1.0 và 190.72.2.0 với mặt nạ tương ứng là 255.255.255.0.



Hình 3.5. Chia mạng thành 2 mạng con

Để kết nối hai mạng trên ta sử dụng một trạm làm gateway. Trạm nàyđược gán hai địa chỉ IP 192.72.1.1 và 192.72.2.1 tương ứng với các giao diện kết nối vào mỗi mạng con (hình 3.5).

5. Các tệp cấu hình hosts và networks

Sau khi đã chia mạng lớn thành các mạng con, cần phải chuẩn bị một số ánh xạ địa chỉ đơn giản trong tệp cấu hình */etc/hosts*. Nếu mạng không sử dụng hệ thống tên miền DNS hoặc hệ thông tin mạng NIS thì tất cả các ánh xạ địa chỉ phải được để trong tệp trên.

Ngay cả trong trường hợp có sử dụng hai dịch vụ nói trên thì một số ánh xạ địa chỉ cũng nên đưa vào tệp trên. Thông tin này cho phép sử dụng tên máy trạm trong các scripts khởi động rc.inet thay vì địa chỉ IP. Khi sửa đổi địa chỉ IP của một trạm nào đó hoặc sử dụng dịch vụ DHCP (cấp phát địa chỉ IP động), không cần phải sửa lại toàn bộ các scripts trên ở tất cả các máy trạm mà chỉ cần sửa đổi phần địa chỉ IP trong tệp */etc/hosts* và sao chép sang tất cả các máy trạm.

Trong quá trình thử nghiệm, nên đảm bảo việc ánh xạ địa chỉ được thực hiện thông qua tệp */etc/hosts*. Một số phần mềm DNS hay NIS ban đầu chưa hoạt động ổn định có thể gây sự cố trong mạng. Để các ứng dụng chỉ tham chiếu đến */etc/hosts* ta sửa đổi tệp */etc/hosts.conf* bằng cách thêm dòng sau:

```
order   host
```

Tệp */etc/hosts* có cấu trúc như sau: Mỗi dòng là một ánh xạ địa chỉ, bao gồm địa chỉ IP, tên cục bộ , định danh tên miền đầy đủ. Mỗi phần cách nhau bởi dấu trắng hoặc tab, để bỏ qua một dòng , hãy dùng ký tự #. Sau đây là ví dụ về */etc/hosts* của một máy trạm trong mạng Brewery, trong đó trạm vlager được dùng với hai tên logic khác là vlager-if1 và vlager-if2:

```
#
# Tệp Hosts của Virtual Brewery/Virtual Winery
#
# IP            local        fully qualified domain name
#
127.0.0.1       localhost
#
191.72.1.1      vlager       vlager.vbrew.com
```

```
191.72.1.1      vlager-if1    vstout.vbrew.com
191.72.1.3      vale          vale.vbrew.com
#
191.72.2.1      vlager-if2
191.72.2.2      vbeaujolais   vbeaujolais.vbrew.com
191.72.2.3      vbardolino    vbardolino.vbrew.com
191.72.2.4      vchianti      vchianti.vbrew.com
```

Tương tự như địa chỉ IP của máy trạm, ta có thể sử dụng tên biểu tượng cho địa chỉ mạng. Tệp cấu hình */etc/networks* cho phép thực hiện chuyển đổi tên biểu tượng sang địa chỉ mạng và ngược lại. Tệp này ở mạng Virtual Brewery như sau:

```
# Tệp tin /etc/networks của mạng Virtual Brewery
brew-net        191.72.1.0
wine-net        191.72.2.0
```

6. Cấu hình giao diện cho IP

Sau khi đã thiết lập cấu hình phần cứng, cần phải chỉ ra cho các thiết bị này biết được phần mềm mạng nào của hạt nhân được sử dụng. Các lệnh được sử dụng để lập cấu hình giao diện mạng và khởi tạo bảng chọn đường là *ifconfig* và *route*. Công việc trên thường được thực hiện từ *rc.inet1* khi khởi động hệ thống.

Ifconfig được sử dụng để tạo một giao diện truy cập được tới của tầng mạng. Lệnh này thực hiện gán địa chỉ IP và các tham số khác cho tầng mạng, kích hoạt giao diện (activate). Một giao diện được kích hoạt có nghĩa là nó đã sẵn sàng cho việc gửi và nhận các gói tin. Lệnh này có cú pháp đơn giản như sau:

```
$ ifconfig    interface    ip-address
```

Lệnh trên thực hiện gán địa chỉ *ip-address* cho giao diện *interface* đồng thời kích hoạt nó, các tham số khác sẽ được gán mặc định. Chẳng hạn mặt nạ mạng được gán cho mạng loại B 255.255.0.0. Lệnh *ifconfig* sẽ được đề cập kỹ hơn trong phần sau.

Route cho phép thêm hoặc bớt thành phần trong bảng chọn đường, sử dụng như sau (trong đó *add* hay *del* cho biết thêm vào hay xoá đi một thành phần trong bảng chọn đường):

```
$ route [ add | del]  target
```

6.1. Giao diện Loopback

Giao diện đầu tiên được khởi tạo trên hầu hết các máy trạm là giao diện loopback:

```
$ ifconfig lo 127.0.0.1
```

Hoặc

```
$ ifconfig lo localhost
```

Thông thường ta sử dụng tên máy cục bộ **localhost** thay cho 127.0.0.1. Khi đó trong tệp */etc/hosts* phải có dòng ánh xạ địa chỉ như sau:

```
# Sample /etc/hosts entry for localhost
localhost      127.0.0.1
```

Để kiểm tra cấu hình hiện tại của một giao diện, sử dụng lệnh ifconfig và tên của giao diện:

```
$ ifconfig lo
lo    Link encap Local Loopback
      inet addr 127.0.0.1 Bcast [ NONE SET] Mask 255.0.0.0
      UP BROADCAST LOOPBACK RUNNING  MTU 2000  Metric 1
      RX packets 0 errors 0 dropped 0 overrun 0
      TX packets 0 errors 0 dropped 0 overrun 0
```

Giao diện loopback được gán mặt nạ 255.0.0.0 vì 127.0.0.1 thuộc lớp địa chỉ loại A. Theo mặc định giao diện loopback không sử dụng thiết lập địa chỉ quảng bá vì điều đó không cần thiết cho giao diện này. Tuy nhiên, khi sử dụng chương trình *rwhod* trên máy cần phải thiết lập địa chỉ quảng bá cho giao diện loopback để *rwhod* có thể hoạt động đúng. Việc thiết lập địa chỉ quảng bá sẽ được đề cập trong phần ifconfig.

Sau khi thêm vào giao diện loopback, cần thêm địa chỉ này vào trong bảng chọn đường. Điều này cho phép IP biết có thể sử dụng giao diện này khi muốn gửi các gói tin đến địa chỉ 127.0.0.1:

```
$ route add 127.0.0.1
```

Tiếp theo, kiểm tra xem giao diện này hoạt động tốt hay chưa bằng lệnh *ping*. Lệnh này cho phép kiểm tra xem một địa chỉ cho trước có thể truy cập tới được hay không, đồng thời cho phép đo thời gian đi và về (Round Trip Time) của một gói tin tới trạm đó.

```
$ ping localhost
PING localhost (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp seq=0 ttl=32 time=1 ms
64 bytes from 127.0.0.1: icmp seq=1 ttl=32 time=0 ms
64 bytes from 127.0.0.1: icmp seq=2 ttl=32 time=0 ms
^C

--- localhost ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0/0/1 ms
```

Lệnh *ping* sẽ gửi liên tục các gói tin để kiểm tra địa chỉ đích cho đến khi ta gõ tổ hợp phím Ctrl-C. Ví dụ trên cho thấy gói tin tới giao diện loopback sẽ được trả về ngay lập tức (RTT = 0ms), chứng tỏ giao diện này đã hoạt động tốt.

Trong trường hợp lệnh ping không hoạt động tốt như đã chỉ ra ở trên, có thể có một số lỗi xảy ra. Trước tiên kiểm tra xem hạt nhân đã được cài đặt có hỗ trợ mạng hay không (kiểm tra sự tồn tại của */proc/net*). Nếu nhận được thông báo "Network is unreachable", có thể bảng chọn đường bị sai, hãy kiểm tra rằng địa chỉ loopback trong *route* và chỉ ra trong *ifconfig* là như nhau. Lỗi còn có thể xảy ra khi lệnh *route* và *ifconfig* không tương thích với phiên bản của hạt nhân hệ điều hành.

Các bước mô tả ở trên là đủ để có thể sử dụng các ứng dụng mạng trên máy tính riêng lẻ. Sau khi thêm các lệnh trên vào *rc.net1* và bảo đảm scripts này được thực hiện từ */etc/rc*, hãy khởi động lại máy tính. Khi đó ta có thể thử các ứng dụng mạng khác nhau trên máy này, chẳng hạn *"telnet localhost"*

Giao diện loopback không chỉ có tác dụng khi thử các hoạt động của mạng máy tính hay thử nghiệm các ứng dụng đang xây dựng, mà còn được sử dụng bởi một số ứng dụng trên mạng thực tế trong quá trình hoạt động bình thường. Do vậy, nên luôn luôn cấu hình giao diện này cho dù máy tính có được nối mạng hay không.

6.2. Giao diện Ethernet

Thiết lập giao diện Ethernet được thực hiện giống như đối với giao diện loopback, chỉ một số tham số khác được yêu cầu khi có sử dụng mạng con.

Hãy lấy ví dụ trong trường hợp mạng Virtual Brewery. Đây là một mạng loại

B được chia ra làm 2 mạng loại C. Để thêm các giao diện eth0 cho trạm vstout trong mạng, thực hiện lệnh sau:

```
$ ifconfig eth0 vstout netmask 255.255.255.0
```

Lệnh trên sẽ gán cho giao diện *eth0* địa chỉ IP của vstout (191.72.1.2). Nếu không thêm vào tham số *netmask* thì *ifconfig* sẽ tự động thêm tham số mặc định cho mặt nạ là 255.255.0.0. Để kiểm tra cần thực hiện lệnh:

```
$ ifconfig eth0
eth0 Link encap 10Mps Ethernet HWaddr  00:00:C0:90:B3:42
     inet addr 191.72.1.2 Bcast 191.72.1.255 Mask 255.255.255.0
     UP BROADCAST RUNNING  MTU 1500  Metric 1
     RX packets 0 errors 0 dropped 0 overrun 0
     TX packets 0 errors 0 dropped 0 overrun 0
```

Lệnh *ifconfig* sẽ tự động thêm vào địa chỉ quảng bá (trường Bcast = 191.72.1.255) và thiết lập giá trị MTU - Message Transfer Unit, là giá trị lớn nhất của đơn vị dữ liệu tầng vật lý mà giao diện có thể tạo ra. Tất cả các giá trị này có thể được sửa đổi bằng lệnh ifconfig.

Giống như giao diện loopback, cần sửa bảng chọn đường để hạt nhân biết có thể truy cập mạng qua giao diện eth0. Tiếp tục ví dụ trên ta có:

```
$ route add -net 191.72.1.0
```

Cơ chế để hạt nhân biết được phải đưa gói tin qua giao diện nào như sau: Hạt nhân kiểm tra tất cả các giao diện đã được thiết lập, thực hiện so sánh phần mạng của địa chỉ đích với phần mạng của địa chỉ giao diện (bằng cách sử dụng mặt nạ ứng với giao diện đó). Nếu trùng nhau thì gửi gói tin qua giao diện đó. Trên trạm vtous, khi địa chỉ đích là 191.72.1.10 thì gói tin đó sẽ được gửi qua giao diện eth0.

Tham số *-net* cho biết 191.72.1.0 là địa chỉ đích của một mạng. Giao diện có thể dẫn đường cho cả mạng hay cho một trạm riêng lẻ. Nếu địa chỉ IP đích có phần địa chỉ trạm (host part) bằng 0 thì đích là cả một mạng, trái lại đích chỉ là một máy trạm. Với địa chỉ 192.72.1.0, hạt nhân sẽ nhận ra đây là một trạm do không biết việc chia mang con. Đó là lý do để phải dùng tham số -net.

Ta cũng có thể sử dụng tên biểu tượng của mạng trong tệp */etc/networks* thay vì phải nhập giá trị số của địa chỉ IP. Khi đó không cần chỉ ra tham số -net trong lệnh route:

```
$ route add brew-net
```

Như vậy, ta đã thực hiện xong các thao tác lập cấu hình cơ bản cho giao diện *eth0*. Để kiểm tra giao diện hoạt động tốt hay không ta sử dụng lệnh ping tới một máy trạm trong mạng, ví dụ:

```
$ ping vlager
PING vlager: 64 byte packets
64 bytes from 191.72.1.1: icmp seq=0. time=11. ms
64 bytes from 191.72.1.1: icmp seq=1. time=7. ms
64 bytes from 191.72.1.1: icmp seq=2. time=12. ms
64 bytes from 191.72.1.1: icmp seq=3. time=3. ms
^C

----vstout.vbrew.com PING Statistics----
4 packets transmitted, 4 packets received, 0% packet loss
round-trip (ms) min/avg/max = 3/8/12
```

Nếu giao diện *eth0* không hoạt động đúng thì có nhiều khả năng xảy ra. Nếu chỉ số gói tin mất (package loss) khác 0 thì có thể do lỗi đường truyền. Nếu không nhận được bất kỳ gói tin nào, nên thực hiện kiểm tra giao diện với lệnh *netstat*. Thống kê các gói tin của *ifconfig* cho biết có gói tin nào được gửi qua giao diện này hay không. Nếu có quyền truy cập vào trạm ở xa, hãy kiểm tra các thống kê về giao diện trên trạm đó, bằng cách này ta sẽ xác định được chính xác nơi các gói tin bị mất. Thêm nữa hãy hiển thị các thông tin về bảng chọn đường để kiểm tra lại nó:

```
$ route -n
Kernel routing table
Destination     Gateway         Genmask         Flags Metric Ref Use
127.0.0.1       *               255.255.255.255 UH    1      0
191.72.1.0      *               255.255.255.0   U     1      0
```

Chi tiết về các trường trong bảng hiển thị ở trên sẽ được giải thích cặn kẽ trong phần Netstat. Trường Flag cho biết một số cờ được thiết lập cho giao diện, cờ U cho biết giao diện đã được kích hoạt, cờ H cho biết đích là một máy trạm. Trường Used cho biết số lần sử dụng giao diện này của hạt nhân.

6.3. Chọn đường qua gateway

Trong phần trước, chúng ta mới thiết lập cấu hình cho trạm làm việc trong mạng cục bộ riêng lẻ. Khi thực hiện kết nối với một mạng khác hoặc nối với mạng toàn cầu Internet, thường phải sử dụng gateway. Để sử dụng các dịch vụ do gateway cung cấp, cần có thêm một số thông tin chọn đường cho tầng mạng của hệ thống.

Tiếp tục ví dụ trên khi mạng Virtual Brewery và mạng Virtual Winery liên kết với nhau qua gateway tên là **vlager**. Giả sử **vlager** được cấu hình để hoạt động tốt trong hai mạng. Ta chỉ cần thêm các thông tin chọn đường cho một trạm của Brewery là vstout để nó có thể truy cập được vào mạng Winery thông qua **vlager**. Điều này được thực hiện như sau:

```
$ route add wine-net gw vlager
```

Đồng thời các máy trạm trên mạng Winery cũng phải chỉ rõ **vlager** là gateway như trên, nếu không sẽ chỉ có thể gửi gói tin từ **vstout** sang mạng Winery mà không nhận được các gói tin trả lời.

Ví dụ trên chỉ mô tả quá trình lập cấu hình khi hai mạng LAN được nối thông qua một gateway. Giả sử **vlager** đã kết nối với mạng Internet (ví dụ như qua một kết nối SLIP). Nếu ta muốn bất cứ gói tin nào có đích không thuộc mạng LAN đều được gửi qua **vlager** để đi ra ngoài thì ra làm như sau:

```
$ route add default gw vlager
```

Tên mạng *default* là tên biểu tượng của 0.0.0.0, có ý nghĩa là chọn đường mặc định. *Default* không cần chỉ ra trong */etc/networks* vì nó được xây dựng sẵn cho lệnh *route*.

Khi sử dụng lệnh ping với các trạm bên ngoài mạng mà chỉ số các gói tin bị mất là khá lớn chứng tỏ mạng đang bị tắc nghẽn tại gateway. Thực tế các gói tin bị mất do lỗi đường truyền vật lý ít hơn rất nhiều so với các gói tin bị mất do tình trạng tắc nghẽn trên mạng.

6.4. Thiết lập cấu hình cho gateway

Việc lập cấu hình cho một máy tính đóng vai trò một chuyển mạch giữa hai mạng Ethernet được thực hiện dễ dàng. Giả sử cần cấu hình cho vlager với 2 bộ giao tiếp mạng Ethernet, mỗi bộ nối với một mạng. Những gì cần làm là thiết lập 2 giao diện tương ứng với hai bộ giao tiếp mạng trên một cách độc lập, gán cho mỗi

giao diện một địa chỉ IP tương ứng.

Trước tiên thêm vào */etc/hosts* các thông tin sau:

```
191.72.1.1  vlager      vlager.vbrew.com
191.72.1.1  vlager-if1
191.72.1.1  vlager-if2
```

Sau đó thiết lập hai giao diện trên bằng các lệnh sau đây:

```
$ ifconfig eth0 vlager-if1
$ ifconfig eth1 vlager-if2
$ route add brew-net
$ route add wine-net
```

6.5. Giao diện PLIP

Liên kết PLIP dùng để kết nối 2 trạm làm việc qua cổng song song. Đây là liên kết điểm nối điểm, khác với liên kết mạng qua bộ giao tiếp Ethernet là liên kết kiểu quảng bá.

Giả sử một máy tính xách tay là **vlite** cần truy cập vào một mạng LAN thông qua một liên kết PLIP với trạm tên là vlager. Khi khởi động hệ thống **vlite**, cổng song song có tên là *plip0*. Để kích hoạt liên kết này trên trạm **vlite** ta dùng lệnh:

```
$ ifconfig plip0 vlite pointopoint vlager
$ route add default gw vlager
```

Lệnh thứ nhất sẽ cấu hình cho giao diện *plip0* tên là **vlite** bằng cách chỉ ra đây là một liên kết điểm nối điểm tới trạm ở xa là **vlager**. Lệnh thứ hai thiết lập chọn đường mặc định cho **vlite** là đường đi qua gateway **vlager**. Tương tự như vậy ta cần phải cấu hình cho giao diện trên **vlager**. Tuy nhiên ta không cần sửa đổi bảng chọn đường trên trạm này:

```
$ ifconfig plip1 vlager pointopoint vlite
```

Giao diện *plip0* trên **vlager** không cần sử dụng một địa chỉ IP khác, nó có thể sử dụng ngay địa chỉ IP của nó trong mạng cục bộ 191.72.1.1.

Trên đây ta mới chỉ thiết lập đường đi từ trạm **vlite** tới các máy tính khác trong mạng. Để thiết lập một đường đi ngược lại ta cần thêm một đường đi từ các

trạm đến **vlite** bằng cách thêm mục chọn đường trên các trạm như sau:

```
$ route add vlite gw vlager
```

Vấn đề chọn đường đi cho các trạm tạm thời được giải quyết bằng kỹ thuật chọn đường động. Ta không thể đến tất cả các trạm để thực hiện lệnh trên được. Một trong những giải pháp chọn đường động là sử dụng chương trình *gated* trên các trạm. Khi thông tin chọn đường có sự thay đổi, ví dụ trên trạm **vlager**, *gated* trên trạm này sẽ lan truyền thông tin này đến tất cả các trạm để *gated* trên các trạm đó cập nhật thông tin chọn đường.

Một cách khác đơn giản hơn là sử dụng proxy ARP. Ta sẽ thiết lập **vlager** là proxy ARP của trạm **vlite**. Bất cứ yêu cầu ARP nào tới **vlite** sẽ được **vlager** trả lời, kết quả là các gói tin gửi cho **vlite** sẽ được gửi tới **vlager** trước khi chuyển tiếp cho nó. Proxy ARP sẽ được cập tới trong phần liên kết PPP .

6.6. Giao diện Dummy

Giao diện Dummy được xây dựng dựa trên cơ sở giao diện loopback. Nó được dùng cho các trạm đơn lẻ không được kết nối vào mạng nhưng vẫn sử dụng một số ứng dụng mạng thật sự.

Thường trên một máy trạm luôn có một giao diện được kích hoạt, đó là giao diện loopback với địa chỉ IP là 127.0.0.1. Trong một số trường hợp, một số ứng dụng mạng cần phải gửi các gói tin đến một địa chỉ IP thực sự mà không phải là địa chỉ loopback. Lấy ví dụ trạm **vlite** kết nối tạm thời vào mạng và sử dụng địa chỉ IP 191.72.1.65. Trên trạm có một ứng dụng mạng tiến hành gửi dữ liệu đến các ứng dụng khác cũng trên trạm này qua giao diện *plip0* được kích hoạt. Khi ta không sử dụng liên kết *plip0* nữa, các ứng dụng đó khi muốn gửi thông tin cho nhau sẽ tìm trong */etc/hosts* và thấy địa chỉ 191.72.1.65 và sẽ gửi thông tin đến địa chỉ này. Kết quả là ứng dụng bị lỗi mặc dù chương trình cần gửi dữ liệu nằm ngay trên máy của mình.

Đó là lý do để có một giao diện *dummy*. Hoạt động của giao diện cũng như giao diện loopback: mọi gói tin có địa chỉ 191.72.1.65 đi ra từ trạm **vlite** sẽ được gửi qua giao diện *dummy* và quay trở lại trạm đó. Để cấu hình cho giao diện này ta thực hiện các lệnh:

```
$ ifconfig dummy0 vlite
$ route add vlite
```

Phiên bản hạt nhân Linux 2.12-20 hiện thời hỗ trợ hai giao diện *dummy* trên cùng một máy trạm.

7. Lệnh ifconfig

Trong phần này ta sẽ mô tả các tham số của lệnh *ifconfig*. Lệnh này rất hay được sử dụng khi thiết lập cấu hình mạng. Cú pháp của lệnh là:

```
ifconfig interface [[ -net|-host] address [ parameters]]
```

interface là tên của giao diện, *address* là địa chỉ IP gán cho giao diện đó. Địa chỉ IP có thể viết dưới dạng ký pháp thập phân có chấm hoặc là một tên chỉ ra trong */etc/hosts* và */etc/networks*. Tuỳ chọn *net* và *host* cho biết địa chỉ đưa ra là địa chỉ của một trạm hay địa chỉ một mạng.

Khi gõ lệnh *ifconfig* mà không có tham số, lệnh trên sẽ hiển thị thông tin về tất cả các giao diện mạng đang được kích hoạt. Nếu có tham số *-a*, lệnh sẽ hiển thị tất cả các giao diện có trên hệ thống. Ví dụ :

```
$ ifconfig eth0
eth0    Link encap 10Mbps Ethernet  HWaddr 00:00:C0:90:B3:42
        inet addr 191.72.1.2 Bcast 191.72.1.255 Mask 255.255.255.0
        UP BROADCAST RUNNING  MTU 1500  Metric 0
        RX packets 3136 errors 217 dropped 7 overrun 26
        TX packets 1752 errors 25 dropped 0 overrun 0
```

Các trường MTU và Metric cho biết giá trị hiện thời của MTU và đơn vị đo của giao diện. Đơn vị đo Metric được sử dụng bởi một số hệ điều hành để đo chi phí của một đường đi. Trong phiên bản hiện thời Linux không hỗ trợ đại lượng này nhưng vẫn định nghĩa nó để bảo đảm tính tương thích.

Các trường RX và TX cho biết bao nhiêu gói tin đã nhận và gửi đi. Số lượng gói tin gửi, nhận bị lỗi, bị loại bỏ...

Các cờ *ifconfig* hiển thị tương ứng với các trạng thái của giao diện. Các cờ này được thiết lập nhờ các tuỳ chọn sau:

up — Là tuỳ chọn để cho phép giao diện có thể được truy cập từ tầng mạng. Tuỳ chọn này được sử dụng cùng với một địa chỉ IP tại dòng lệnh. Nó còn được dùng để kích hoạt lại một giao diện bị dừng tạm thời bởi tuỳ chọn down.

down Tuỳ chọn này ngăn không cho giao diện có thể truy cập được từ tầng mạng. Nó được dùng để ngăn các gói tin IP đi qua giao diện này. Tuy nhiên, tuỳ chọn trên không xoá các thành phần tương ứng trong bảng chọn đường. Do đó ta phải sửa đổi bảng chọn đường và thay thế các đường đi tương ứng.

netmask mask

Gán một mặt nạ cho giao diện. Mặt nạ có thể là một số nhị phân 31 bit viết dưới dạng số Hexa (ví dụ 0xAFF04FF0) hoặc ký pháp thập phân có chấm.

pointtopoint address

Tuỳ chọn này được sử dụng cho các liên kết điểm nối điểm như PLIP, PPP, SLIP. Khi tuỳ chọn này được thiết lập, *ifconfig* sẽ hiển thị cờ POINTTOPOINT.

Broadcast address

Thông thường địa chỉ quảng bá được tạo ra bằng cách đặt các bit trong phần địa chỉ trạm bằng 1. Một số phiên bản của IP lại sử dụng cách khác. Tuỳ chọn này được sử dụng để tương thích với các môi trường này. Khi tuỳ chọn này được thiết lập, ifconfig sẽ hiển thị cờ BROADCAST.

metric number

Tuỳ chọn này được sử dụng để gán giá trị của đơn vị đo được sử dụng trong bảng chọn đường. Đơn vị này được sử dụng bởi giao thức RIP - Routing Infomation Protocol để xây dựng các bảng chọn đường cho mạng. Đơn vị mặc định của *ifconfig* là 0. Nếu không sử dụng một RIP daemon, ta không cần sử dụng tuỳ chọn này.

arp Đây là một tuỳ chọn đặc biệt trong các mạng quảng bá như mạng Ethernet hay mạng radio. Tuỳ chọn này được sử dụng để kích hoạt giao thức ARP -Address Resolution Protocol để tìm địa chỉ vật lý ứng với một địa chỉ IP. Đối với các mạng quảng bá thì đây là tuỳ

chọn mặc định. Khi tuỳ chọn này không được thiết lập, ifconfig sẽ hiển thị cờ NOARP.

-arp Không sử dụng ARP cho giao diện này.

promisc Kích hoạt chế độ promisc, trong đó giao diện sẽ nhận tất cả các gói tin đến mà không quan tâm tới địa chỉ đích của nó. Tuỳ chọn này thường được dùng khi muốn phân tích đường đi của các gói tin trong mạng bằng bộ lọc gói tin (Ethernet snooping).

-promisc Loại bỏ chế độ promics

allmulti Kích hoạt chế độ multicast: là một dạng của broadcast nhưng các địa chỉ multicast không nhất thiết phải có chung địa chỉ mạng con. Tuỳ chọn này ứng với cờ ALLMULTI.

-allmulti Loại bỏ chế độ multicast.

8. Kiểm tra mạng bằng lệnh netstat

8.1. Bảng chọn đường

Khi thực hiện lệnh *netstat* với tham số -r, nó sẽ in ra bảng chọn đường mà hệ thống sử dụng, ví dụ:

```
$ netstat -nr
Kernel routing table
Destination     Gateway         Genmask         Flags Metric Ref
Use
127.0.0.1       *               255.255.255.255 UH    1      0
191.72.1.0      *               255.255.255.0   U     1      0
191.72.2.0      191.72.1.1      255.255.255.0   UGN   1      0
```

Với tuỳ chọn -n, *netstat* sẽ in ra địa chỉ IP theo ký pháp thập phân có chấm thay vì tên miền của trạm. Với tham số này ta tránh được việc sử dụng dịch vụ ánh xạ địa chỉ bằng DNS hay NIS khi hiển thị bảng chọn đường.

Dòng thứ hai trong đầu ra của *netstat* chỉ gateway mà thành phần tương ứng trong bảng chọn đường chỉ tới. Nếu không có gateway nào thì ta dùng dấu sao (*).

Khi tìm đường cho một gói tin IP, hạt nhân sẽ làm một phép AND các bit địa chỉ IP của gói tin với mặt nạ genmask trước khi đem so sánh với phần Destination của bảng chọn đường.

Cột thứ tư của bảng chọn đường là các cờ có ý nghĩa như sau:

G	Chọn đường qua một gateway
U	Giao diện sử dụng đang được kích hoạt
H	Chỉ có các trạm riêng lẻ mới được sử dụng đường đi này. Ví dụ địa chỉ loopback 127.0.0.1
D	Thành phần này trong bảng chọn đường được thiết lập từ thông báo ICMP redirect.
M	Thành phần này trong bảng chọn đường được sửa từ thông báo ICMP redirect.

Cột Ref trong bảng chọn đường cho biết số lần số tham chiếu đến đường đi này, có nghĩa là có bao nhiêu đường đi khác dựa trên đường đi đó. Cột cuối cùng là số lần mà đường đi được sử dụng tên giao diện mà các gói tin đi qua.

8.2. Thống kê về giao diện mạng

Khi dùng tuỳ chọn -i, *netstat* sẽ hiển thị các thông tin về giao diện đã được sử dụng. Với tham số -a, tất cả các giao diện trong hạt nhân sẽ được hiển thị.

```
$ netstat -i
Kernel Interface table
Iface MTU Met RX-OK RX-ERR RX-DRP RX-OVR TX-OK TX-ERR TX-DRP TX-OVR
lo     0   0   3185      0      0      0  3185      0      0
eth0 1500 0 972633     17     20    120 628711    217      0
```

Trường MTU và Met cho biết giá trị MTU và đơn vị đo hiện thời của giao diện. RX và TX cho biết có bao nhiêu gói tin đã được nhận và gửi qua giao diện. OK là không lỗi, còn ERR là có lỗi. DRP là bị loại bỏ còn OVR là bị loại bỏ do quá tải.

Cột cuối cùng chỉ ra các cờ đã được thiết lập cho giao diện này. Các cờ này

là tên rút gọn của các cờ chỉ ra trong lệnh *ifconfig*.

B	Đã thiết lập địa chỉ quảng bá.
L	Đây là địa chỉ loopback
M	Tất cả các gói tin đều được nhận (Không phân biệt lỗi hay không)
N	Tránh ghi lại vết cho gói tin
O	ARP không được kích hoạt cho giao diện này.
P	Đây là một liên kết điểm nối điểm
R	Giao diện đang được chạy
U	Giao diện được kích hoạt

8.3. Thông tin về liên kết mạng

Netstat hỗ trợ một tập hợp các tuỳ chọn để hiển thị thông tin về các socket được kích hoạt hay không.. Các tuỳ chọn là -t -u -w -x hiển thị các socket TCP, UDP, RAW và UNIX đang kích hoạt. Tuỳ chọn -a hiển thị tất cả các socket đang chờ yêu cầu liên kết, tức là tất cả các server đang chạy trên hệ thống.

```
# netstat -ta
Active Internet connections
Proto Recv-Q Send-Q Local Address Foreign Address (State)
tcp       0      0 *:domain        *:*             LISTEN
tcp       0      0 *:time          *:*             LISTEN
tcp       0      0 *:smtp          *:*             LISTEN
tcp       0      0 vlager:smtp     vstout:1040     STABLISHED
tcp       0      0 *:telnet         *:*            LISTEN
tcp       0      0 localhost:1046 vbardolino:telnet ESTABLISHED
tcp       0      0 *:chargen        *:*            LISTEN
tcp       0      0 *:daytime        *:*            LISTEN
```

```
tcp        0      0 *:discard               *:*                     LISTEN
tcp        0      0 *:echo                  *:*                     LISTEN
tcp        0      0 *:shell                 *:*                     LISTEN
tcp        0      0 *:login                 *:*                     LISTEN
```

Trong ví dụ trên dòng thứ 4 cho biết một liên kết SMTP từ trạm **vtous**, dòng thứ 6 chỉ một phiên làm việc *telnet* từ trạm **vbardolino**. Các dòng khác cho biết các server đang nghe yêu cầu dịch vụ trên socket tương ứng với nó.

9. Kiểm tra bảng ARP

Vì lý do nào đó mà ta có thể phải xem lại nội dung hay sửa đổi bảng ARP của hạt nhân. Công cụ *arp* cho phép thực hiện điều đó. Các tuỳ chọn của lệnh *arp* là:

```
arp [ -v] [ -t hwtype] -a [ hostname]
arp [ -v] [ -t hwtype] -s hostname hwaddr
arp [ -v] -d hostname [ hostname...]
```

Tham số *hostname* có thể viết dưới dạng tên máy trạm hoặc địa chỉ IP. Dòng lệnh thứ nhất hiển thị thành phần ARP của một máy trạm khi chỉ ra tên máy hoặc địa chỉ IP. Nếu không có tên máy thì tất cả các thành phần sẽ được đưa ra:

```
$ arp -a
IP address       HW type             HW address
191.72.1.3       10Mbps Ethernet     00:00:C0:5A:42:C1
191.72.1.2       10Mbps Ethernet     00:00:C0:90:B3:42
191.72.2.4       10Mbps Ethernet     00:00:C0:04:69:AA
```

Sử dụng tham số -t để hạn chế việc hiển thị các thành phần tương ứng với những dạng phần cứng nhất định. Các dạng có thể là *ether*, *ax*25, pronet tương ứng với 10Mbps Ethernet, AMPR AX.25 và thiết bị dạng token ring IEEE802.5.

Tham số -s được dùng để thêm một thành phần vào bảng ARP. Tham số *hwaddr* là địa chỉ phần cứng của thiết bị, mặc định là địa chỉ Ethernet với 48 bit. Có thể đưa vào các dạng phần cứng khác bằng tham số -t.

Lý do để sửa đổi bảng ARP bằng tay là trong một số trường hợp, yêu cầu ARP đến một trạm khác bị lỗi, vì lý do trình điều khiển ARP bị lỗi hay một trạm khác nữa tự gán cho nó địa chỉ IP trùng với trạm ta cần liên lạc. Khi đó việc liên kết cứng giữa địa chỉ IP và địa chỉ vật lý trong bảng ARP là một biện pháp để khắc phục vấn đề này.

Lệnh *arp -d* sẽ xoá một thành phần trong bảng ARP của một trạm chỉ ra. Lệnh này được dùng khi ta bắt hạt nhân phải thử tìm lại ánh xạ giữa địa chỉ IP và địa chỉ vật lý của trạm đó. Điều này là cần thiết khi một trạm cấu hình sai lại thực hiện quảng bá các thông tin không đúng về ARP của nó.

Tham số -s dùng để cài đặt một proxy ARP. Đây là một kỹ thuật đặc biệt khi một trạm làm một gateway cho một trạm khác. Ví dụ trạm gate được uỷ quyền là gateway cho trạm **may1**. Khi đó trạm gate sẽ phát ra một thông điệp ARP trong đó địa chỉ IP của may1 ứng với địa chỉ vật lý của gate. Khi một trạm nào đó gửi một yêu cầu ARP đến may1, gate sẽ nhận được yêu cầu đó và trả lời là địa chỉ vật lý của nó. Như vậy tất cả các gói tin gửi cho may1 sẽ đến gate và sẽ được chuyển tiếp đến may1.

Kỹ thuật proxy ARP được dùng khi một trạm MS-DOS kết nối vào mạng mà không quan tâm nhiều đến thông tin chọn đường. Nó cũng được ứng dụng rộng rãi khi một trạm kết nối vào mạng qua cổng song song PLIP, liên kết quay số như PPP, SLIP. Để cài một proxy ta thực hiện lệnh trên gate:

```
$ arp -s may1 00:00:c0:a1:42:e0 pub
```

Thành phần proxy ARP được xoá bằng lệnh:

```
$ arp -d may1
```

IV. LIÊN KẾT MẠNG IP QUA ĐƯỜNG TRUYỀN NỐI TIẾP

Các giao thức SLIP - Serial Line IP và PPP - Point to Point Protocol được sử dụng để một trạm kết nối vào Internet qua đường điện thoại. Để làm việc với các giao thức trên, máy tính chỉ cần trang bị một modem và một cổng nối tiếp. Do tính đơn giản và tiện lợi, ngày càng có nhiều ISP cung cấp dịch vụ kết nối Internet bằng cách quay số này.

1. Giao thức SLIP

1.1. Các yêu cầu chung

Để sử dụng SLIP hay PPP, ta cần thiết lập một số tính năng cơ bản của mạng như TCP/IP, giao diện loopback và giải pháp ánh xạ địa chỉ. Khi kết nối với Internet, ta nên sử dụng dịch vụ tên miền DNS. Một cách đơn giản là sử dụng Name Server của nhà cung cấp dịch vụ, server này sẽ được sử dụng mỗi khi SLIP hoạt động. Nên chọn Name Server càng gần với nơi quay số vào càng tốt.

Giải pháp trên là không tối ưu vì mọi yêu cầu DNS đều phải đi qua liên kết SLIP/PPP. Để khỏi lãng phí băng thông của các liên kết này, ta nên sử dụng một caching-only Name Server cho dịch vụ DNS. Máy chủ này không cung cấp dịch vụ cho một miền nào, mà chỉ chuyển tiếp các yêu cầu DNS cho trạm sử dụng SLIP và lưu lại kết quả tìm kiếm. Ưu điểm của nó là hầu hết các yêu cầu DNS chỉ được gửi qua liên kết SLIP một lần. Máy chủ này có tệp *named.boot* như sau:

```
; named.boot tệp for caching-only server
directory                              /var/named
primary          0.0.127.in-addr.arpa    db.127.0.0 ; loopback
net
cache                                          db.cache    ; root
servers
```

Để có được tệp db.cache chứa địa chỉ các Name Server gốc, ta dùng tiện ích nslookup như đã mô tả trong chương trước.

1.2. Hoạt động của SLIP

Các máy chủ có dịch vụ quay số vào thường cho phép kết nối bằng SLIP qua các account của người dùng. Khi đăng nhập vào, ta không đăng nhập vào một shell như thường lệ, thay vào đó là một chương trình hay một shell script được thực hiện để kích hoạt trình điều khiển SLIP trên máy chủ và đặt cấu hình cho giao diện mạng tương ứng.

Trong một số hệ điều hành, trình điều khiển SLIP là chương trình của người sử dụng. Trong Linux, nó là một phần của hạt nhân, do đó tốc độ của nó nhanh hơn. Để sử dụng SLIP thì cổng nối tiếp phải được chuyển sang chế độ SLIP. Trong chế độ thường, các tty sẽ trao đổi dữ liệu bình thường với chương trình của người sử dụng. Trong chế độ SLIP, mọi yêu cầu đọc và ghi cổng của các tiến trình của

người dùng sẽ bị chặn lại, tất cả dữ liệu đến cổng nối tiếp đều được gửi qua trình điều khiển SLIP.

Trình điều khiển SLIP có một số các biến thể tương thích với nó, như CSLIP, một trình điều khiển cổng nối tiếp có hỗ trợ header nén khi gửi các gói tin IP. Trình điều khiển này làm tăng thông lượng cho các phiên làm việc tương tác. Mỗi trình điều khiển của SLIP đều có phiên bản 6 bit dữ liệu.

Một cách để chuyển cổng nối tiếp sang chế độ SLIP là sử dụng công cụ slattach. Giả sử chúng ta có một modem được nối với cổng nối tiếp */dev/cua3*, ta thực hiện lệnh:

```
$ slattach /dev/cua3 &
```

Lệnh trên sẽ chuyển *cua3* sang chế độ SLIP và gắn cho nó một giao diện mạng SLIP tương ứng. Nếu đây là giao diện SLIP đầu tiên, tên của nó sẽ là *sl0*, tiếp theo là giao diện *sl1*... Hạt nhân Linux hiện hỗ trợ 8 giao diện SLIP.

Chế độ mặc định của *slattach* là *cslip*. Để chuyển sang các chế độ khác ta dùng tham số -p, ví dụ để kích hoạt SLIP (không header nén):

```
$ slattach -p slip /dev/cua3&
```

Các chế độ khác là *cslip*, *slip6*, *cslip6* (các phiên bản 6 bit của SLIP) và *adaptive*. Chế độ *adaptive* cho phép hạt nhân của hệ điều hành tìm chế độ của SLIP mà trạm ở xa sử dụng. Chú ý rằng các trạm ở hai đầu liên kết SLIP phải sử dụng chung một chế độ, vậy tốt hơn là sử dụng từ khoá *adaptive*.

Công cụ SLIP không những cho phép kích hoạt trình điều khiển SLIP mà còn được sử dụng cho các giao thức khác như PPP hay KISS. Để biết thêm chi tiết hãy tham khảo tài liệu giúp đỡ của *slattach* bằng lệnh *man*.

Sau khi quay số và kích hoạt trình điều khiển SLIP, ta cần phải cấu hình cho giao diện mạng tương ứng với nó bằng các lệnh *ifconfig* và *route*. Giả sử ta cần thiết lập liên kết từ trạm **vlager** đến một máy chủ tên là **cowslip**:

```
$ ifconfig sl0 vlager pointopoint cowslip
$ route add cowslip
$ route add default gw cowslip
```

Dòng lệnh đầu tiên cấu hình giao diện là một liên kết điểm nối điểm tới **cowslip**, các lệnh sau thêm các đường đi tới **cowslip** và sử dụng **cowslip** như một gateway chọn đường mặc định.

Khi loại bỏ một liên kết SLIP, trước tiên ta phải bỏ hết các mục chọn đường qua **cowslip** bằng lệnh *route del*. Dừng giao diện lại và gửi tín hiệu *hangup* đến **slattach** và sau cùng là tắt modem đi.

```
$ route del cowslip
$ ifconfig sl0 down
$ kill -HUP 516
```

1.3. Sử dụng trình thông dịch lệnh dip

Các thao tác để thiết lập một liên kết SLIP như trên có thể được thực hiện nhờ một tiện ích là *dip*. Đây là một trình thông dịch ngôn ngữ kịch bản đơn giản cho các lệnh quản lý modem, chuyển đường truyền sang chế độ SLIP và cấu hình giao diện mạng.

dip đòi hỏi mức đặc quyền cao nhất trong việc cấu hình cho giao diện SLIP. Ta có thể để cho dip tự động thiết lập mức đặc quyền của root bằng lệnh *setuid*. Phương pháp này không an toàn vì các lệnh của *dip* có thể làm hỏng bảng chọn đường khi nó sửa đổi chọn đường mặc định. Hơn nữa, việc này cho phép người sử dụng có thể kết nối đến bất cứ một SLIP server nào trên mạng, làm cho mạng khó được bảo vệ an toàn. Do đó, nếu muốn người dùng sử dụng một liên kết SLIP đến một máy chủ nào đó, ta nên viết một chương trình riêng ứng với máy chủ đó, chương trình này sẽ gọi các lệnh *dip* để thiết lập liên kết. Khi đó việc sử dụng SLIP sẽ an toàn hơn.

1.3.1 Một ví dụ về chương trình dip

Dưới đây là một ví dụ đơn giản của một chương trình script để kết nối với máy chủ **cowslip**:

```
# Ví dụ một dip script để quay số vào trạm cowslip
# Gán tên các tram cho các biến local và remote
get $local vlager
get $remote cowslip
port cua3                  # Chọn cổng nối tiếp số 3
speed 38400                # Thiết lập tốc độ cực đại
modem HAYES                # Thiết lập kiểu modem
reset                      # Khởi tạo modem
```

```
flush                    # Làm sạch vùng đệm của modem
# Chuẩn bị quay số
send ATQ0V1E1X1\r
wait OK 2
if $errlvl != 0 goto error
dial 0123456789
if $errlvl != 0 goto error
wait CONNECT 60
if $errlvl != 0 goto error
# Đã kết nối xong
sleep 3
send \r\n\r\n
wait ogin: 10
if $errlvl != 0 goto error
send Cvlager\r
wait ssword: 5
if $errlvl != 0 goto error
# Nhập mật khẩu
password
wait running 30
if $errlvl != 0 goto error
# Hiển thị thông tin về 2 đầu liên kết
get $remote remote
print remote = $remote
if $errlvl != 0 goto error
wait to 3
get $local remote
print local = $local
if $errlvl != 0 goto error
# Đã đăng nhập thành công.
print Connected to $remote with address $rmtip
```

```
default                    #  Chế độ chọn đường mặc định
mode CSLIP                 #  Dùng chế độ CSLIP
# Trong trường hợp có lỗi
error:
print CSLIP to $remote failed.
```

Ta có thể kết nối đến trạm cowslip bằng cách gõ lệnh dip với tham số là chương trình script:

```
# dip cowslip.dip
DIP: Dialup IP Protocol Driver version 3.3.7 (12/13/93)
Written by Fred N. van Kempen, MicroWalt Corporation.
connected to cowslip.moo.com with addr 193.174.7.129
```

Sau khi kết nối với **cowslip** và kích hoạt giao diện CSLIP, *dip* sẽ thoát khỏi terminal và chạy như một tiến trình mặt sau. Ta đã có thể sử dụng các dịch vụ mạng thông thường qua liên kết CSLIP. Để loại bỏ kết nối, ta chỉ việc gõ dip -k, lệnh này sẽ kết thúc liên kết bằng cách lấy số hiệu tiến trình của dip trong */etc/dip.pid*.

```
# dip -k
```

Trong ngôn ngữ của *dip*, từ khoá bắt đầu bằng dấu $ là một biến. *Dip* có một tập các biến được định nghĩa trước sẽ được liệt kê ở dưới đây. *$remote* and *$local* là các biến chứa tên của trạm cục bộ và trạm ở xa trong liên kết SLIP.

Trong hai lệnh đầu tiên của script, các biến *$local* và *$remote* được gán các giá trị tương ứng bằng lệnh *get*.

Năm lệnh tiếp theo thiết lập kênh truyền và modem. Lệnh *reset* gửi một xâu tới modem để kích hoạt nó . Đối với các modem thuộc họ Hayes hoặc tương thích, lệnh *reset* là ATZ. Lệnh *flush* để làm sạch bộ đệm trả lời của modem cho lần làm việc tiếp theo. Sau đó, chương trình sẽ quay số vào **cowslip** bằng lệnh dial 41988 với tài khoản là **Vlager** và mật khẩu là hey-jude. Lệnh *wait ssword: 5* làm cho modem chờ xâu ssword trong vòng 5 giây trước khi đưa ra một timeout. Nếu nhận được xâu trả lời đúng, liên kết đã được thiết lập thành công.

Lệnh cuối cùng đặt chế độ chọn đường mặc định, chuyển cổng nối tiếp sang chế độ SLIP và cấu hình giao diện mạng cho hệ thống.

1.3.2. Một số lệnh của dip

Trong phần này ta sẽ xem xét một số lệnh của dip. Để biết *dip* có những lệnh nào ta gõ *help* tại dấu nhắc lệnh. Muốn biết cách sử dụng một lệnh, ta gõ lệnh đó mà không có tham số. Tuy nhiên cách này không thực hiện được đối với các lệnh không nhận tham số nào.

```
$ dip -t
DIP: Dialup IP Protocol Driver version 3.3.7 (12/13/93)
Written by Fred N. van Kempen, MicroWalt Corporation.
DIP> help
DIP knows about the following commands:
databits default  dial     echo     flush
get      goto     help     if       init
mode     modem    parity   print    port
reset    send     sleep    speed    stopbits
term     wait
DIP> echo
Usage: echo on|off
DIP>
```

a. Các lệnh modem

Các lệnh modem dùng để thiết lập các tham số cho đường truyền và modem. Một số lệnh hay gặp là lệnh *port* dùng để lựa chọn cổng nối tiếp, các lệnh *speed*, *databits*, *stopbits*, *parity* dùng để thiết lập tham số đường truyền.

Lệnh modem dùng để lựa chọn kiểu modem ta dùng. Hiện tại, Linux hỗ trợ modem kiểu HAYES. Ta phải thực hiện lệnh này để dip biết được dạng của modem , nếu không nó sẽ không thực hiện được các lệnh *dial* và *reset*. Lệnh *reset* thiết lập modem bằng cách gửi một xâu đến modem, xâu này tuỳ thuộc vào loại của modem, với các modem họ HAYES, xâu này là ATZ.

Lệnh *flush* dùng để loại hết các tín hiệu trả lời mà modem nhận được trước đó. Nếu không dùng lệnh này thì một số ứng dụng sau đó không hoạt động được. Ví dụ chat phải nhận được tín hiệu OK trước khi thực hiện.

Lệnh *init* chọn một xâu khởi tạo để gửi cho modem trước khi quay số, xâu mặc định của HAYES là "ATE Q0 V1 X1", xâu này có tác dụng kích hoạt chế độ

phản hồi lệnh, lựa chọn chế độ quay số.

Lệnh *dial* sẽ thực hiện gửi lệnh khởi tạo tới modem và tiến hành quay số đến trạm ở xa. Lệnh quay số mặc định của các modem họ HAYES là STD.

b. Lệnh *echo*

Lệnh *echo* là một công cụ gỡ rối. Với *echo on*, *dip* sẽ gửi trả về màn hình tất cả các lệnh mà nó gửi đi. Để thoát khỏi chế độ này, ta gõ lệnh *echo off*.

dip cho phép tạm thời thoát khỏi chế độ thông dịch lệnh và trở về màn hình terminal. Trong chế độ này ta có thể sử dụng dip như một ứng dụng thông thường dùng để gửi và nhận dữ liệu từ cổng nối tiếp. Để thoát khỏi chế độ này ta gõ phím "Ctrl-]".

c. Lệnh *get*

Lệnh *get* dùng để đặt giá trị cho một biến. Ta thường gán cho biến một giá trị hằng và sử dụng trong suốt chương trình. Có thể nhập giá trị của biến từ dòng lệnh bằng cách dùng từ khoá *ask*:

```
DIP> get $local ask
Enter the value for $local:
```

Cách thứ ba là nhận giá trị của biến từ trạm ở xa. Một số máy chủ SLIP không cho phép ta sử dụng địa chỉ IP riêng trong liên kết SLIP, nhưng cho phép ta nhận một địa chỉ IP khi bắt đầu kết nối vào. Nếu nhận được thông báo ``Your address: 193.174.7.202", ta dùng các lệnh dip sau để nhận địa chỉ IP:

```
wait address: 10
get $locip remote
```

c. Lệnh *print*

Lệnh này dùng để in ra màn hình các dòng thông báo, giá trị các biến ra màn hình khi thực hiện, ví dụ:

```
DIP> print Using port $port at speed $speed
Using port cua3 at speed 38400
```

d. Tên biến

dip chỉ có thể hiệu được các biến được định nghĩa sẵn. Tên biến được bắt đầu với dấu $ và phải được viết bằng chữ thường.

Biến *$local* và *$locip* chứa tên và địa chỉ IP. Khi thiết lập tên máy, dip sẽ lưu tên chính thức của trạm cục bộ trong *$local* và gán địa chỉ IP tương ứng cho *$locip*. Tương tự như vậy khi ta gán địa chỉ IP cho biến *$locip*.

Biến *$remote* và *$rmtip* giống như hai biến trên nhưng cho trạm ở xa. *$mtu* chứa giá trị MTU cho kết nối. (MTU -Maximum Transfer Unit).

Năm biến trên là các biến duy nhất được gán giá trị thông qua lệnh *get*. Các biến khác phải được gán qua các lệnh đặc biệt như *$modem, $port, $speed*.

$errlvl là biến nhận kết quả trạng thái làm việc của lệnh trước đó. Giá trị 0 ứng với không có lỗi nào, giá trị 1 tương ứng với một lỗi.

e. Lệnh *if* và *goto*.

if là một lệnh rẽ nhánh có điều kiện. Cú pháp của lệnh này như sau:

if `var op number goto label`

Biểu thức so sánh phải là sự so sánh đơn giản giữa các biến $errlvl, $locip, $rmtip, toán hạng thứ hai là một số nguyên. Toán tử là một trong ==, !=, <, >, <= và >=.

Lệnh *goto* sẽ thực hiện các lệnh bắt đầu từ nhãn *label*. Một nhãn phải được viết trên một dòng và kết thúc bằng dấu hai chấm.

f. Lệnh *send, wait* và *sleep*

Các lệnh này được dùng để viết một ứng dụng chatting đơn giản trong dip. Lệnh *send* gửi một xâu lên đường truyền, nó không hỗ trợ các biến nhưng hiểu được các ký tự đặc biệt của C như \n, \b. Ký tự (~) được dùng thay cho các ký tự CRLF.

Lệnh *wait word timeout* dùng để duyệt tất cả các đầu vào của modem cho đến khi nhận được từ *word* trong khoảng thời gian timeout (s). Từ đó không được chứa dấu cách. Nếu sau thời gian timeout nó không nhận được từ mong muốn thì sẽ trả về lỗi trong *$errlvl*.

Lệnh *sleep* dùng để chờ một khoảng thời gian cho chương trình hoàn thành một công việc nào đó. Đơn vị thời gian là giây.

g. Lệnh *mode* và *default*

Các lệnh này dùng để chuyển chế độ của SLIP và cấu hình giao diện. *Mode* là lệnh cuối cùng mà dip thực hiện trước khi chuyển sang hoạt động của một

daemon. Nếu không có lỗi xảy ra thì lệnh này không trả về kết quả thực hiện.

Lệnh *mode* có tham số là tên giao thức tuần tự. Các giao thức mà *dip* hỗ trợ là SLIP và CSLIP. Sau khi kích hoạt chế độ của cổng nối tiếp, *mode* sẽ gọi lệnh *ifconfig* để thiết lập giao diện mạng như một liên kết điểm nối điểm, cuối cùng nó gọi lệnh route để thiết lập bảng chọn đường tới trạm ở xa.

Nếu trước khi thực hiện lệnh *mode*, *dip* đã thực hiện lệnh *default* thì nó sẽ thiết lập chọn đường mặc định tới liên kết SLIP này.

1.4. Máy chủ SLIP

Việc thiết lập cấu hình cho một máy chủ SLIP phục vụ cho quay số (dial-in) vào đơn giản hơn nhiều so với thiết lập một trạm khách để quay số ra (dial-out).

Để sử dụng SLIP ở chế độ máy chủ, ta dùng chương trình *diplogin*. Tệp cấu hình của chương trình này là */etc/diphosts*, trong đó chỉ ra tên đăng nhập và địa chỉ trạm quay số. Ngoài ra ta có thể sử dụng chương trình *sliplogin*, một công cụ có những tính năng linh hoạt, cho phép thực hiện các shell script mỗi khi đăng nhập và thoát khỏi hệ thống.

Cả hai chương trình trên đều đòi hỏi có một tài khoản cho mỗi người dùng muốn quay số vào. Ví dụ ta muốn cấp một tài khoản cho Le Van Anh tại trạm **httt.hut-it.edu.vn** quay số vào, ta tạo một người sử dụng mới với các thuộc tính như sau:

```
anhlv:*:501:500:Le Van Anh SLIP account:/tmp:/usr/sbin/diplogin
```

Khi anhlv muốn đăng nhập vào hệ thống, dip sẽ khởi động ở chế độ server. Để kiểm tra xem anhlv có được quyền sử dụng SLIP hay không, *dip* sẽ kiểm tra tệp tin */etc/diphosts*. Tệp cấu hình này cho biết các chi tiết về quyền truy cập và các tham số của liên kết cho mỗi người sử dụng. Một thành phần trong tệp trên cho anhlv có dạng:

```
anhlv::httt.hut-it.edu.vn:Le Van Anh:SLIP,296
```

Tham số trước dấu hai chấm thứ nhất là tên đăng nhập, tham số thứ hai có thể là mật khẩu bổ sung (sẽ giải thích sau). Tiếp theo là địa chỉ IP của trạm muốn quay số. Mục cuối cùng là các tham số của liên kết, trong ví dụ trên là tên giao thức và giá trị của MTU.

Khi anhlv quay số để đăng nhập vào hệ thống, *diplogin* sẽ nhận các thông tin từ */etc/diphosts*. Nếu trường mật khẩu không rỗng, anhlv sẽ được hỏi thêm một

mật khẩu để tăng tính an toàn, đây gọi là mật khẩu ngoài. Nếu mật khẩu đưa ra không trùng với giá trị chỉ ra trong */etc/hosts*, anhlv sẽ không được đăng nhập. Nếu đúng, *diplogin* sẽ chuyển sang chế độ SLIP hoặc CSLIP, thiết lập giao diện mạng và bảng chọn đường. Kết nối này được duy trì khi có yêu cầu kết thúc từ phía người dùng. Khi đó *diplogin* sẽ đặt cổng nối tiếp ở chế độ bình thường và dừng chương trình.

2. Giao thức PPP

2.1. Giới thiệu giao thức PPP

Cũng như SLIP, PPP là giao thức cho phép trao đổi dữ liệu qua cổng nối tiếp. Với nhiều tính năng hơn, giao thức này cho phép người sử dụng chọn các tham số khi kết nối như địa chỉ IP, kích thước gói tin, quyền của người dùng. Với mỗi chức năng, PPP có một giao thức tương ứng. Dưới đây chúng ta sẽ tìm hiểu sơ lược các thành phần của giao thức PPP.

Giao thức bên dưới cùng của PPP là giao thức liên kết dữ liệu High-level Data Link Control Protocol hay HDLC, là giao thức đóng gói các khung tin của PPP và cung cấp mã sửa lỗi checksum 16 bit. So với SLIP chỉ hoạt động với giao thức liên mạng IP, PPP có khả năng tương thích với các giao thức tầng mạng khác như Novell's IPX, Apple Talk. PPP thêm vào trong các khung tin HDLC một trường số hiệu giao thức để xác định giao thức tầng mạng.

Phía trên HDLC là giao thức LCP - Link Control Protocol, được sử dụng để thương lượng các tuỳ chọn cho việc liên kết dữ liệu như MRU - Maximum Receive Unit, là kích thước gói tin mà mỗi bên của liên kết đồng ý nhận.

Một trong những bước quan trọng khi cấu hình cho một liên kết PPP là quyền hạn người dùng. Thông thường khi quay số vào một máy chủ, trạm khách sẽ phải trả lời một vài khoá bí mật nào đó, nếu sai thì kết nối sẽ bị huỷ bỏ. Với PPP, vấn đề chứng thực sẽ được tiến hành cả hai chiều, có nghĩa là người dùng cũng có thể đề nghị máy chủ xác nhận chính nó. Các thủ tục chứng thực là độc lập đối với mỗi bên. Có hai giao thức khác nhau cho vấn đề chứng thực, đó là PAP - Pasword Authentication Protocol và CHAP - Challenge Handshake Authentication Protocol.

Khi các giao thức tầng mạng như IP, Apple Talk gửi các gói tin qua cổng nối tiếp, chúng sẽ được cấu hình động bằng giao thức Network Control Protocol - NCP. Ví dụ khi các thực thể PPP truyền dữ liệu qua liên kết nối tiếp, chúng phải thương

lượng để biết địa chỉ IP của mỗi bên. Giao thức NCP trong trường hợp này là IPCP - Internet Protocol Control Protocol.

Ngoài việc gửi các gói tin IP chuẩn qua liên kết nối tiếp, PPP còn hỗ trợ các gói tin IP có header nén (Van Jacobson IP header). Đây là kỹ thuật cho phép thu nhỏ kích thước header của gói tin TCP xuống còn 3 bytes. Kỹ thuật này còn được sử dụng trong CSLIP với tên là header nén VJ . PPP cho phép lựa chọn kỹ thuật nén này có được dùng hay không khi kết nối thông qua giao thức IPCP.

2.2. PPP trong Linux

Trong Linux, các chức năng của PPP được chia làm hai phần chính, phần thứ nhất là trình điều khiển HDLC ở mức thấp được cài đặt trong hạt nhân. Phần chương trình người dùng là daemon pppd để quản lý các giao thức điều khiển. Phiên bản hiện thời của Linux có modul PPP của hạt nhân, pppd và chương trình chat dùng để quay số đến trạm khác.

Cũng như SLIP, để sử dụng PPP ta thiết lập một liên kết qua modem và chuyển cổng nối tiếp sang chế độ PPP. Trong chế độ này, tất cả dữ liệu đến đều **được chuyển cho trình điều khiển PPP để kiểm tra checksum**, mở gói tin để nhận dữ liệu và gửi lên trên. Chương trình này có khả năng quản lý các gói tin IP, hỗ trợ tuỳ chọn VJ header và có thể mở rộng để quản lý các gói tin IPX.

Trình điều khiển của PPP trong hạt nhân được hỗ trợ bởi pppd hay PPP daemon. pppd thực hiện việc khởi tạo và các quá trình chứng thực trước khi dữ liệu được gửi qua liên kết này. PPP là một giao thức phức tạp và có rất nhiều các tuỳ chọn để hoạt động, trong các phần tiếp theo ta chỉ giải thích một số tuỳ chọn cơ bản. Để biết thêm chi tiết về PPP hãy tham khảo tài liệu man và READMEs của nó.

2.3. Thực hiện chương trình pppd

Để kết nối vào Internet qua liên kết PPP, trước tiên ta cần thiết lập một số tính năng cơ bản như TCP/IP, giao diện loopback , giải pháp ánh xạ địa chỉ. Phần này được tiến hành giống như đối với liên kết SLIP.

Giả sử ta đã có một máy chủ là **p3server** cho phép thiết lập liên kết PPP tới nó bằng cách quay số. Sau khi dùng các chương trình quay số để kết nối tới máy chủ này, ta tạo một liên kết ppp bằng lệnh:

```
$ pppd /dev/cua3 38400 crtscts defaultroute
```

Lệnh trên sẽ chuyển cổng nối tiếp *cua3* sang chế độ PPP và thiết lập một kết nối IP với máy chủ **p3server**. Tốc độ truyền sử dụng là 38400kpbs. Với tốc độ truyền lớn hơn 9600 bps, cần thiết lập chế độ có tín hiệu bắt tay tại các cổng bằng tuỳ chọn *crtscts*.

Sau khi khởi động, việc đầu tiên là *pppd* tiến hành thương lượng các tham số của liên kết với trạm ở xa bằng giao thức LCP. Thông thường các tuỳ chọn mặc định sẽ được *pppd* thương lượng. Chi tiết về LCP sẽ được đề cập chi tiết trong phần sau.

Tiếp theo, *pppd* sẽ xác định địa chỉ IP của hai đầu liên kết bằng giao thức IPCP. Giao thức này sẽ tìm các địa chỉ IP thông qua các **resolver** trên trạm cục bộ. Sau đó, mỗi bên sẽ gửi địa chỉ IP của mình cho phía bên kia. Cho dù máy trạm là thành viên của một mạng LAN, ta vẫn có thể sử dụng một địa chỉ IP khác ứng với PPP cho nó.

Sau bước thực hiện IPCP, *pppd* sẽ cấu hình cho tầng mạng để sử dụng liên kết PPP. Trước tiên *pppd* sẽ tạo một giao diện mạng cho liên kết điểm-nối-điểm, tên là *ppp0* cho giao diện *ppp* thứ nhất, *ppp1* cho giao diện thứ hai... Tiếp đến thêm các thành phần cho bảng chọn đường, trong lệnh trên chọn đường mặc định sẽ chỉ tới **p3server** do có tham số defaultroute khi khởi động *pppd*.

2.4. Sử dụng các tệp tin cấu hình

Trước khi pppd phân tích các tham số dòng lệnh, nó sẽ duyệt các tệp tin cấu hình để nhận các tham số mặc định. Tệp tin này có thể chứa bất kỳ một tham số nào trong các tham số dòng lệnh của pppd.

Tệp cấu hình thứ nhất là */etc/ppp/options*. Sử dụng tệp tin này cho phép người dùng không mắc phải các sai sót thiếu tính an toàn, chẳng hạn để *pppd* thực hiện một số dạng chứng thực (PAP hay CHAP), ta nên thêm tham số auth vào tệp này. Tuỳ chọn này không bị các tham số dòng lệnh làm thay đổi.

Tệp cấu hình được đọc tiếp là *.ppprc* trong thư mục home của người sử dụng. Tệp này chứa các tuỳ chọn mặc định riêng của người sử dụng.

Ví dụ về tệp cấu hình */etc/ppp/options*:

```
# Global options for pppd running on vlager.vbrew.com
auth                    # require authentication
usehostname             # use local hostname for CHAP
```

```
lock                    # use UUCP-style device locking
domain vbrew.com        # our domain name
```

Hai tuỳ chọn đầu tiên dành cho việc chứng thực sẽ được giải thích sau. Với Từ khoá lock, *pppd* sẽ tuân theo việc khoá các thiết bị sử dụng theo chuẩn UUCP bằng cách tạo tệp tin *LCK.cua_x* trong thư mục spool của UUCP để báo cho các chương trình khác biết cổng *cua_n* đã được sử dụng.

Một số tuỳ chọn trong tệp tin */etc/ppp/options* không bị ghi đè bởi người sử dụng như tuỳ chọn auth, do đó nó bảo đảm một mức an toàn nhất định.

2.5. Quay số tới máy chủ PPP bằng chat

Để có thể tạo một liên kết PPP, ta cần phải quay số vào PPP server trước khi thực hiện *pppd*. Không giống như *dip*, *pppd* không có các script để thực hiện quay số, ta phải dùng các chương trình hoặc các shell script để quay số. Các chương trình quay số có thể được chỉ ra trong tham số dòng lệnh của *pppd* bằng tuỳ chọn *connect*. *pppd* sẽ hướng các đầu vào, ra của chương trình này đến cổng nối tiếp. Dưới đây chúng ta sẽ dung chương trình *chat* để quay số.

Chat là một chương trình dùng để gửi và nhận các xâu tới một trạm ở xa. Chúng ta gọi là các xâu đợi và các xâu gửi, các xâu đợi, ví dụ:

login: anhlv passwd: vbgnhJ1

Trong chuỗi ký tự ở trên, chat đợi thông báo "login" từ trạm ở xa, sau đó gửi tên đăng nhập, tiếp theo nó đợi thông báo nhập mật khẩu "password"và gửi mật khẩu đi.

Để quay số thì chúng ta phải có các lệnh modem. Giả sử modem của ta thuộc họ HAYES và số điện thoại là 8203154, ta có lệnh để quay số vào máy chủ ppp như sau:

```
$chat '' ATZ   Ok ATDT82013154 CONNECT '' login: ppp
password: hellwer
```

Theo định nghĩa, xâu đầu tiên là xâu đợi. Nhưng ta chưa kích hoạt modem nên nó chưa thể trả lời, do đó chat bỏ qua xâu này bằng cách đưa vào xâu rỗng. Tiếp theo gửi lệnh khởi động modem ATZ và đợi trả lời OK. Thực hiện quay số bằng ATDT và đợi thông báo trả lời CONNECT. Tiếp theo là xâu rỗng vì ta không muốn gửi một xâu nào đi mà đợi thông báo "login:" và gửi đi tên đăng nhập cùng với mật khẩu của mình.

Nếu không muốn chỉ ra các xâu đợi trong dòng lệnh, ta có thể để dòng ký tự trên vào một tệp tin và chỉ ra tệp tin đó bằng tham số -f. Kết hợp với tuỳ chọn *option* ta có lệnh thiết lập một liên kết *pppd* như sau:

```
$ pppd connect "chat -f dial-c3po" /dev/cua3 38400 -detach \
                crtscts modem defaultroute
```

Ngoài tuỳ chọn *connect* để chỉ ra chương trình quay số, *pppd* còn có tuỳ chọn *detach* để trở thành tiến trình mặt sau, tuỳ chọn modem để *pppd* nhận biết các hoạt động của modem như ngắt đường truyền, trạm ở xa huỷ bỏ cuộc gọi...

Ngoài các thao tác cơ bản trên , chat còn có các tham số nâng cao khác, chẳng hạn ta muốn dừng ngay lệnh quay số khi nhận được các thông báo bận BUSY hay không có tín hiệu NO CARRIER trước khi hết timeout:

```
$ chat -v ABORT BUSY ABORT 'NO CARRIER' '' ATZ OK ...
```

2.6. Gỡ rối cho pppd

Theo mặc định, các thông báo lỗi và cảnh báo của *pppd* sẽ được gửi tới *daemon syslog*. Ta cần sửa đổi tệp */etc/syslog.conf* để daemon này ghi các thông báo này ra một tệp tin, nếu không nó sẽ huỷ bỏ các thông báo đó. Để các thông báo được ghi ra */var/log/ppp-log*, ta thêm dòng sau:

```
daemon.*        /var/log/ppp-log
```

Nếu thiết lập PPP không thành công, ta có thể xem nội dung tệp này để biết được các lỗi có thể xảy ra. Nếu không có những thông tin bổ ích ta có thể dùng *pppd* với tuỳ chọn *debug*. Tuỳ chọn này làm cho *pppd* sẽ gửi tất cả nội dung của các gói tin điều khiển mà nó gửi hay nhận đến *syslog*.

Cuối cùng, ta có thể sử dụng gỡ rối ở mức độ hạt nhân bằng tuỳ chọn kdebug. Có 3 mức độ gỡ rối ở cấp hạt nhân theo giá trị của *kdebug*. Giá trị 1, pppd sẽ hiện các thông báo chung. Giá trị 2, pppd sẽ in tuỳ nội dung của các khung tin HDLC nhận được. Giá trị 3 sẽ in nội dung các khung tin HDLC gửi đi.

2.7. Các tuỳ chọn cho địa chỉ IP

2.7.1. Lựa chọn địa chỉ IP

Trong ví dụ ở phần trước, ta đã thực hiện quay số vào trong p3server và thiết lập một liên kết IP. Trong ví dụ này không có các tuỳ chọn cho địa chỉ IP, vì vậy

các đầu của liên kết sẽ sử dụng địa chỉ IP mặc định của mỗi trạm. Để lựa chọn địa chỉ IP cho liên kết này , ta đưa và tuỳ chọn:

```
local_addr:remote_addr
```

Cả hai tuỳ chọn địa chỉ có thể viết dưới dạng ký pháp thập phân có chấm hoặc tên máy trạm. *local_addr* là địa chỉ của trạm cục bộ, *remote-addr* là địa chỉ của trạm ở xa. Nếu trạm ở xa không chấp nhận địa chỉ IP trong quá trình thương lượng IPCP, liên kết IP sẽ không được thiết lập.

Nếu chỉ có tham số thứ nhất thì IPCP sẽ để cho trạm ở xa tự chọn địa chỉ IP, tương tự ta có thể chọn địa chỉ cho trạm ở xa, còn trạm cục bộ sử dụng địa chỉ ứng với tên trạm này.

Đối với các máy chủ PPP cung cấp dịch vụ quay số vào cho nhiều trạm, quá trình gán địa chỉ IP được thực hiện tự động khi quay số vào. Khi đó ta không được yêu cầu máy chủ này chấp nhận địa chỉ IP tự đưa ra mà phải nhận địa chỉ IP từ máy chủ. Trong trường hợp này ta không được để tuỳ chọn *local_addr* và phải dùng *pppd* với tuỳ chọn *noipdefault*.

2.7.2. Chọn đường qua liên kết PPP

Sau khi cài đặt một giao diện mạng, pppd chỉ thiết lập việc chọn đường đến trạm ở xa. Nếu trạm ở xa là thành viên của một mạng LAN và ta muốn các máy khác trong đó có thể truyền thông được với máy của mình, ta cần thiết lập thêm một số thành phần trong bảng chọn đường.

Khi PPP server hoạt động như một gateway của Internet, ta thiết lập nó thành điểm chọn đường mặc định bằng cách thêm tuỳ chọn *defaultroute* trong lệnh *pppd*. Ngược lại, khi muốn các trạm khác có thể truy cập được vào máy của mình qua PPP server, ta dùng tuỳ chọn *proxyarp* cho *pppd* để thiết lập một thành phần ARP cho máy của mình trên máy chủ.

Mọi việc trở nên phức tạp hơn khi dùng liên kết PPP để kết nối 2 mạng LAN. Trong trường hợp này ta cần thêm các thành phần cụ thể cho các bảng chọn đường vì mỗi mạng đã có chọn đường mặc định. Nếu chọn liên kết PPP là chọn đường mặc định sẽ dẫn đến tình trạng một gói tin không rõ địa chỉ quanh quẩn trong liên kết này.

Ta hãy xét một ví dụ sau: Mạng **comnet** của công ty X có địa chỉ mạng 192.168.0.0. Công ty có một chi nhánh ở xa có một mạng con **subcom** 192.168.3.0 cần kết nối với mạng lớn comnet qua đường điện thoại. PPP server trên

comnet là **p3server**, PPP client trên mạng con là **p3client** có địa chỉ IP là 192.168.3.1.

Khi **p3client** quay số đến **p3server**, nó sẽ chọn **p3server** cho chọn đường mặc định. Trên trạm **p3server** ta cần thiết lập mục chọn đường tới các địa chỉ 192.168.3.x là tới gateway **p3client**. Có thể sửa bảng chọn đường bằng cách thông thường (xem phần cấu hình mạng TCP/IP) hoặc dùng tiện ích ip-up trong */etc/ppp*:

```
ip-up iface device spêd local_addr remote_addr
```

iface là tên giao diện mạng, *device* là thiết bị nối tiếp (/dev/cuax), *speed* là tốc độ truyền, *local_addr* và *remote_addr* là địa chỉ của trạm gọi và trạm được gọi. *ip up* có dạng như sau:

```
#!/bin/sh
case $5 in
191.72.3.1)             # this is p3lient
 route add -net 191.72.3.0 gw 191.72.3.1;;
...
esac
exit 0
```

Tương tự, tiện ích *ip-down* được dùng để loại bỏ các thành phần của bảng chọn đường đã thêm vào ở trên.

Tuy nhiên việc chọn đường vẫn chưa hoàn thành. Nếu như **p3client** và **p3server** là các trạm chọn đường mặc định cho mỗi mạng thì ta không phải làm gì nữa. Nếu không, để tất cả các trạm trên hai mạng có thể liên lạc được với nhau, ta cần sửa đổi bảng chọn đường trên tất cả các trạm. Việc này có thể thực hiện bằng cách sử dụng chương trình *gated* trên mỗi trạm. Sau khi sửa đổi bảng chọn đường trên **p3server**, *gated* sẽ lan truyền thông tin chọn đường đến các trạm khác trong mạng.

2.8. Các tuỳ chọn điều khiển liên kết

Như đã giới thiệu ở trước, giao thức LCP dùng để thương lượng các tính chất của liên kết và kiểm tra liên kết PPP.

Tập ký tự điều khiển dị bộ - Asynchronous Control Chasracter Map hay async map, được dùng trong các liên kết không đồng bộ như đường điện thoại để

xác định các ký tự điều khiển đặc biệt cần phải thay thế bằng hai ký tự khác. Ví dụ một số modem sẽ bị lỗi khi nhận được ký tự XON hay XOFF, là các ký tự điều khiển của PPP. Các ký tự chỉ ra trong async map sẽ không được sử dụng riêng lẻ trong PPP.

Tập ký tự điều khiển dị bộ được xác định bằng một số 32 bit, mỗi bit ứng với 32 ký tự từ 0 đến 31 của bảng mã ASCII. Khi khởi tạo, số này là 0xFFFFFFFF, có nghĩa là tập này bao gồm 32 ký tự đó.

Nếu muốn chỉ ra rằng ta chỉ việc tránh sử dụng các ký tự ^S và ^Q (có mã ASCII là 17 và 19), ta dùng tuỳ chọn sau:

```
asyncmap 0x000A0000
```

Tuỳ chọn MRU - Maximum Receive Unit cho biết kích thước khung tin HDLC cực đại mà ta muốn nhận. Cần phân biệt với MTU - Maximum Transfer Unit, là kích thước gói tin cực đại mà giao diện mạng có thể quản lý được. MRU dùng để thông báo cho trạm ở xa biết là không được gửi một khung tin có kích thước lớn hơn giá trị này. Giá trị cực đại của MRU có thể lên đến 1500 byte.

Đối với các ứng dụng tương tác, ta nên lựa chọn MRU nhỏ. Để đặt giá trị MRU bằng 256, ta dùng tuỳ chọn mru 256. Giá trị MRU nhỏ bảo đảm ứng dụng của ta không bị gián đoạn nếu có các ứng dụng khác cũng sử dụng đường truyền. Với các ứng dụng cần tận dụng thông lượng đường truyền, ta nên sử dụng giá trị MRU lớn.

Để kiểm tra hoạt động đường truyền PPP định nghĩa hai thông báo phản hồi là Echo Request và Echo Response. Nếu sau một thời gian không nhận được khung tin nào từ trạm ở xa, *pppd* sẽ phát một Echo Request và đợi Echo Response. Nếu không nhận được thông báo phản hồi, pppd sẽ lặp lại việc gửi Request một số lần trước khi huỷ bỏ liên kết. Tuỳ chọn *lcp-echo-interval* dùng để gán giá trị cho khoảng thời gian trước khi gửi Request, tuỳ chọn *lcp-echo-failure* để xác định số lần lặp lại. Theo mặc định các tuỳ chọn này không được kích hoạt.

2.9. Máy chủ PPP

Để thiết lập một máy chủ PPP, ta chỉ việc chạy chương trình *pppd* với các tuỳ chọn tương ứng. Sau đó thêm một tài khoản để cho phép người dùng quay số vào. Tốt nhất là ta thêm một tài khoản có tên đăng nhập *ppp*, có shell mặc định thay bằng một script để thực hiện *ppp*. Ta thêm tài khoản này vào */etc/passwd* như sau (giá trị uid và gid có thể khác):

ppp:*:500:200:Public PPP Account:/tmp:/etc/ppp/ppplogin

Chương trình shell script có dạng như sau:

```
#!/bin/sh
# ppplogin - script to fire up pppd on login
mesg n
stty -echo
exec pppd -detach silent modem crtscts
```

Lệnh *mesg* ngăn không cho các người sử dụng khác ghi vào terminal đang sử dụng. Lệnh *stty -echo* tắt chế độ phản hồi ký tự, lệnh này là cần thiết vì nếu không, dữ liệu đến từ trạm gọi có thể bị phản hồi trở lại. Tuỳ chọn *-detach* của *pppd* ngăn không cho chương trình này chạy ở mặt sau. Khi *pppd* chạy ở mặt sau, shell script sẽ thoát ra và liên kết sẽ bị treo. Tuỳ chọn *silent* làm cho *pppd* chờ cho đến khi nhận được một gói tin từ trạm gọi trước khi nó bắt đầu gửi. Điều này cho phép không xảy ra timeout khi trạm gọi chưa kịp khởi động PPP client. Tuỳ chọn modem cho phép pppd theo dõi thanh ghi DTR để theo dõi xem liên kết có bị dừng từ phía *client* hay không. Tuỳ chọn *crtscts* kích hoạt chế độ bắt tay khi truyền dữ liệu.

Ngoài các tuỳ chọn trên, *pppd* còn có nhiều tuỳ chọn khác có thể để trong dòng lệnh hay các tệp cấu hình. Để biết thêm chi tiết hãy xem tài liệu *man* của *pppd*.

Chương IV

QUẢN LÝ CÁC DỊCH VỤ MẠNG

I. DỊCH VỤ ÁNH XẠ ĐỊA CHỈ

Mạng máy tính sử dụng giao thức TCP/IP có nhiều cách khác nhau để thực hiện ánh xạ từ tên máy sang địa chỉ IP và ngược lại. Một cách đơn giản nhất là lưu tất cả các thông tin về địa chỉ IP của các trạm vào trong tệp */etc/hosts*. Giải pháp này là thích hợp cho các mạng LAN cỡ nhỏ và chỉ có một người quản trị mạng, đồng thời không thực hiện truyền thông với các mạng khác bên ngoài. Cấu trúc của tệp */etc/hosts* đã được nhắc đến trong chương 2.

Một giải pháp khác là sử dụng dịch vụ BIND - Berkeley Internet Name Domain Service. Thiết lập cấu hình cho BIND là một công việc phức tạp và khó khăn. Tuy nhiên nó thích hợp với việc mở rộng và thay đổi hình trạng mạng. Trong Linux và các hệ thống Unix khác, dịch vụ ánh xạ địa chỉ được cung cấp bởi chương trình *named*. Khi khởi động, *named* sẽ nạp một số tệp dữ liệu chủ yếu vào trong cache của nó và chờ các yêu cầu ánh xạ từ các trạm khác.

Trong chương này ra sẽ xem xét một số vấn đề cốt lõi trong quá trình cài đặt và quản lý hoạt động của một Name Server. Nếu muốn sử dụng BIND trong một môi trường có nhiều mạng LAN hoặc tham gia vào mạng toàn cầu, ta cần phải tham khảo thêm nhiều thông tin chi tiết hơn.

1. Thư viện ánh xạ địa chỉ

Thư viện ánh xạ địa chỉ là một tập hợp các hàm thư viện chuẩn của C có liên quan đến việc chuyển đổi giữa tên máy - địa chỉ IP. Các hàm thường dùng là *gethosthyname()* và *gethosthyaddr()*, dùng để tìm tất cả các địa chỉ IP của một máy trạm và ngược lại. Các hàm trên hoạt động tuỳ thuộc cấu hình hoạt động của thư viện. Chúng có thể tra cứu trong tệp */etc/host*, hoặc gửi các yêu cầu tới hệ thông tin mạng NIS, hoặc tới hệ thống tên miền DNS.

1.1. Tệp tin host.conf

Đây là tệp điều khiển hoạt động của Resolver, nó quy định các dịch vụ sử dụng của resolver và thứ tự sử dụng chúng.

Các tuỳ chọn trong */etc/host.conf* được để trên các dòng cách biệt. Các trường được cách nhau bởi dấu trắng hay dấu Tab, ký tự "#" dùng để chỉ một lời chú thích.

Sau đây là các tuỳ chọn của tệp cấu hình trên:

order — Cho biết thứ tự các dịch vụ được sử dụng. Các giá trị có thể là *bind* để truy vấn Name server, hosts để tra cứu trong */etc/hosts* và *nis* để sử dụng dịch vụ hệ thông tin mạng. Thư tự của chúng cho biết dịch vụ nào sẽ được thử dùng trước, nếu không tìm thấy thì nó sẽ dùng dịch vụ tiếp theo.

multi — Nhận giá trị là *on* hoặc *off*. Cho phép một máy trạm có nhiều địa chỉ IP khi sử dụng */etc/hosts*. Tuỳ chọn này không có ý nghĩa khi sử dụng NIS hoặc DNS.

nospoof — Như đã giải thích trong phần trước, DNS cho phép ánh xạ ngược từ tên máy sang địa chỉ IP bằng miền **in-addr.arpa.** Để tránh việc nhận được một tên máy sai, Resolver có thể được thiết lập để kiểm tra lại xem có đúng địa chỉ IP này là của máy có tên như vậy không, nếu sai nó sẽ thông báo lỗi (spoofing). Tuỳ chọn này nhận các giá trị on hoặc off.

alert — Nhận các giá trị on hoặc off. Nếu đặt on, bất kỳ một thao tác spoof nào cũng sẽ được ghi lại trong tệp *syslog* của hệ thống.

trim

Nhận giá trị là một tên miền. Resolver sẽ loại bỏ giá trị đó này trong tên máy trạm khi trước khi thực hiện. Tuỳ chọn này thích hợp cho việc ánh xạ địa chỉ của các trạm trong mạng nội bộ. Nếu ta không muốn thêm vào tên của miền nội bộ vào trong trên máy trong tệp */etc/hosts*, khi đó phải có tuỳ chọn này thì các ánh xạ địa chỉ của các trạm nội bộ mới thực hiện được.

Ví dụ về tệp */etc/host.conf* của trạm *vlager* trong mạng của trường đại học Groucho Marx như sau:

```
# /etc/host.conf
# Sử dụng dịch vụ DNS, sau đó đến ánh xạ trực tiếp
order   bind hosts
# Cho phép một trạm có nhiều địa chỉ IP
multi   on
# Thiết lập tuỳ chọn nospoof
nospoof on
# Thiết lập tham số cho tuỳ chọn trim
trim    vbrew.com.
```

1.2. Các biến môi trường của thư viện ánh xạ địa chỉ

Các thiết lập trong tệp */etc/hosts* có thể được thực hiện thông qua các biến môi trường. Chú ý rằng các biến môi trường sẽ được ưu tiên hơn.

`RESOLV_HOST_CONF`

Chỉ ra một tệp cấu hình khác thay vì tệp mặc định */etc/host.conf*.

`RESOLV_SERV_ORDER`

Xác định thứ tự sử dụng các dịch vụ thay vì tuỳ chọn order. Các tham số được là host, bind, nis cách nhau bởi dấu trắng, dấu phẩy hoặc dấu tab.

`RESOLV_SPOOF_CHECK`

Có ý nghĩa giống như *nospoof* nhưng các giá trị tuỳ chọn có khác nhau. Giá trị *off* nghĩa là không dùng biện pháp spoofing. *warn* và *warn off* cho sử dụng nhưng không ghi lại kết quả, còn giá trị * sẽ cho ghi lại kết quả vào *syslog*.

RESOLV_MULTI

Nhận giá trị *on* và *off*, có ý nghĩa giống như tuỳ chọn *multi* của */etc/host.conf*.

RESOLV_OVERRIDE_TRIM_DOMAINS

Nhận giá trị là một danh sách các tên miền, thay thế hoàn toàn hoạt động của tuỳ chọn *trim* trong tệp */etc/host.conf*

RESOLV_ADD_TRIM_DOMAINS

Nhận giá trị là một danh sách các tên miền, bổ xung thêm cho tuỳ chọn *trim* trong tệp */etc/host.conf*

1.3. Tệp cấu hình của Name Server - /etc/resolv.conf

Khi thư viện ánh xạ địa chỉ sử dụng dịch vụ BIND để ánh xạ địa chỉ, ta cần chỉ rõ tên của Name Server được sử dụng. Tệp cấu hình chứa thông tin đó là */etc/resolv.conf*. Nếu không có tệp này hoặc nội dung của nó rỗng thì thư viện sẽ xem máy trạm hiện thời chính là Name Server.

Khi sử dụng Name Server trên cùng máy trạm thì việc đặt cấu hình sẽ có những điểm khác so với việc đặt cấu hình để sử dụng dịch vụ do một Name Server có sẵn trên mạng cung cấp.

Sau đây là một số tuỳ chọn quan trọng:

nameserver — Chỉ ra địa chỉ IP của Name Server được sử dụng. Ta có thể chỉ ra nhiều máy chủ một lúc và các hàm thư viện sẽ thử lần lượt các máy chủ này theo thứ tự đưa ra. Hiện thời Linux hỗ trợ ba Name Server. Nếu không đưa ra tuỳ chọn này thì giá trị mặc định là địa chỉ IP của trạm cục bộ.

domain — Đưa ra tên miền sẽ được nối thêm vào tên máy trạm nếu như lần truy vấn đầu tiên của BIND bị lỗi.

search — Cùng với domain, tuỳ chọn này đưa ra danh sách các tên miền sẽ được thử để thêm vào tên máy trạm khi không tìm ra ánh xạ địa chỉ. Nếu không có tuỳ chọn này thì danh sách các tên miền mặc định sẽ được sử dụng, đầu tiên là tên miền cục bộ, sau đó đến miền cha của nó,... cho đến miền gốc. Tên miền cục bộ được đưa vào qua lệnh domain, nếu không Resolver sẽ sử dụng hàm hệ thống *getdomainname()*.

Ví dụ về tệp resolve.conf của mạng Virtual Brewery:

```
# /etc/resolv.conf
# tên miền là vbrew.com
#
# Sử dụng NameServer là vlarger:
nameserver 191.72.1.1
```

Trong tệp trên không có tuỳ chọn search, vì vậy khi tìm địa chỉ của **vale,** Resolver sẽ tìm tên trạm **vale** và bị lỗi, nó sẽ tìm tiếp **vale.vbrew.com** và **vale.com**.

1.4. Ánh xạ địa chỉ trong các mạng lớn

Nếu ta sử dụng một mạng LAN trong một mạng lớn thì ta nên sử dụng ngay dịch vụ của Name Server của mạng lớn đó nếu có thể. Ưu điểm của phương pháp này là các Name Server như vậy có thông tin trong cache khá lớn vì các yêu cầu truy vấn địa chỉ đều đi qua nó. Tuy nhiên, nhược điểm của nó là nếu có lỗi đường truyền tới máy chủ này hoặc máy chủ bị hỏng thì các trạm trong mạng cục bộ sẽ không thể liên lạc được với nhau nữa.

Mặc dù khả năng xảy ra lỗi là nhỏ song ta vẫn nên có những biện pháp đề phòng. Cách thứ nhất là sử dụng một Name Server cục bộ để thực hiện các ánh xạ địa chỉ cho các máy trong mạng cục bộ. Tuy nhiên, cách này chỉ dùng được khi ta được quyền sử dụng một miền con riêng.

Cách thứ hai là tạo một bảng các địa chỉ IP của các trạm của mạng cục bộ tại tệp */etc/hosts*. Trong tệp cấu hình */etc/host.conf*, ta để tuỳ chọn *order* là *"bind host"* để Resolver sử dụng */etc/hosts* trong trường hợp các Name Server không hoạt động.

2. Sử dụng chương trình named

Named là chương trình cung cấp dịch vụ chuyển đổi địa chỉ trong hầu hết các hệ thống Unix. Đây là một chương trình server, được phát triển bởi BSD nhằm cung cấp dịch vụ cho các trạm khách và các Name Server khác. Phiên bản hiện thời của chương trình này là BIND-4.9.3.

Để sử dụng được chương trình *named*, ta cần hiểu được cơ chế hoạt động của hệ thống tên miền DNS. Có thể xem lại các kiến thức cơ bản của DNS trong phần trước hoặc các tài liệu có liên quan.

Named thường được kích hoạt khi khởi động hệ thống và hoạt động cho đến khi hệ thống ngừng làm việc. Chương trình này nhận thông tin trong tệp */etc/named* và các tệp dữ liệu chứa ánh xạ địa chỉ - tên miền và ngược lại (hay còn được gọi là tệp thông tin vùng (zones files). Cấu trúc cụ thể của các tệp này sẽ được trình bày ở phần sau.

Để chạy named, ta chỉ cần gõ lệnh sau tại dấu nhắc:

```
$ /usr/sbin/named
```

Khi đó named sẽ khởi động, đọc tệp named.boot và các tệp thông tin vùng. Nó sẽ ghi các tiến trình của nó vào tệp */var/run/named.pid* dưới dạng mã ASCCI, nạp các tệp vùng cần thiết từ Name Server chủ và bắt đầu nghe yêu cầu DNS tại cổng 53.

2.1. Tệp named.boot

Tệp named.boot tệp chứa các thông tin chỉ đến các tệp thống tin vùng và các Name Server khác. Ví dụ sau là tệp *named.boot* của trạm vlager:

```
;
; Tệp /etc/named.boot của trạm vlager.vbrew.com
;
directory      /var/named
;
;              domain                    tệp
;-----------------------------------------------
cache                                    named.ca
primary        vbrew.com                 named.hosts
primary        0.0.127.in-addr.arpa      named.local
primary        72.191.in-addr.arpa       named.rev
```

Các lệnh *cache* và *primary* chỉ ra các thông tin cần nạp của *named*. Các thông tin trên được chứa trong các tệp chỉ ra trong tham số thứ hai. Chúng chứa các thông tin dưới dạng văn bản về các bản ghi nguồn của DNS.

Trong ví dụ trên, ta cấu hình *named* là Name Server chủ của 3 miền bằng cách sử dụng 3 lệnh *primary* như trên. Lệnh cache để cho phép sử dụng cơ chế

cache trong Name Server và chỉ ra tên tệp chứa thông tin đã được ghi lại.

Sau đây là các tuỳ chọn có thể sử dụng trong *named.boot*:

directory

Thư mục chứa các tệp thông tin vùng. Tên của tệp có thể chứa đường dẫn tương đối đến thư mục này. Có thể chỉ ra nhiều thư mục bằng tham số này. Đối với hệ thống tệp Linux chuẩn thì thư mục này nên là */var/named*.

primary

Nhận hai tham số là tên miền và tên tệp. Lệnh này nhằm khai báo Name Server có thẩm quyền đối với miền đó và nạp các thông tin vùng từ tệp đã chỉ ra.

Thường luôn có một lệnh primary trong tệp named.boot tương ứng với ánh xạ địa chỉ ngược của địa chỉ loopback 127.0.0.1

secondary

Lệnh này có 3 tham số là tên miền, danh sách địa chỉ IP và tên tệp, nhằm khai báo các Name Server phụ cho miền được chỉ ra.

Các Name Server phụ cũng nạp các thông tin vùng song nó không nhận từ các tệp vùng mà nhận nó từ Name Server chính. Địa chỉ IP của các Name Server chính phải được đưa ra trong danh sách địa chỉ IP. Named sẽ thử cho đến khi nạp được một cơ sở dữ liệu từ một trong những máy chủ được chỉ ra và ghi lại vào tệp (tham số thứ 3). Nếu không có Name Server nào trả lời, named sẽ lấy dữ liệu từ các tệp này.

Named sẽ cập nhật thông tin vùng trong những khoảng thời gian nhất định. Điều này cho phép đồng bộ hoá thông tin vùng giữa các Name Server.

cache

Lệnh này có hai tham số là tên miền và tên tệp. Tệp chỉ ra chứa các bản ghi loại A và NS trỏ tới Name Server gốc. Miền thường là miền gốc được ký hiệu là "."

Nếu không có tuỳ chọn cache, Name Server sẽ không có sẵn các thông tin về ánh xạ địa chỉ IP của các trạm bên ngoài, dẫn đến nó sẽ phải thực hiện rất nhiều lần tìm kiếm, làm giảm hiệu năng hoạt động của

mạng. Hơn nữa do không có địa chỉ của Name Server gốc nên các ánh xạ địa chỉ của các trạm trên mạng Internet sẽ không thực hiện được. Trừ khi ta sử dụng tuỳ chọn forwarders dưới đây.

forwarder

Tuỳ chọn này nhận giá trị là một danh sách địa chỉ IP. Named sẽ truy vấn các Name Server theo danh sách này nếu tìm kiếm trong cache không thành công. Việc tìm kiếm được tiến hành theo thứ tự cho đến khi một Name Server trả lời.

slave

Tuỳ chọn này cho biết đây là một Name Server phụ (slave), tức là nó không truy vấn chính nó mà chỉ gửi yêu cầu tới các server trong danh sách của tuỳ chọn forwarders.

2.2. Các tệp tin cơ sở dữ liệu DNS

Các tệp tin cơ sở dữ liệu sử dụng bởi named đều có một tên miền tương ứng. Tên miền này được chỉ ra trong lệnh cache và lệnh primary và gọi là tên gốc. Trong các tệp tin này ta được quyền sử dụng tên miền và tên máy *tương đối*. Các tên miền với dấu "." ở cuối được xem như là tên *tuyệt đối*, trái lại nó được xem là tương đối so với tên gốc. Tên gốc có thể ký hiệu là @.

Dữ liệu bên trong một tệp tin được chia thành các bản ghi nguồn RR, là đơn vị nhỏ nhất của cơ sở dữ liệu DNS. Mỗi bản ghi nguồn có một dạng tương ứng. Dạng chung của bản ghi nguồn là:

```
[domain] [ttl] [class] type rdata
```

Các trường được phân tách bởi dấu trắng hoặc dấu tab. Mỗi thành phần có thể bao gồm nhiều dòng và được để trong dấu ngoặc ôm { }.

domain Tên của miền mà phần đó áp dụng. Nếu không có phần tên miền thì RR sẽ áp dụng tên miền của bản ghi trước đó.

ttl Để loại bỏ thông tin sau một thời gian, mỗi bản ghi được gán một thời gian sống - time to live. Đây là thời gian một bản ghi có giá trị từ khi nó được lấy về từ server. Nếu ttl không được chỉ ra, named sẽ sử dụng giá trị ttl nhỏ nhất của các bản ghi SOA trước đó.

class Lớp địa chỉ, IN cho các địa chỉ IP, HS cho các địa chỉ trong lớp Hesood. Đối với mạng TCP/IP, ta luôn sử dụng lớp địa chỉ IN.

type Dạng của bản ghi nguồn. Các dạng thông dụng nhất là A, SOA, PTR và NS.

rdata Dữ liệu của bản ghi. Mỗi một loại bản ghi nguồn có một kiểu dữ liệu riêng tương ứng với nó.

Sau đây là một số các bản ghi thường gặp trong tệp tin DNS:

SOA Là bản ghi mô tả cho một vùng (Start of Authority). Các bản ghi khác tiếp theo bản ghi này trong tệp cơ sở dữ liệu sẽ chứa các thông tin cho các trạm của một vùng. Tất cả các tệp chỉ ra trong lệnh primary đều phải chứa ít nhất một bản ghi SOA. Dữ liệu của nó chứa những trường sau:

origin Tên chính thức của Name Server chính trong miền. Tên này luôn ở dạng tên miền tuyệt đối.

contact Địa chỉ email của nhà quản trị mạng, trong đó thay @ bằng dấu chấm, ví dụ *myname.it.hut.vn*

serial Phiên bản của tệp tin vùng, dưới dạng số thập phân. Khi dữ liệu trong tệp thay đổi thì số này phải được tăng lên.

Số hiệu trên được các Name Server phụ nhận biết xem cơ sở dữ liệu có bị thay đổi hay không. Việc so sánh số hiệu được tiến hành trong những khoảng thời gian nhất định.

refresh Là khoảng thời gian tính bằng giây mà Name Server chính quy định để các Name Server phụ cập nhật thông tin. Do hình trạng mạng ít thay đổi nên thời gian này thường là một hay nhiều ngày.

retry Là khoảng thời gian mà một Name Server phụ liên hệ lại với Name Server chính khi một yêu cầu truy vấn

hay cập nhật thông tin bị lỗi. Thời gian này không nên để quá nhỏ vì một lỗi tạm thời trên server chính có thể gây ra việc sử dụng lãng phí tài nguyên mạng. Giá trị được chọn là một hoặc nửa giờ.

expire Thời gian để một server loại bỏ thông tin về dữ liệu trong vùng nếu như nó không liên hệ được với Name Server chính. Thời gian này thường lớn, theo Craig Hunt khuyên, giá trị đó nên là 42 ngày.

minimum Là thời gian sống mặc định cho các bản ghi không chỉ ra giá trị này. minimum nên nhận giá trị lớn, nhất là trong các mạng LAN mà hình trạng mạng gần như không thay đổi, có thể là một tuần hoặc một tháng. Khi có một bản ghi hay thay đổi, ta có thể gán cho nó một ttl riêng.

A Là bản ghi liên kết một địa chỉ IP với một tên miền. Dữ liệu nguồn chứa địa chỉ ở ký pháp thập phân có chấm.

Với mỗi trạm, nó phải có một và chỉ một bản ghi loại A. Tên trạm chỉ ra trong bản ghi loại A là tên chính thức của trạm đó. Các tên khác của trạm là các bí danh và phải được ánh xạ sang tên chính thức trong các bản ghi CNAME.

NS Là bản ghi trỏ tới các Name Server của các vùng con. Trường dữ liệu của nó chứa tên miền của Name Server, một bản ghi loại A được dùng để chứa địa chỉ IP của nó. Bản ghi này gọi là bản ghi liên kết.

CNAME Liên kết một bí danh với tên chính thức của máy trạm.

PTR Liên kết một tên miền **in-addr.arpa** với một tên máy trạm. Bản ghi này dùng để ánh xạ địc chỉ IP ngược. Tên trạm phải lâ tên chính thức.

MX Là bản ghi khai báo một *mail exchanger* cho một miền. Cú pháp của bản ghi MX là

```
[domain] [ttl] [class] MX preference host
```

host là tên của mail server trong miền, mỗi mail server đều có một số tham chiếu *preference* tới nó. Các trạm chuyển tiếp thư khi muốn chuyển thư cho một miền sẽ thử chuyển cho các trạm có bản ghi cho này đến khi thành công. Server nào có số tham chiếu nhỏ hơn sẽ được thử trước.

HINFO — Bản ghi này cung cấp các thông tin về phần cứng và phần mềm của hệ thống. Cú pháp của nó như sau:

```
[domain] [ttl] [class] HINFO hardware software
```

Các giá trị của trường *hardware* và *software* định danh cho phần cứng và phần mềm của hệ thống. Tên định danh này được quy định trong RFC1340 "Assigned Numbers".

2.3. Ví dụ về cơ sở dữ liệu DNS

Để dễ hiểu, chúng ta hãy lấy ví dụ về cơ sở dữ liệu của một Name Server trong mạng Virtual Brewery như đã mô tả trong các chương trước: **vlager**. Ví dụ này tương đối dễ hiểu cho việc thiết lập DNS cho một mạng LAN. Để có thể sử dụng DNS trong điều kiện phức tạp hơn, có thể tham khảo tài liệu "DNS and BIND" của tác giả Cricket Liu và Paul Albitz.

Dưới đây là một phần của tệp dữ liệu cache: *name.ca*. Tệp này chứa các bản ghi về các Name Server gốc. Có rất nhiều các server như vậy, ta có thể sử dụng công cụ *nslookup* để liệt kê các Name Server gốc đó.

```
;
; /var/named/named.ca          Tệp Cache của miền brewery.
;
; .                  99999999   IN    NS   NS.NIC.DDN.MIL
; NS.NIC.DDN.MIL     99999999   IN    A    26.3.0.103
; .                  99999999   IN    NS   NS.NASA.GOV
; NS.NASA.GOV        99999999   IN    A    128.102.16.10
```

Các ví dụ sau là các tệp tin được chỉ ra trong lệnh *primary*:

Tệp tin *named.na*:

```
;
```

```
; /var/named/named.hosts     Các máy trạm cục bộ của brewery
;                            Miền gốc là vbrew.com
;
@               IN  SOA   vlager.vbrew.com. (
                    janet.vbrew.com.
                              16        ; serial
                              86400     ; refresh: once per day
                              3600      ; retry:   one hour
                              3600000   ; expire:  42 days
                              604800    ; minimum: 1 week
                          )
                IN  NS    vlager.vbrew.com.
;
; Mail server
                IN  MX    10 vlager
;
; Địa chỉ quay ngược
localhost.      IN  A     127.0.0.1
; brewery Ethernet
vlager          IN  A     191.72.1.1
vlager-if1      IN  CNAME vlager
; news server
news            IN  CNAME vlager
vstout          IN  A     191.72.1.2
vale            IN  A     191.72.1.3
; winery Ethernet
vlager-if2      IN  A     191.72.2.1
vbardolino      IN  A     191.72.2.2
vchianti        IN  A     191.72.2.3
vbeaujolais     IN  A     191.72.2.4
```

Tệp tin *named.local*:

```
;
; /var/named/named.local    ánh xạ đfai chỉ ngược của  127.0.0
;                           Origin is 0.0.127.in-addr.arpa.
;
@                   IN  SOA   vlager.vbrew.com. (
                        joe.vbrew.com.
                                1          ; serial
                                360000     ; refresh: 100 hrs
                                3600       ; retry:   one hour
                                3600000    ; expire:  42 days
                                360000     ; minimum: 100 hrs
                        )
                    IN  NS    vlager.vbrew.com.
1                   IN  PTR   localhost.
```

Tệp tin named.rev:

```
;
; /var/named/named.rev      ánh xạ địa chỉ ngược
;                           Origin is 72.191.in-addr.arpa.
;
@                   IN  SOA   vlager.vbrew.com. (
                        joe.vbrew.com.
                             16         ; serial
                             86400      ; refresh: once per day
                             3600       ; retry:   one hour
                             3600000    ; expire:  42 days
                             604800     ; minimum: 1 week
                        )
                    IN  NS    vlager.vbrew.com.
; brewery
1.1                 IN  PTR   vlager.vbrew.com.
```

```
2.1                     IN  PTR   vstout.vbrew.com.
3.1                     IN  PTR   vale.vbrew.com.
; winery
1.2                     IN  PTR   vlager-if1.vbrew.com.
2.2                     IN  PTR   vbardolino.vbrew.com.
3.2                     IN  PTR   vchianti.vbrew.com.
4.2                     IN  PTR   vbeaujolais.vbrew.com.
```

2.4. Kiểm tra cấu hình Name Server

Nslookup là một tiện ích cho phép ta dễ dàng kiểm tra cấu hình của một Name Server. Tiện ích này có thể hoạt động ở chế độ tương tác hoặc chế độ dòng lệnh:

```
$ nslookup     hostname
```

Lệnh trên sẽ truy vấn server chỉ ra trong tệp *resolv.conf* và tìm địa chỉ của *hostname*. Nếu tệp trên có chứa nhiều server thì *nslookup* sẽ chọn ngẫu nhiên một trong số đó.

Trong chế độ tương tác, bên cạnh truy vấn cho một trạm, ta có thể truy vấn bất cứ bản ghi DNS nào, có thể hiển thị toàn bộ thông tin vùng của một miền bất kỳ.

Khi thực hiện lệnh *nslookup* mà không có tham số, nó sẽ hiển thị thông tin về Name Server mà nó sử dụng. Theo mặc định nslookup sẽ tìm kiếm các bản ghi loại A. Ta có thể thay đổi dạng bản ghi được truy vấn bằng lệnh *"set type = Stype"* trong đó *Stype* là một trong những dạng bản ghi nguồn, ví dụ:

```
$ nslookup
Default Name Server:  rs10.hrz.th-darmstadt.de
Address:  130.83.56.60
> sunsite.unc.edu
Name Server:  rs10.hrz.th-darmstadt.de
Address:  130.83.56.60
Non-authoritative answer:
Name:    sunsite.unc.edu
Address:  152.2.22.81
```

Nếu ta thử truy vấn một tên mà không có địa chỉ IP tương ứng, song lại có một bản ghi khác trong cơ sở dữ liệu DNS thì sẽ nhận được thông báo: "No type A records found". Để truy vấn các bản ghi loại khác ta dùng lệnh "*set type*", ví dụ bản ghi SOA cho **unc.edu**:

```
> unc.edu
*** No address (A) records available for unc.edu
Name Server:  rs10.hrz.th-darmstadt.de
Address:  130.83.56.60
> set type=SOA
> unc.edu
Name Server:  rs10.hrz.th-darmstadt.de
Address:  130.83.56.60

Non-authoritative answer:
unc.edu
origin = ns.unc.edu
mail addr = shava.ns.unc.edu
serial = 930408
refresh = 28800 (8 hours)
retry   = 3600 (1 hour)
expire  = 1209600 (14 days)
minimum ttl = 86400 (1 day)
Authoritative answers can be found from:
UNC.EDU nameserver = SAMBA.ACS.UNC.EDU
SAMBA.ACS.UNC.EDU      internet address = 128.109.157.30
```

Tương tự ta có thể truy vấn các bản ghi loại MX. Nếu dùng type = ANY thì lệnh này sẽ hiển thị tất cả các bản ghi nguồn liên kết với tên trạm.

```
> set type=MX
> unc.edu
Non-authoritative answer:
```

```
unc.edu preference = 10, mail exchanger = lambada.oit.unc.edu
lambada.oit.unc.edu      internet address = 152.2.22.80
Authoritative answers can be found from:
UNC.EDU nameserver = SAMBA.ACS.UNC.EDU
SAMBA.ACS.UNC.EDU        internet address = 128.109.157.30
```

Một trong những ứng dụng của nslookup là lấy thông tin hiện thời về các Name Server gốc cho tệp tin *named.ca*. Khi đó ta truy vấn với dạng bản ghi là NS liên kết với miền gốc:

```
> set typ=NS
>
Name Server:  fb0430.mathematik.th-darmstadt.de
Address:  130.83.2.30
Non-authoritative answer:
root)  nameserver = NS.INTERNIC.NET
(root)  nameserver = AOS.ARL.ARMY.MIL
(root)  nameserver = C.NYSER.NET
(root)  nameserver = TERP.UMD.EDU
(root)  nameserver = NS.NASA.GOV
(root)  nameserver = NIC.NORDU.NET
(root)  nameserver = NS.NIC.DDN.MIL

Authoritative answers can be found from:
(root)  nameserver = NS.INTERNIC.NET
(root)  nameserver = AOS.ARL.ARMY.MIL
(root)  nameserver = C.NYSER.NET
(root)  nameserver = TERP.UMD.EDU
(root)  nameserver = NS.NASA.GOV
(root)  nameserver = NIC.NORDU.NET
(root)  nameserver = NS.NIC.DDN.MIL
NS.INTERNIC.NET internet address = 198.41.0.4
```

```
AOS.ARL.ARMY.MIL        internet address = 128.63.4.82
AOS.ARL.ARMY.MIL        internet address = 192.5.25.82
AOS.ARL.ARMY.MIL        internet address = 26.3.0.29
C.NYSER.NET     internet address = 192.33.4.12
TERP.UMD.EDU    internet address = 128.8.10.90
NS.NASA.GOV     internet address = 128.102.16.10
NS.NASA.GOV     internet address = 192.52.195.10
NS.NASA.GOV     internet address = 45.13.10.121
NIC.NORDU.NET   internet address = 192.36.148.17
NS.NIC.DDN.MIL  internet address = 192.112.36.4
```

Để biết thêm chi tiết về cách sử dụng và các tham số của lệnh *nslookup*, hãy gõ help tại dấu nhắc của lệnh.

II. MỘT SỐ TIỆN ÍCH CƠ BẢN

Sau khi đã cài đặt phần cơ sở về giao thức mạng IP và ứng dụng chuyển đổi địa chỉ, chúng ta sẽ cài đặt các dịch vụ cơ bản trên mạng. Trong phần này sẽ đề cập việc thiết lập và quản trị các ứng dụng cơ bản như *inetd* server, các dịch vụ đăng nhập từ xa *rlogin*. Giao diện cho việc gọi thủ tục từ xa (Remote Procedure Call) - cơ sở của hệ thống tệp mạng NFS và hệ thông tin mạng NIS.

1. Inetd super-server

Các ứng dụng thường được cung cấp bởi các chương trình chạy liên tục gọi là daemon. Một chương trình daemon thường mở một *cổng* và nghe liên tục các yêu cầu dịch vụ đến từ các trạm trên mạng. Khi có một yêu cầu dịch vụ, nó sẽ tạo một tiến trình con để chấp nhận kết nối và đáp ứng yêu cầu, trong khi tiến trình cha vẫn tiếp tục duy trì việc lắng nghe trên mạng. Do phải chạy liên tục như vậy mà công việc của các daemon thường chiếm nhiều tài nguyên của hệ thống.

Trong Unix, để quản lý chung các yêu cầu truy cập dịch vụ, hệ điều hành sẽ sử dụng một superserver để "nghe" tất cả các yêu cầu dịch vụ đồng thời bằng mà

hệ thống cung cấp. Khi có một yêu cầu dịch vụ đến, superserver sẽ nhận dạng yêu cầu và gọi một server tương ứng để đáp ứng dịch vụ.

Superserver thường sử dụng có tên là *inetd* hay Internet Daemon. Inetd bắt đầu thực hiện khi hệ thống khởi động và nhận danh sách các dịch vụ từ tệp tin cấu hình dịch vụ */etc/inetd.conf*. Ngoài nhiệm vụ thực hiện "nghe" trên mạng, nó còn quản lý các dịch vụ nội bộ của hệ thống như *daytime* để lấy ngày giờ hệ thống, *chargen* để sinh các chuỗi ký tự ...

Tệp cấu hình */etc/inetd.conf* được tạo thành từ các dòng riêng biệt có các thành phần như sau:

```
Service type protocol wait user server cmdline
```

Mỗi trường có ý nghĩa như sau:

service — Tên của dịch vụ. Mỗi dịch vụ tương ứng với một cổng truy cập quy định trong tệp */etc/services*. Nội dung tệp này sẽ được trình bày dưới đây.

type — Kiểu socket mà server chấp nhận. Đối với giao thức hướng liên kết như TCP sẽ dùng kiểu socket là stream. Giao thức UDP là giao thức không liên kết thì có kiểu socket là dgram.

protocol — Tên của giao thức tầng giao vận mà dịch vụ sử dụng. Giao thức này phải được chỉ ra trong tệp */etc/protocol*.

wait — Áp dụng cho socket kiểu dgram. Có thể đặt tuỳ chọn là *wait* hoặc *nowait*. Nếu đặt *wait*, inetd chỉ gọi dịch vụ khi có yêu cầu. Ngược lại chương trình server của dịch vụ sẽ ngay lập tức nghe cổng truy nhập ngay sau khi khởi động.

Tuỳ chọn wait thích hợp cho các server đơn tiến trình. Chúng thường nhận tất cả dữ liệu tới từ mạng rồi kết thúc, chẳng hạn như các server RPC. Các server đa luồng có nhiều tiến trình con chạy đồng thời thường sử dụng tuỳ chọn *nowait*. Tuỳ chọn này là thích hợp cho socket kiểu stream.

user — Là số hiệu đăng nhập của người sử dụng thực hiện chương trình này. Thông thường là root, tuy vậy có một số dịch vụ có thể sử dụng tài khoản khác. Ta không nên thực hiện một chương trình dưới mức đặc quyền cao nhất trừ khi chương trình đó đòi hỏi phải

như vậy.

server — Đường dẫn đầy đủ đến chương trình server của dịch vụ. Các dịch vụ nội bộ được đánh dấu bằng từ khoá internal.

cmdline — Lệnh và các tham số của chương trình server. Đối với các dịch vụ của trạm cục bộ thì trường này để trống.

```
#
# inetd services //các dịch vụ mạng do inetd quản lý
ftp     stream tcp nowait root   /usr/sbin/ftpd    in.ftpd -l
telnet    stream  tcp  nowait  root      /usr/sbin/telnetd
in.telnetd -b/etc/issue
#finger stream tcp nowait bin    /usr/sbin/fingerd in.fingerd
#tftp   dgram  udp wait   nobody /usr/sbin/tftpd   in.tftpd
#tftp      dgram    udp  wait      nobody  /usr/sbin/tftpd
in.tftpd /boot/diskless
login   stream tcp nowait root   /usr/sbin/rlogind in.rlogind
shell   stream tcp nowait root   /usr/sbin/rshd    in.rshd
exec    stream tcp nowait root   /usr/sbin/rexecd  in.rexecd
#
#       inetd internal services //Các dịch vụ trong hệ thống
của inetd
#
daytime   stream tcp nowait root internal
daytime   dgram  udp nowait root internal
time      stream tcp nowait root internal
time      dgram  udp nowait root internal
echo      stream tcp nowait root internal
echo      dgram  udp nowait root internal
discard   stream tcp nowait root internal
discard   dgram  udp nowait root internal
chargen   stream tcp nowait root internal
chargen   dgram  udp nowait root internal
```

2. Chương trình điều khiển truy nhập

Để bảo đảm an toàn cho hệ thống cũng như thông tin trong hoàn cảnh máy tính được nối vào mạng, các ứng dụng trên nó cần được thiết lập một cơ chế bảo vệ. Một trong những mối nguy cơ là các chương trình ứng dụng không phân biệt được các yêu cầu truy cập dịch vụ thực sự với các yêu cầu với mục đích xấu. Ví dụ finger là một ứng dụng cung cấp thông tin về người sử dụng trên hệ thống, với những kẻ tin tặc chúng có thể dùng ứng dụng này để lấy những thông tin đó để dùng vào mục đích xấu.

Vấn đề bảo vệ hệ thống là một lĩnh vực rất rộng với nhiều giải pháp kỹ thuật khác nhau, trong phần này chúng ta sẽ xem xét một cơ chế đơn giản của Linux để bảo vệ hệ thống, đó là chương trình tcpd.

Tcpd hoạt động bằng cách chuyển các yêu cầu truy cập dịch vụ đến daemon syslog để kiểm tra xem yêu cầu đó có hợp lệ hay không. Nếu được chấp nhận nó mới chuyển yêu cầu đến các chương trình server để thực hiện. Tcpd không hoạt động với các dịch vụ sử dụng giao thức UDP.

Ví dụ để bảo vệ cho chương trình finger, ta thêm dòng tham số sau vào trong tệp */etc/inetd*:

```
# wrap finger daemon
finger stream tcp nowait root /usr/sbin/tcpd   in.fingerd
```

Việc điều khiển quyền truy cập được thực hiện bằng cách sử dụng các tệp điều khiển */etc/hosts.deny* và */etc/hosts.allow*. Các tệp trên cho biết một trạm được chấp nhận hay từ chối việc sử dụng dịch vụ. Khi *tcpd* nhận được một yêu cầu dịch vụ từ một trạm ở xa, nó sẽ lần lượt kiểm tra các tệp này. Nếu tên trạm có trong tệp */etc/hosts.allow* thì yêu cầu được chấp nhận ngay. Nếu không nó sẽ kiểm tra tiếp tệp */etc/hosts.deny*, nếu có tên trạm thì yêu cầu sẽ bị từ chối. Nếu tên trạm không có cả trong hai tệp thì yêu cầu được chấp nhận.

Các dòng của những tệp điều khiển có dạng sau:

```
servicelist: hostlist [ :shellcmd]
```

Trong đó *servicelist* là danh sách các tên dịch vụ cần điều khiển từ tệp */etc/services*. Có thể dùng từ khoá ALL hoặc ALL EXCEPT, ví dụ "ALL EXCEPT finger, tftp" để chỉ các dịch vụ, trừ finger và tftp.

Hostlist là danh sách tên trạm hoặc địa chỉ IP, hoặc từ khoá ALL, LOCAL, UNKNOWN. Trong đó ALL chỉ tất cả các trạm, UNKOWN chỉ các trạm không

thực hiện được ánh xạ địa chỉ. LOCAL chỉ các trạm mà tên của nó không chứa dấu "." (thường là các trạm cục bộ vì tên trạm cục bộ thường bỏ qua dấu chấm). Với tên có dấu chấm ở đầu sẽ tương đương với tất cả các trạm có tên miền trùng với tên này, ví dụ **.hut.vn** đại diện cho **cntt.hut.vn, dtvt.hut.vn** ...

Để từ chối truy cập đến dịch vụ *finger* và *tftp* của các trạm bên ngoài mạng cục bộ để tệp */etc/hosts.allow* rỗng và thêm dòng sau vào tệp */etc/hosts.deny*:

```
in.tftpd, in.fingerd: ALL EXCEPT LOCAL, .your.domain
```

Trường tuỳ chọn *shellcmd* có thể chứa các lệnh của shell được thực hiện khi một yêu cầu có tên trạm được chỉ ra trong các tệp này. Cơ chế này cho phép đặt các bẫy để tìm thông tin của những kẻ bị tình nghi là tin tặc.

```
in.ftpd: ALL EXCEPT LOCAL, .vbrew.com :
        echo "request from %d@%h" >> /var/log/finger.log;
        if [ %h != "vlager.vbrew.com" ]; then
        finger -l @%h >> /var/log/finger.log
        fi
```

Các tham số %h và %d là tên của trạm truy cập và dịch vụ được truy cập. Để biết thêm chi tiết có thể tham khảo tài liệu của *hosts-access(5)*.

3. Các tệp tin services và protocols

Mỗi một loại dịch vụ thường tương ứng với những cổng truy cập nhất định và được quy định tại RFC1340 "Assigned Numbers". Để các chương trình server và client có thể tìm được cổng truy cập từ tên dịch vụ, danh sách trên sẽ được liệt kê trong tệp */etc/services* trên mỗi máy. Mỗi dịch vụ tương ứng với một hoặc nhiều dòng có dạng sau:

```
service port/protocol   [aliases]
```

service là tên dịch vụ, *port* là số hiệu cổng mà dịch vụ được cung cấp, protocol cho biết giao thức tầng giao vận được sử dụng, thường là tcp hoặc udp. Một dịch vụ có thể sử dụng nhiều giao thức. Aliases là các tên khác tương đương của dịch vụ. Nói chung, ta không bao giờ cần sửa đổi tệp */etc/services* của hệ điều hành Linux. Sau đây là một phần ví dụ của tệp này.

```
# Tệp tin /etc/services
#
```

```
# well-known services
echo            7/tcp       # Echo
echo            7/udp       #
ftp-data       20/tcp       # File Transfer Protocol (Data)
ftp            21/tcp       # File Transfer Protocol (Control)
....
#
# UNIX services
exec          512/tcp          # BSD rexecd
biff          512/udp  comsat # mail notification
login         513/tcp          # remote login
who           513/udp  whod    # remote who and uptime
shell         514/tcp  cmd     # remote command, no passwd
.....
```

Tương tự như tệp tin *services*, các thư viện ứng dụng mạng cần phải chuyển đổi tên giao thức sang số hiệu giao thức được sử dụng bởi tầng IP. Việc chuyển đổi này được thực hiện thông qua tệp */etc/protocols*. Nó chứa thông tin về tên giao thức và số hiệu giao thức tương ứng:

```
#
# Internet (IP) protocols
#
ip      0       IP              # internet protocol
icmp    1       ICMP            # internet control message protocol
igmp    2       IGMP            # internet group multicast protocol
tcp     6       TCP             # transmission control protocol
udp     17      UDP             # user datagram protocol
raw     255     RAW             # RAW IP interface
```

4. Gọi thủ tục từ xa - RPC

Một trong những cơ chế thường dùng cho các ứng dụng khách-chủ là cơ chế gọi thủ tục từ xa - Remote Procedure Call. RPC là một tập hợp các hàm thư viện

và các tiện ích được phát triển bởi Sun Microsystem phục vụ cho gọi thủ tục từ xa. Các ứng dụng quan trọng được xây dựng trên nền của RPC là hệ thống tệp mạng NFS và hệ thông tin mạng NIS.

Một máy chủ RPC chứa tập hợp các thủ tục mà các mấy khách có thể thực hiện bằng cách gửi các yêu cầu RPC và các tham số cho thủ tục. Máy chủ sẽ thực hiện thủ tục này và gửi trả lại kết quả cho máy khách. Để đảm bảo tính độc lập đối với hệ thống, dữ liệu gửi đi được để dưới dạng mở rộng XDR - Extended Data Representation, sau đó bên nhận sẽ chuyển lại về dạng dữ liệu thông thường.

Với sự phát triển của các phiên bản RPC khác nhau, sự không tương thích có thể xảy ra. Do vậy mỗi chương trình RPC đều gắn với một số hiệu phiên bản. Mỗi RPC server có thể đáp ứng được nhiều phiên bản khác nhau đồng thời, nếu số hiệu phiên bản của RPC client trùng với một trong những phiên bản của server thì thủ tục mới được thực hiện.

Mỗi một máy chủ RPC cung cấp một hoặc nhiều tập hợp các thủ tục. Mỗi tập được thực hiện nhờ một chương trình riêng với số hiệu tương ứng. Danh sách các chương trình này được liệt kê trong tệp */etc/rpc*. Một phần của tệp này như sau:

```
#
# /etc/rpc - Các dịch vụ sử dụng RPC
#
portmapper      100000  portmap sunrpc
rstatd          100001  rstat rstat_svc rup perfmeter
rusersd         100002  rusers
nfs             100003  nfsprog
ypserv          100004  ypprog
mountd          100005  mount showmount
```

Các máy chủ RPC cung cấp một cổng truy cập TCP và một cổng UDP cho mỗi chương trình daemon. Các ứng dụng RPC thường sử dụng giao thức UDP và chỉ sử dụng TCP khi dữ liệu không vừa cho một gói tin UDP.

Các chương trình RPC khách không sử dụng tệp cấu hình */etc/inetd* để tìm cổng truy cập dịch vụ vì RPC không sử dụng các cổng cố định. Các cổng mà RPC sử dụng có thể dùng cho nhiều chương trình khác nhau, do vậy nó có thể bị chương trình khác chiếm mất. Cách thức RPC sử dụng cổng là RPC chọn một cổng có thể và đăng ký với *portmapper*, một chương trình quản lý cổng truy cập. Khi muốn

truy cập dịch vụ, ứng dụng RPC khách phải truy vấn portmapper của máy chủ trước để nhận cổng truy cập dịch vụ.

Cũng giống như chương trình *inetd*, nếu *portmapper* không hoạt động thì các ứng dụng RPC sẽ không thể hoạt động được. Khi đó ta cần khởi động lại *portmapper* bằng tay hoặc khởi động lại toàn bộ hệ thống.

Trong Linux, để khởi động *portmapper* ta cần gọi lệnh *rpc.portmap* trong */usr/bin* hoặc gọi tự động từ tệp *rc.inet2*. Chương trình này không cần các cấu hình đặc biệt.

5. Cấu hình để thực hiện lệnh từ xa

Có nhiều lệnh trong Linux có thể thực hiện trên các trạm ở xa như *rlogin*, rsh, *rcp*, *rexec*. Các lệnh này mở một shell trên trạm ở xa và cho phép người dùng thực hiện các lệnh trên đó. Do vậy họ cần có một tài khoản trên trạm cần thực hiện lệnh, trên trạm này sẽ có một cơ chế kiểm tra, thông thường người dùng đưa ra tên đăng nhập và trạm đó sẽ hỏi mật khẩu để xác nhận.

Khi một người dùng thường xuyên sử dụng lệnh trên các máy trạm từ xa, ta có thể bỏ qua việc xác nhận mật khẩu. Có hai cách để thực hiện:

Cách thứ nhất là người dùng root trên một trạm cho phép một hay nhiều người trên một hay nhiều trạm truy cập vào mà không hỏi mật khẩu. Khi đó ta cần thêm các tên máy trạm và tên người dùng vào trong tệp */etc/hosts.equiv*. Hệ thống sẽ xem các người dùng ở xa đó là tương đương với các người dùng trên trạm cục bộ.

Cách thứ hai là một người sử dụng cho phép các người khác đăng nhập vào hệ thống bằng tài khoản của mình. Những người này được liệt kê trong tệp tin .rhosrs trong thư mục home của họ. Để bảo đảm an toàn, tệp này phải thuộc quyền sở hữu của người đó hoặc của root và không được là một liên kết biểu tượng.

Khi một người sử dụng gọi một lệnh trên trạm ở xa, tên của họ sẽ được kiểm tra trong tệp */etc/hosts.equiv* trước, sau đó đến tệp *.rhosts* trong thư mục home của tài khoản mà họ muốn đăng nhập.

Ví dụ: huymc là người dùng trên trạm **may1**, phuongnt là người dùng trên trạm **may2**. huymc muốn đăng nhập từ trạm **may1** vào trạm **may2** bằng lệnh:

```
$ rlogin may2
```

Khi đó **may2** sẽ kiểm tra tệp **/etc/hosts.equiv** để xác định xem huymc có được đăng nhập tự do hay không, nếu sai nó sẽ kiểm tra tiếp tệp *.rhosts* trong thư mục home của phuongnt.

Giả sử tệp */etc/hosts.equiv* của may2 như sau:

```
may1

-may3

may4.hut-it.edu.vn  hungnt
```

Mỗi dòng của tệp trên gồm tên máy trạm và tên người dùng (tuỳ chọn). Nếu không chỉ ra tên người dùng thì mặc định là tất cả người dùng trên trạm đó. Dấu - cho biết người dùng trên trạm đó không được quyền đăng nhập. Trong ví dụ trên huymc sẽ được quyền đăng nhập tự do vào **may2**. Trong trường hợp muốn đăng nhập với tên của phuongnt thì sẽ bị hỏi mật khẩu:

```
$ rlogin -l phuongnt may2
```

Giả sử tệp *.rhosts* trong thư mục home của phuongnt trên **may2** có dạng như sau:

```
may3

may1    huymc
```

Dòng thứ nhất cho phép mọi người trên trạm **may3** có thể đăng nhập tự do vào tài khoản của phuongnt. Dòng thứ hai chỉ cho mình huymc trên trạm may1 được truy nhập tự do vào tài khoản này.

Do việc trao đổi định danh máy được thực hiện thông qua địa chỉ IP, do đó máy chủ RPC sẽ nhận được địa chỉ IP của trạm khách và sử dụng ánh xạ địa chỉ ngược để tìm tên máy trạm. Vì vậy:

- Tên máy trạm trong các tệp */etc/hosts.equiv* và .rhost phải là tên chính thức của trạm đó.

- Nếu tên trạm khách là tên miền đầy đủ (trong trường hợp sử dụng DNS) thì tên máy trạm rút gọt trong các tệp trên sẽ được bổ xung thêm phần tên miền cục bộ.

III. HỆ THÔNG TIN MẠNG NIS

1. Giới thiệu về NIS

Khi sử dụng hệ thống mạng nói chung, mục đích của chúng ta là làm cho môi trường mạng trở nên trong suốt đối với người sử dụng. Một trong những điểm quan trọng là làm cho các dữ liệu quan trọng như thông tin về người sử dụng, về các trạm trong mạng là đồng nhất trên tất cả các trạm làm việc. DNS là một trong những ứng dụng như vậy. Tuy nhiên việc sử dụng DNS là tương đối phức tạp và trong một mạng LAN thì giải pháp "con nhà giàu" này có thể gây khó khăn cho người quản trị mạng.

Chính vì vậy mà Sun Microsystem đã phát triển hệ thông tin mạng NIS - Network Information System. NIS sung cấp các tiện ích truy cập cơ sở dữ liệu để phân phối thông tin, chẳng hạn như dữ liệu trong */etc/passwds* và */etc/groups* cho tất cả các trạm trong mạng. Điều này làm cho mạng trở nên như một hệ thống duy nhất. Tương tự như vậy cho các thông tin về các trạm trong tệp */etc/hosts*.

NIS được xây dựng trên việc gọi thủ tục từ xa RPC. Nó bao gồm một thư viện của máy chủ, thư viện trạm khách và các công cụ quản trị. Ban đầu NIS được gọi là YP -Yellow Pages. Song một sản phẩm khác cũng tên là YP có bản quyền thuộc công ty British Telecom. Do vậy Sun không được sử dụng tên này nhưng vẫn sử dụng yp như một tiền tố cho các tiện ích như *ypserv, ypbind*...

Cùng với sự phát triển của NIS mà xuất hiện sự khác nhau trong các phiên bản. NIS "truyền thống" được xây dựng trên thư viện libc 4/5. NIS+ là mở rộng của NIS song có hỗ trợ bảo mật thông tin. NYS là một phiên bản chuẩn có hỗ trợ cả NIS và NIS+. Cả NIS và NIS+ ban đầu đều không hỗ trợ mật khẩu ẩn (Shadow Password).

Người dùng Linux Redhat phiên bản 6.0 có thể không cần để ý đến sự khác nhau trên. NIS hiện thời trong Linux sử dụng thư viện GNU C Library 2.x (aka libc6) có hỗ trợ cả NIS, NIS+ và mật khẩu ẩn.

2. Hoạt động của NIS

NIS lưu trữ cơ sở dữ liệu về thông tin quản trị mạng trong các tệp tin gọi là

maps. Các tệp tin này được đặt trên một NIS server trung tâm, từ đó các NIS client có thể truy cập đến thông quan các thủ tục từ xa RPC. Maps thường là các tệp dạng DMB, một dạng cơ sở dữ liệu đơn giản dưới Linux được quản lý bởi chương trình gdbm.

Các tệp tin maps được tạo ra từ các tệp văn bản như */etc/hosts* hay */etc/paswds*. Mỗi tệp văn bản có thể tạo ra nhiều tệp maps khác nhau tuỳ thuộc vào khoá của nó, ví dụ nếu khoá là tên trạm thì ta có tệp *hosts.byname*, nếu khoá là địa chỉ IP thì ta có tệp *hosts.byaddr*. Các tệp maps thông dụng là:

Bảng 4.1. Một số tệp cơ sở dữ liệu của NIS

Tệp Master	Tệp map tương ứng	
/etc/hosts	hosts.addr	Hosts.byname
/etc/networks	network.byname	Network.byaddr
/etc/passwd	passwd.byname	Passwd.byid
/etc/group*s*	group*s*.byname	Group*s*.byid
/etc/services	services.byname	Services.bynumber
/etc/rpc	rpc.bynumber	Rpc.byname
/etc/protocol	protocol.byname	Protocol.bynumber
/usr/lib/aliases	mail.aliases	

Mỗi một tệp maps có một tên ngắn hơn để dễ nhớ đối với người sử dụng gọi là các *nickname*. Để hiển thị danh sách các *nickname* ta dùng công cụ *ypcat*:

```
$ ypcat  -x
NIS map nickname translation table:
"hosts" ->     "hosts.byname"
"networks"     ->     "network.byaddr"
"passwd"       ->     "passwd.byname"
"groups"       ->     "groups.byname"
"services"     ->     "services.byname"
  .....
```

Các chương trình máy chủ của NIS thường có tên là *ypserv*. Trong các mạng cỡ nhỏ và vừa nên sử dụng một máy chủ NIS để dễ quản lý. Các mạng lớn thường có nhiều máy chủ NIS, mỗi máy quản lý một phân đoạn mạng để tránh quá tải cho một máy chủ. Các máy chủ này được đồng bộ dữ liệu bằng cách chia làm máy chủ chính, máy chủ phụ. Các tệp tin map chỉ được tạo ra trên các máy chủ chính và phân bố đến tất cả các máy chủ phụ.

Một miền NIS là tập hợp các máy trạm được quản lý bởi một máy chủ NIS. Không có sự đồng nhất giữa miền NIS và miền DNS. Khái niệm miền NIS chỉ có ý nghĩa về quản trị, nó là trong suốt đối với người sử dụng. Do vậy tên miền NIS chỉ có ý nghĩa đối với người quản trị mạng. Để hiển thị và thiết lập tên miền NIS ta sử dụng lệnh *domainname*. Lệnh này khi không có tham số sẽ hiển thị tên miền, có tham số sẽ thiết lập tên miền, ví dụ:

```
$ domainname nis-domain
```

Tên miền NIS sẽ cho biết máy chủ của miền nào các ứng dụng sẽ truy cập để nhận thông tin cần thiết. Để biết được máy chủ nào trong mạng là NIS server, các chương trình ứng dụng phải hỏi *ypbind*, một chương trình daemon có nhiệm vụ phát hiện NIS server trên mạng.

Ypbind phát các gói tin quảng bá để tìm máy chủ NIS. Khi nhận được gói tin trả lời đầu tiên của một máy chủ NIS nào đó, *ypbind* xem như đó là NIS server gần nó nhất và sử dụng cho các yêu cầu NIS lần sau. Sau một khoảng thời gian, hoặc không truy cập được vào máy chủ NIS, *ypbind* sẽ tiến hành dò tìm lại máy chủ NIS.

Nếu không sử dụng các gói tin quảng bá, *ypbind* có thể tìm được máy chủ NIS từ các tệp cấu hình do người quản trị mạng thiết lập. Việc quảng bá có thể làm cho hệ thống dễ bị tấn công phá hoại nếu có một máy sử dụng một NIS server giả để đánh lừa *ypbind*.

3. Cài đặt và cấu hình cho máy chủ NIS

Trong phần này ta sẽ tiến hành cài đặt một máy chủ NIS cho một mạng chưa có một máy chủ NIS nào. Với mạng đã có một hệ thống NIS, ta chỉ có thể cài đặt thêm một máy chủ NIS phụ, nếu không toàn bộ các dịch vụ mạng sẽ không thể hoạt động được.

3.1. Cài đặt máy chủ NIS chính

Trước tiên ta cần cài đặt phần mềm *ypserv*d lên máy tính. Chương trình server của NIS là *ypserv* sẽ được cài vào trong thư mục */usr/bin*. Các tệp dữ liệu map sẽ được để trong thư mục */var/yp/nis-domain* với giả sử **nis-domain** là tên miền của hệ thống NIS. Ngoài ra tên của thư mục */var/yp/nis-domain* còn cho NIS server biết tên miền nó đang phục vụ. Nếu phần mềm *ypserv*d chưa được cài và đang để trên đĩa CD, ta làm như sau:

```
$ rpm -i  /mnt/cdrom/Redhat/RPMS/ypservd*
$ mkdir /var/yp/nis-domain
```

Để tạo các tệp cơ sở dữ liệu map từ các tệp văn bản, ta sử dụng chương trình *makedbm*. Do đó phải đảm bảo rằng chương trình trên đã được cài đặt trên máy. Việc tạo các tệp map được thực hiện tự động thông qua một *makefile*. Trong tệp này đã chứa tất cả các lệnh cần thiết để tạo các tệp tin map bằng *makedbm*. Sau khi cài đặt phần mềm *ypserv*d, tệp trên đã có trong thư mục */var/yp*. Để tạo các tệp map ta dùng lệnh "*make*". Lệnh này nhận tham số tên miền từ lệnh *domainname*. Tóm lại ta thực hiện:

```
$ domainname nis-domain
$ cd /var/yp
$ make
```

Các tệp tin map không tự động cập nhật mỗi khi ta sửa đổi thông tin quản trị mạng. Do vậy khi có sự thay đổi, ta cần thực hiện lại lệnh *make* để cập nhật sự sửa đổi đó. Cần bảo đảm rằng chương trình *ypserv* luôn được thực hiện mỗi khi ta khởi động lại hệ thống. Điều này được thực hiện bằng cách đưa nó vào trong */etc/rc*.

3.2. Cài đặt máy chủ NIS phụ

Trong các mạng với nhiều trạm làm việc, sử dụng một máy chủ NIS có thể làm cho nó bị quá tải. Hoặc khi có sự cố thì toàn bộ hệ thống mạng sẽ không hoạt động được. Việc sử dụng thêm các máy chủ NIS phụ sẽ làm hệ thống hoạt động hiệu quả hơn.

Việc cài đặt máy chủ NIS phụ chỉ khác với máy chủ NIS chính ở chỗ tạo các tệp tin map. Chúng không được tạo bằng *makedbm* mà được lấy về từ máy chủ chính. Trước tiên thiết lập máy chủ phụ như một trạm khách NIS (xem phần sau).

Thêm tên của máy chủ phụ vào tệp */var/yp/servers* trên máy chính và thực hiện lệnh sau trên trạm khách:

```
$ /usr/lib/yp/ypinit -s masterhost
```

Trong đó *masterhost* là tên của máy chủ chính. Để các máy chủ phụ cập nhật dữ liệu từ máy chủ chính nhằm bảo đảm sự đồng bộ, ta sử dụng các lệnh sau trong tệp *crontab*:

```
20   *      *   *   *   /usr/lib/yp/ypxfr_1perhour
40   6      *   *   *   /usr/lib/yp/ypxfr_1perday
55   6,18   *   *   *   /usr/lib/yp/ypxfr_2perday
```

4. Cài đặt trạm khách NIS

Trước tiên cần cài đặt phần mềm *ypbind* lên máy trạm (nếu ta chưa cài ypbind lúc cài đặt hệ thống) :

```
$ rpm -i /mnt/cdrom/Redhat/RPMS/ypbind*
```

Bước tiếp theo là phải chỉ ra tên của máy chủ và tên miền NIS mà trạm khách sử dụng trong tệp */etc/yp.conf*. Một ví dụ đơn giản cho tệp tin trên trong miền **nis-domain** và máy chủ là **ln-server**:

```
# yp.conf  -YP configuration for NYS library.
#
domainname     nis-domain
server  ln-server
```

Dòng lệnh đầu tiên cho biết trạm khác thuộcvào miền NIS tên là **nis-domain**. Nếu không có dòng lệnh này thì ta có thể chỉ ra tên miền NIS bằng lệnh *domainname* tại dấu nhắc lệnh. Dòng lệnh thứ hai chỉ ra tên của máy chủ được sử dụng. Địa chỉ IP của máy chủ phải được chỉ ra trong tệp */etc/hosts*. Hoặc ta có thể sử dụng trực tiếp địa chỉ IP trong dòng lệnh này.

Khi ta sử dụng máy tính thường xuyên phải thay đổi miền NIS, ví dụ như máy tính xách tay, ta có thể chỉ ra nhiều miền NIS và các máy chủ tương ứng với nó bằng lệnh server. Tệp cấu hình như ví dụ dưới đây cho phép thực hiện điều đó:

```
# yp.conf  - Cấu hình của máy Nis client
#
```

```
server ln-server1 domainname1
server ln-server2 domainname2
```

Khi muốn mang máy tính đến sử dụng trong một mạng tương ứng với miền nào, ta sẽ dùng lệnh *domainname* tại dấu nhắc lệnh để thiết lập tên miền đó.

Sau khi đã tạo ra các tệp cấu hình cơ bản, ta nên kiểm tra xem chương trình *ypbind* đã hoạt động tốt hay chưa. Trước hết khởi động *ypbind*, sau đó dùng tiện ích *ypcat* để lấy thông tin quản lý bởi NIS server. Để xem thông tin về địa chỉ IP của các trạm ta làm như sau:

```
$ ypbind
$ ypcat hosts.byname
192.168.50.1          may1
192.168.50.2          may2
192.168.50.3          may3
192.168.50.20         may20
192.168.50.4          may4
... ...
```

Nếu lệnh *ypcat* không cho kết quả như trên hoặc nhận được thông báo "Can't bind to servers domain", có nghĩa là hệ thống NIS hoạt động chưa tốt. Kiểm tra lại xem tên miền và tên của server đặt trong *yp.conf* có trùng với tên ứng với miền và máy chủ NIS hay không. Kết hợp với kiểm tra liên kết tới máy chủ bằng lệnh *ping*. Nếu máy chủ đã hoạt động, kiểm tra lại sự hoạt động của *ypserv* bằng lệnh *rpcinfo*:

```
$ rpcinfo     -u     serverhost  ypserv
program 10004  version 2 ready and waiting
```

Trong đó *serverhost* là tên máy chủ NIS, nếu nhận được thông báo như trên có nghĩa là *ypserv* đã hoạt động tốt.

Cuối cùng, ta nên cho chương trình *ypbind* tự thực hiện mỗi khi hệ thống khởi động bằng cách thêm vào trong thư mục */etc/rc*.

5. Lựa chọn các tệp tin map

Thực chất, NIS cung cấp các thông tin giống như trong các tệp cấu hình của một trạm cục bộ như */etc/hosts*, */etc/passwd*.... Khi sử dụng NIS ta cần lựa chọn những tệp cấu hình nào của trạm cục bộ cần được thay thế bởi NIS. Thông thường

NIS được sử dụng để tra các thông tin về máy trạm và tài khoản của người dùng. Các thông tin rất này hữu dụng trong trường hợp mạng không sử dụng hệ thống tên miền DNS, đồng thời cho phép người sử dụng đăng nhập vào bất kỳ một trạm làm việc nào trong hệ thống NIS. Khi đó ta cần sử dụng một thư mục */home* chung và chia sẻ cho tất cả các trạm bằng hệ thống tệp mạng NFS. Một tệp tin map khác là *service.byname* được sử dụng khi ta cài đặt thêm các dịch vụ mạng có tên không thuộc tệp chuẩn services, khi đó ta không phải sửa lại các tệp *services* trên tất cả các trạm.

Mặc dù đã sử dụng NIS như một hệ quản trị tập trung, hệ thống này vẫn cho phép các trạm làm việc được quyền tự do lựa chọn hoặc sử dụng cấu hình cục bộ, hoặc sử dụng cấu hình từ NIS server.... Thứ tự sử dụng được chỉ ra trong tệp */etc/nsswitch.conf*, còn được gọi là Name Service Switch.

Thứ tự sử dụng dịch vụ tuỳ thuộc vào từng dạng dữ liệu cụ thể. Đối với thông tin về tên của dịch vụ (services), tệp tin services.byname không có thành phần khác với tệp tin */etc/services*. Nó chỉ có thể chứa nhiều thông tin hơn khi ta sử dụng một số dịch vụ mới mà hệ thống ban đầu không có. Do vậy thứ tự sử dụng là tệp tin cục bộ trước rồi mới đến NIS. Trái lại tệp tin */etc/hosts* chứa dữ liệu về địa chỉ của các trạm là thông tin thường thay đổi, vậy nên sử dụng NIS hay DNS trước, sau đó đến tệp tin cục bộ phòng khi NIS hay DNS không hoạt động.

Ví dụ sau đây cho biết thứ tự sử dụng dịch vụ của các hàm *gethostbyname*(), *gethostbyaddr()* và *getservbyname*(). Các dịch vụ liệt kê trước sẽ được sử dụng, nếu chưa thành công sẽ tiếp tục dịch vụ sau đó.

```
# Ví dụ  /etc/nsswitch.conf
#
hosts:              nis dns files
services:  files  nis
```

Dưới đây là danh sách các dịch vụ có thể sử dụng trong tệp tin */etc/nsswitch.conf*. Các tệp tin, chương trình cụ thể được sử dụng phụ thuộc vào từng loại dịch vụ:

nisplus hay nis+

Sử dụng NIS+ server cho miền NIS hiện thời. Tên của server được chỉ

ra trong tệp tin */etc/nis.conf*.

nis

Sử dụng NIS server cho domain hiện thời. Tên của server được chỉ ra trong tệp tin */etc/yp.conf*. Với thành phần hosts, các tệp map là *hosts.byname* và *hosts.byaddr* sẽđược sử dụng.

dns

Sử dụng DNS server. Dịch vụ này được sử dụng cho mình thành phần hosts. Tên của server truy vấn được đặt trong tệp tin */etc/resolv.conf*.

files

Sử dụng các tệp cấu hình cục bộ, ví dụ */etc/passwd* cho thành phần passwd.

dbm

Tìm thông tin trong các tệp tin cơ sở dữ liệu */var/dbm*. Tên của các tệp tin là các tệp tin map tương ứng của dịch vụ NIS.

Các thành phần được hỗ trợ hiện thời của NYS là: hosts, networks, passwd, group, shadow, gshadow, services, protocols, rpc, and ethers.

Nếu có từ khoá [NOTFOUND=return] trong các thành phần của tệp tin *nsswitch.conf*, NIS sẽ thoát ra ngay mà không sử dụng tiếp các dịch vụ sau trong trường hợp nó không tìm thấy thông tin ở dịch vụ trước đó. Chỉ khi nào dịch vụ trước bị lỗi, NIS mới dùng tiếp dịch vụ sau. Trong ví dụ dưới NIS chỉ sử dụng các tệp tin cục bộ khi khởi động hoặc DNS, NIS server bị hỏng.

```
# /etc/nsswitch.conf
#
hosts:  nis dns [ NOTFOUND=return] files
network:        nis [ NOTFOUND=return]  files
services:       files nis
protocol:       files nis
rpc:            files nis
```

6. Sử dụng các tệp map passwd và group

Một trong những ứng dụng chính của NIS là đồng bộ thông tin về các tài khoản của người sử dụng trên tất cả các trạm trong miền NIS. Khi đó thông tin về người dùng trên trạm được liệt kê một phần nhỏ trong */etc/passwd*, phần còn lại trong tệp tin map *passwd.byname*. Việc chọn nis trong tệp */etc/nsswitch.conf* chưa đủ để NIS có thể hoạt động đúng.

Khi sử dụng tài khoản của người dùng được cung cấp bởi NIS, trước tiên phải bảo đảm số hiệu của người dùng trong tệp passwd phải trùng với số hiệu của người dùng đó trên NIS. Nếu số hiệu người dùng, số hiệu nhóm của người đó khác với thông tin của trên NIS, ta cần sửa lại cho trùng nhau.

Trước tiên thay đổi số hiệu người dùng (uid) và số hiệu nhóm (gid) trong tệp */etc/passwd* và */etc/group* trên trạm cục bộ sang các giá trị mới của NIS. Sau đó đổi quyền sở hữu của tất cả các tệp tin bằng cách thay số hiệu uid và gid cũ sang uid và gid mới. Giả sử người dùng anhnv có số hiệu uid là 501, thuộc nhóm sinhvien có gid là 423, ta sửa đổi quyền sở hữu như sau:

```
# find / -uid 501 -print > /tmp/uid.501
# find / -gid 423 -print > /tmp/gid.423
# cat /tmp/uid.501 | xargs chown  anhnv
# cat /tmp/gid.423 | xargs chgrp sinhvien
```

Sau khi thực hiện các công việc trên, số hiệu uid và gid của một người dùng trên máy trạm sẽ đồng nhất với NIS. Bước tiếp theo là sửa đổi tệp */etc/nsswitch.conf* như sau:

```
# /etc/nsswitch.conf
passwd: nis files
group:  nis files
```

Tệp trên quy định lệnh *login* và các lệnh thuộc họ này sẽ truy vấn NIS server khi một người dùng muốn đăng nhập, nếu không tìm thấy nó sẽ tìm tiếp đến các tệp cục bộ. Thông thường ta loại bỏ hầu hết người dùng khỏi các tệp cục bộ, chỉ giữ lại root hoặc các tài khoản chung như mail, news... cho một số tác vụ cần chuyển đổi số hiệu uid sang tên và ngược lại. Ví dụ chương trình quản lý công việc cron sử dụng lệnh *su* để tạm trở thành news. Nếu news không có trong */etc/passwd*, chương trình trên sẽ không thực hiện được.

Khi người dùng muốn thay đổi mật khẩu, họ không thể dùng lệnh passwd

như khi chưa có NIS. Lệnh *passwd* chỉ có tác dụng sửa đổi các tệp cấu hình cục bộ. NIS cung cấp một công cụ là *yppasswd*, nó không những cho phép sửa mật khẩu người dùng trên NIS mà còn thay đổi các thuộc tính khác như shell... Chương trình này được thực hiện khi khởi động hệ thống bằng cách thêm *rpc.yppasswdd* vào trong rc.inet2.

Vì thói quen mà người sử dụng có thể gõ lệnh *passwd* khi muốn đổi mật khẩu. Giải pháp ở đây là thay *passwd* bằng một liên kết đến *yppasswd*:

```
# cd /bin
# mv passwd passwd.old
# ln yppasswd passwd
```

IV. HỆ THỐNG TỆP MẠNG NFS

1. Giới thiệu về NFS

Hệ thống tệp mạng - Network File System, là một dịch vụ mạng được dùng khá phổ biến. NFS được xây dựng dựa trên việc gọi thủ tục từ xa, cho phép sử dụng các tệp tin trên các trạm ở xa giống như chúng ở trên trạm cục bộ. NFS bao gồm chương trình client trên máy cần sử dụng các tệp tin và chương trình server cung cấp các tệp tin. NFS có thể được sử dụng trên nhiều hệ thống khác nhau, việc truy cập các tệp tin là trong suốt đối với người sử dụng.

NFS có các ưu điểm sau:

- Dữ liệu dành cho người sử dụng có thể đặt trên một trạm tập trung, các trạm khác có thể thiết lập thư mục này lúc khởi động. Một ví dụ điển hình là khi dùng NIS, tất cả tài khoản của người sử dụng được lưu trên máy chủ, thư mục */home* trên máy chủ được thiết lập trên các trạm lúc khởi động. Khi đó người dùng có thể đăng nhập vào bất kỳ trạm nào nhưng thư mục home của họ không thay đổi.
- Dữ liệu chiếm nhiều dung lượng đĩa có thể được giữ tại một trạm và phân phối cho các trạm khác.
- Thông tin quản trị mạng được tập trung tại một trạm. Không cần phải dùng rpc để thiết lập các tệp cấu hình giống hệt nhau trên tất cả các trạm.

NFS trên Linux được viết chủ yếu bởi Rick Sladkey, là người viết mã cho hạt nhân và mã nguồn server của NFS. Chương trình NFS server được kế thừa từ *NFS server unfsd* của tác giả Mark Shand và *hnfsd* của tác giả Donald Becker.

2. NFS hoạt động như thế nào

Một chương trình client có thể thiết lập một thư mục của một trạm ở xa trên một thư mục của mình bằng lệnh mount giống như đối với các thiết bị lưu trữ. Chỉ lưu ý cách thức để chỉ ra thư mục trên một trạm ở xa là *hostname:/dirname*, ví dụ trên trạm may1 ta thực hiện:

```
$ mount    -t nfs  may2:/home  /users
```

Lệnh mount sẽ kết nối với chương trình *mountd* trên trạm **may2** thông qua các thủ tục RPC. Mountd kiểm tra xem thư mục */home* có được cho phép thiết lập từ xa hay không, **may1** có được quyền thiết lập thư mục này không. Nếu có *mountd* sẽ trả về một tệp handle để sử dụng cho các yêu cầu truy cập đến thư mục */users* trên **may1**.

Khi người sử dụng truy cập tệp tin trên NFS, một lời gọi RPC sẽ được gửi tới *nfsd* (NFS daemon) trên máy chủ. Tham số của thủ tục này là tệp *handle*, tên tệp cần truy cập, số hiệu người sử dụng và số hiệu nhóm. Các tham số này được để xác định quyền truy cập của người dùng. Để tránh việc sửa và đọc các tệp tin trái phép, số hiệu người dùng và số hiệu nhóm phải trùng nhau trên cả máy chủ và máy khách.

3. Chuẩn bị cho NFS

Trước khi sử dụng NFS cho dù là máy chủ hay máy khách thì hạt nhân hệ thống phải hỗ trợ NFS. Để kiểm tra ta có thể truy cập vào hạt nhân thông qua giao diện */proc/filesystem*:

```
$ cat /proc/filesystems
minix
ext2
msdos
nodev   proc
nodev   nfs
```

Nếu nfs không được liệt kê trong danh sách hệ thống tệp, ta cần biên dịch lại hạt nhân với tuỳ chọn là có hỗ trợ NFS. Xem lại phần " Cấu hình cho hạt nhân" để biết thêm chi tiết.

Với Linux có hạt nhân phiên bản từ 1.1, ta có thể dùng một cách khác để kiểm tra xem hạt nhân đã hỗ trợ NFS hay chưa. Tạo một thư mục trong */tmp* và thử thiết lập trên trạm cục bộ:

```
$ mkdir /tmp/test
$ mount localhost:/etc /tmp/test
```

Nếu nhận được thông báo: "fs type nfs no supported by kernel" chứng tỏ hạt nhân chưa hỗ trợ NFS. Các thông báo khác cho biết ta chưa cấu hình đúng cho các chương trình NFS trên hệ thống.

4. Thiết lập một thư mục NFS

Một thư mục trên NFS được thiết lập như các hệ thống tệp bình thường. Cú pháp lệnh mount như sau:

```
$ mount -t nfs nfs_volume local_dir option
```

nfs_volume có dạng *remote_host:remote_dir*. Do ký pháp trên chỉ dùng trong NFS nên có thể không cần dùng cả tuỳ chọn *-t nfs*.

Có nhiều tuỳ chọn khi sử dụng lệnh mount để thiết lập một thư mục trên NFS. Các tuỳ chọn có thể đưa trên dòng lệnh bằng tham số -o, hoặc trong trường *option* của */etc/fstab*. Trong cả hai trường hợp, các tuỳ chọn được viết cách nhau bằng dấu phẩy. Tuỳ chọn trong dòng lệnh luôn phủ quyết tuỳ chọn trong */etc/fstab*.

Ví dụ một thành phần của */etc/fstab* như sau:

```
# volume                mount point       type  options
news:/usr/spool/news    /usr/spool/news   nfs   timeo=14,intr
```

Khi đó ta sử dụng lệnh sau để thiết lập thư mục NFS:

```
# mount news:/usr/spool/news
```

Rõ ràng việc sử dụng */etc/fstab* cho phép ta tiết kiệm được công sức gõ lệnh mount khi có nhiều tuỳ chọn. Danh sách đầy đủ của các tuỳ chọn được tham khảo tại tài liệu nfs. Sau đây là một số tuỳ chọn thường dùng:

rsize=n, wsize=n

Kích thước của gói tin sử dụng bởi NFS client khi đọc và ghi lên thư mục NFS. Mặc định là 1024 bytes, là giới hạn của một gói tin UDP.

timeo=n

Thời gian tính theo giây mà NFS client chờ một yêu cầu hoàn thành. Giá trị mặc định là 0,7s.

hard

Thiết lập cứng một thư mục NFS (hard-mounted). Đây là tuỳ chọn mặc định.

soft

Thiết lập mềm một thư mục (soft-mounted).

intr

Cho phép dùng tín hiệu ngắt tới gọi NFS. Tuỳ chọn này được sử dụng khi máy chủ NFS không trả lời.

Trừ các tuỳ chọn *rsize* và *wsize*, các tuỳ chọn khác đều dùng trong trường hợp máy chủ tạm thời không trả lời. Hoạt động của chúng như sau: Khi một client gửi một yêu cầu đến NFS server, nó sẽ chờ một khoảng thời gian timeout. Nếu không có trả lời (minor timeout), thời gian này sẽ được nhân lên gấp đôi cho đến khi vượt quá 60s (major timeout).

Khi một minor timeout xảy ra, client sẽ hiển thị một thông báo lên màn hình và tiến hành yêu cầu lại với gía trị timeout tăng gấp đôi. Nếu quá trình cứ phải tiếp tục diễn ra như vậy cho đến khi thành công mới thôi thì quá trình này gọi là thiết lập cứng. Nếu quá trình phải dừng sau major timeout thì gọi là thiết lập mềm. Khi ghi lên một thư mục NFS, dữ liệu được đưa vào vùng đệm đến khi đầy mới ghi hoặc ghi ở lần gọi sau (write-behind) nên các chương trình sử dụng thiết lập mềm sẽ không biết được quá trình ghi có thành công hay không.

Việc lựa chọn thiết lập mềm hay thiết lập cứng là tuỳ thuộc vào dữ liệu của chương trình nào được sử dụng. Các dữ liệu chương trình X quan trọng với nhiều cửa sổ, thực hiện chậm ta nên thiết lập cứng để bảo đảm trạm khách chờ cho đến khi dữ liệu đó được thiết lập. Các dữ liệu không quan trọng của FTP, News có thể thiết lập mềm để tránh hệ thống bị treo khi một số máy chủ đó tạm thời không

hoạt động. Khi kết nối với máy chủ qua router, ta phải tăng thời gian timeout, đặt thiết lập cứng và cho phép dùng ngắt để huỷ bỏ yêu cầu thiết lập.

Để hiển thị thông tin hiện thời về các thư mục đang được thiết lập, ta có thể dùng tiện ích *showmount*. Tiện ích này được phân phối kèm theo phần mềm NFS server.

5. Các chương trình của dịch vụ NFS

Để một trạm có thể cung cấp dịch vụ NFS cho các trạm khác, ta cần chạy các daemon như *moutd* và *nfsd* trên trạm đó. Cũng như các chương trình RPC khác, chúng không được quản lý bởi *ined*. Các daemon này được khởi động khi boot hệ thống và phải đăng ký với *portmapper*. Do vậy chúng chỉ được thực hiện khi nào *rpc.portmap* đã thực hiện. Để khởi động chúng tự động ta thêm chúng vào *rc.inet2* như sau:

```
if [  -x /usr/sbin/rpc.mountd ]; then
    /usr/sbin/rpc.mountd; echo -n " mountd"
fi
if [  -x /usr/sbin/rpc.nfsd ]; then
        /usr/sbin/rpc.nfsd; echo -n " nfsd"
fi
```

Thông tin về quyền sở hữu tệp mà các NFS daemon gửi cho các client của nó bao gồm số hiệu người dùng và số hiệu nhóm. Nếu máy khách và máy chủ có cùng thông tin về người dùng thì chúng được gọi là chia sẻ cùng không gian uid/gid. Đây là trường hợp ta dùng NIS trong một mạng LAN.

Nếu không sử dụng NIS, ta có thể dùng chương trình ugidd để chuyển đổi không gian uid/gid. Khi có tuỳ chọn map_daemon (sẽ giải thích sau), chương trình này thực hiện trên tạm khách sẽ chuyển uid/gid của máy chủ sang uid/gid của máy khách.

Ugidd là một chương trình RPC, nó cũng được thực hiện từ rc.inet2 như *nfsd* và *mountd*:

```
if [  -x /usr/sbin/rpc.ugidd ]; then
       /usr/sbin/rpc.ugidd; echo -n " ugidd"
fi
```

6. Tệp tin exports

Các tuỳ chọn ở trên được áp dụng cho các trạm khách sử dụng NFS, Ngoài ra còn có các tuỳ chọn khác trên máy chủ để định cấu hình cho dịch vụ mà nó cung cấp. Các tuỳ chọn này được để trong tệp */etc/exports*.

Mặc định, *mountd* không cho phép thiết lập bất kỳ thư mục nào trên trạm. Để cho phép ai đó được quyền thiết lập một thư mục, thư mục đó phải được xuất ra (exported), có nghĩa là liệt kê trong tệp */etc/exports*. Một ví dụ của *exports* như sau:

```
# tệp tin exports của vlager
/home              vale(rw) vstout(rw) vlight(rw)
/usr/X386          vale(ro) vstout(ro) vlight(ro)
/usr/TeX           vale(ro) vstout(ro) vlight(ro)
/                  vale(rw,no root squash)
/home/ftp          *(ro)
```

Mỗi dòng định nghĩa một thư mục và tên các trạm mà nó cho phép thiết lập thư mục đó. Tên trạm có thể là tên rút gọn hoặc tên đầy đủ, có thể sử dụng các ký tự thay thế như * , ? , ... giống như trong Bourne Shell. Nếu một thư mục không có tên trạm kèm theo thì tất cả các trạm đều có quyền thiết lập nó.

Khi kiểm tra tên máy trạm trong tệp tin exports, chương trình *mountd* sẽ tìm tên tương ứng bằng hàm *gethosthyaddr*(). Với DNS, hàm này trả về tên chính thức của máy trạm, do vậy không được sử dụng tên bí danh trong *exports*. Với NIS và */etc/hosts*, hàm trả về tên ứng với địa chỉ đầu tiên mà nó tìm thấy.

Các cờ trong dấu ngoặc đơn có ý nghĩa như sau:

insecure

Cho phép truy cập không cần xác nhận.

unix-rpc

Yêu cầu xác nhận lời gọi RPC của UNIX-domain. Có nghĩa là lời gọi phải sử dụng cổng riêng (reserved port), là cổng có số hiệu nhỏ hơn 1024.

secure

Yêu cầu xác nhận an toàn cho RPC.

Kerberos

Yêu cầu chứng thực Kerberos khi truy cập.

root *squash*

Phủ định các quyền ở mức cao của superuser của trạm khác trên thư mục này. Uid của yêu cầu là 0 sẽ bị chuyển sang uid là 65534. Đây là uid của người dùng nobody.

*no*_root *squash*

Không phủ quyết quyền của superuser. Đây là tuỳ chọn mặc định.

ro

Thiết lập sang một hệ thống tệp chỉ đọc.Đây là tuỳ chọn mặc định.

rw

Thiết lập hệ thống tệp ghi-đọc.

link-relative

Chuyển các liên kết logic tuyệt đối sang liên kết logic tương đối. Tuỳ chọn này cho phép các liên kết logic vẫn trỏ đúng đến tệp tin hay thư mục mà nó cần chỉ đến, nó được dùng khi thiết lập toàn bộ hoặc một cây thư mục của hệ thống tệp. Tuỳ chọn này được sử dụng mặc định.

link-absolute

Để nguyên các liên kết logic như cũ.

map_identify

Cho biết máy chủ và trạm khách sử dụng cùng một không gian các số hiệu uid/gid.

*map*_daemon

Cho máy chủ biết rằng nó và trạm khách không sử dụng cùng không gian số hiệu uid/gid. Nfsd sẽ xây dựng một danh sách ánh xạ các id giữa máy chủ và máy khách bằng cách truy vấn ugidd trên trạm khách.

Khi có một lỗi trong quá trình duyệt tệp */etc/exports*, lỗi đó sẽ được chuyển cho chương trình *syslogd* sau khi mountd và *nfsd* đã khởi động.

V. CHIA SẺ TÀI NGUYÊN VỚI WINDOWS

Samba là một tập các tiện ích dùng để chia sẻ tài nguyên như tệp tin, máy in giữa các máy tính dùng hệ điều hành UNIX và các máy tính dùng hệ điều hành Window98/NT. Giao thức truyền tin sử dụng trong Samba là SMB - Server Message Block hay còn được gọi là Session Message Block. Với Samba chúng ta có thể:

- Chia sẻ các tệp tin Linux cho Windows.
- Chia sẻ máy in Linux cho Windows.
- Chia sẻ các tệp tin Windows cho Linux.
- Chia sẻ máy in Windows cho Linux.

Trong phần này chúng ta sẽ đề cập đến vấn đề chia sẻ tệp tin giữa Linux và Windows. Phần chia sẻ máy in có thể tham khảo tại tài liệu Samba Howto đi kèm theo hệ điều hành Linux.

1. Cài đặt Samba

Để có thể sử dụng Samba cần có giao thức TCP/IP trên cả hai hệ điều hành Linux và Windows, nó không hoạt động với các giao thức khác. Cả hai hệ này đều hỗ trợ TCP/IP như một giao tác khá phổ biến.

Hạn chế của Samba là chỉ hoạt động trong một mạng LAN, không có hỗ trợ khi ta muốn chia sẻ tài nguyên qua router. Nếu muốn sử dụng Samba qua router chúng ta cần thiết lập một IP tunnel, việc này nằm ngoài phạm vi đề cập của tài liệu này.

Việc cài đặt samba rất đơn giản. Tuỳ thuộc và ứng dụng samba là khách hay chủ mà cần cài các phần mềm tương ứng. Phiên bản mới nhất là samba.2.0.5a và samba-client.2.0a. Cả client và server đều cần cài các hàm thư viện chung samba-common.2.0.5a.

Các chương trình daemon cần thiết để chạy samba là *smbd* (samba daemon) và *nmbd*, một chương trình hỗ trợ dịch vụ NetBiosb NameServer cho các trạm khách. Các chương tình trên được để trong thư mục */usr/bin*.

Chú ý rằng dịch vụ chuyển đổi tên-địa chỉ của nmbd là khác với dịch vụ này của DNS. Nmbd cung cấp dịch vụ giống như giao thức NetBIOS của Windows

cho ứng dụng samba. DNS không có khả năng cung cấp một dịch vụ tên-địa chỉ như vậy cho Samba.

Các lệnh của chương trình samba được để trong thư mục */usr/bin* bao gồm:

smbclient	Chương trình samba client trên các máy trạm Linux
smbprint	Chương trình phục vụ in ấn cho Samba trên trạm Linux.
smbprint.sysv	Giống chương trình trên nhưng cho các trạm Linux SRV4
smbrun	Một script để khởi động, kết thúc các chương trình samba.

2. Thực hiện các chương trình Samba Daemon

Hai chương trình cần thực hiện để cho Samba hoạt động là */usr/sbin/smbd* và */usr/sbin/nmbd*.

Các chương trình này có thể thực hiện từ trong *inetd* hoặc chạy như các tiến trình độc lập. Với kiểu chạy độc lập, samba sẽ hoạt động nhanh hơn so với thực hiện nó từ *inetd*.

Kiểm tra lại tệp */etc/services* để bảo đảm các dịch vụ của NetBIOS đã sẵn sàng hoạt động:

```
netbios-ns      137/tcp         nbns
netbios-ns      137/udp         nbns
netbios-dgm     138/tcp         nbdgm
netbios-dgm     138/udp         nbdgm
netbios-ssn     139/tcp         nbssn
```

Tuỳ thuộc vào từng phiên bản của Linux mà các dòng trên đã có sẵn hoặc ta cần thêm vào bằng tay. Samba sẽ không thể tìm thấy cổng dịch vụ dành cho nó nếu không có những dòng trên trong */etc/services*.

Để thực hiện Samba từ inetd, ta thêm các dòng sau vào tệp cấu hình */etc/inetd.conf*:

```
# SAMBA NetBIOS services (for PC tệp and print sharing)
netbios-ssn stream tcp nowait  oot /usr/sbin/smbd  smbd
netbios-ns dgram udp wait root /usr/sbin/nmbd  nmbd
```

Sau đó khởi động lại *inetd* bằng cách gửi tín hiệu HUP đến nó:

```
$ kill -HUP `cat /var/run/inetd.pid`
```

Khi khởi động samba, nếu ta nhận được thông báo không sử dụng được cổng 139, chứng tỏ có một chương trình samba nào đó đang sử dụng cổng này. Hãy kiểm tra lại tất cả các tiến trình đang chạy bằng lệnh "*ps* -aux" và dung lệnh kill để kết thúc tiến trình nào đang chiếm cổng này.

Đối với phiên bản Redhat Linux 6.1, đã có sẵn các chương trình script để sử dụng samba. Do vậy để khởi động các daemon ta chỉ cần gõ :

```
$ samba start
```

3. Tệp cấu hình /etc/smb.conf

Cấu hình của samba trên một trạm Linux hoặc trên các máy tính Unix khác được điều khiển bởi tệp tin */etc/smb.conf*. Tệp tin này xác định tài nguyên nào của hệ thống được chia sẻ, đồng thời xác định các quy tắc để truy cập được vào tài nguyên này.

Tệp */etc/smb.conf* được chia làm nhiều mục, mỗi mục là các tham số cho một tài nguyên được chia sẻ. Mỗi mục được bắt đầu bằng tên mục chẳng hạn [[global], [homes], [printers] ...

Trong phần tham số chung, ta cần chỉ ra tên của server khi dùng trong mạng và tên của nhóm hay miền mà nó có ý định chia sẻ tài nguyên.

Mục [global] section định nghĩa một số tham số toàn cục cho tất cả các tài nguyên được chia sẻ.

Mục [homes] cho phép người dùng trên các trạm ở xa (và chỉ người đó) được truy cập vào trong thư mục home của họ trên máy chủ Linux. Có nghĩa là khi một người sử dụng kết nối với máy Linux từ một trạm Windows, thư mục home của người đó sẽ được chia sẻ ngay cho người đó, với điều kiện là người này phải có một tài khoản trên máy Linux đó.

Trong ví dụ dưới đây, tệp *smb.conf* cho phép người dùng kết nối vào thư mục home của mình và một thư mục tạm thời có thể ghi được. Người sử dụng Windows chỉ việc kết nối vào máy Linux này qua mạng từ cửa sổ Windows File Manager hoặc Windows Explorer.

```
; /etc/smb.conf
;
servername     Sambaserver
group          Parallel

[global]
; Thiết lập kục này nếu ta muốn một tài khoản guest
; guest account = nobody
   log file = /var/log/samba-log.%m
   lock directory = /var/lock/samba
   share modes = yes
[homes]
   comment = Home Directories
   browseable = no
   read only = no
   create mode = 0750
[tmp]
   comment = Temporary file space
   path = /tmp
   read only = no
   public = yes
```

Sau khi sửa đổi tệp */etc/smb.conf*, ta cần phải kiểm tra lại các tham số đã được thiết lập đúng hay chưa bằng tiện ích *testparm*. Nếu các tham số này đã được thiết lập đúng, smbd có thể nạp cấu hình từ tệp tin này để hoạt động.

Với những máy trạm có nhiều hơn một giao diện mạng, *smbd* có thể không chọn được giao diện mạng để hoạt động. Ta cần thêm vào tham số interrfaces ở mục [global] của */etc/smb.conf*, ví dụ:

```
interfaces = 192.168.1.1/24
```

Tham số interfaces nhận giá trị là một địa chỉ IP và số hiệu lớp mạng. 24 là số hiệu của mạng loại C. Các số hiệu khác được chỉ ra trong tài liệu IP Masquerade Mini Howto.

4. Chia sẻ tệp tin của Linux cho Window

Như ví dụ chỉ ra trong tệp *smb.conf*, chia sẻ tệp tin từ máy Linux sang máy Windows rất đơn giản. Tuy nhiên, ta có thể điều khiển sự chia sẻ ở các mức độ khác nhau.

Để chia sẻ một thư mục cho tất cả mọi người, tạo một mục [public] trong smb.conf và thêm các tham số như sau:

```
[ public]
    comment = Public Stuff
    path = /home/public
    public = yes
    writable = yes
    printable = no
```

Để hạn chế số người được ghi vào thư mục trên, chẳng hạn chỉ cho phép nhóm staff là được quyền ghi, ta thêm tham số write list:

```
[ public]
    comment = Public Stuff
    path = /home/public
    public = yes
    writable = yes
    printable = no
    write list = @staff
```

Sau đó ta khởi động samba bằng script khởi động như đã mô tả trong phần trước:

```
$  samba start
```

Bây giờ ta đã có thể sử dụng máy Windows để truy cập vào thư mục public được chia sẻ trên máy Linux. Tuy nhiên với sự cải tiến của Microsoft nhằm tăng cường tính bảo mật của hệ thống, Windows98, WindowNT với service pack3 trở lên có sử dụng mật khẩu mã hoá (encrypted password) khi đăng nhập. Theo mặc định, Samba sử dụng mật khẩu dạng văn bản (plaintext password), do đó ta không thể truy cập được vào các thư mục của Linux do mật khẩu không đồng nhất.

Khi từ của sổ của Windows Explorer, ta nhấn vào biểu tượng của máy Linux và nhận được thông báo: *"You are not authorized to access that account from this*

machine", có nghĩa là mật khẩu đăng nhập là không đồng nhất. Có hai giải pháp để khắc phục vấn đề này: Hoặc là cho Samba hoạt động với mật khẩu mã hoá, hoặc là sửa đổi để Windows sử dụng mật khẩu văn bản thông thường.

Để cho Windows 95/98 hoạt động với mật khẩu văn bản, ta dùng bộ soạn thảo *registry editor* (regedit), trong mục:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\VxD\VNETSUP
```

Ta tạo một biến DWORD với tên và giá trị như sau:

```
EnablePlainTextPassword: 0x01
```

Tương tự cho Windows NT, vào mục:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Rdr\Parameters
```

Thêm một biến DWORD với tên và giá trị:

```
EnablePlainTextPassword: 0x01
```

Sau khi sửa đổi xong registry, khởi động lại Windows và ta đã có thể truy cập được vào Samba server vì lúc này, cả Samba và Windows đều sử dụng chung một dạng mật khẩu.

Để cấu hình cho Samba sử dụng mật khẩu mã hoá, trong mục [global] của tệp */etc/smb.conf*, ta thêm dòng tham số:

```
encrypt passwords = yes
smb passwd tệp = /etc/smbpasswd
```

Một số chi tiết về việc sử dụng mật khẩu mã hoá trong Samba được đề cập đến trong các tệp ENCRYPTION.txt, WinNT.txt và Win98.txt. Trước khi sử dụng Samba với mật khẩu mã hoá, ta nên tham khảo các tài liệu này.

Nếu như chúng ta nghi ngờ dịch vụ NetBIOS không hoạt động đúng do đặt cấu hình sai (có thể nhận được thông báo "host not found" khi thử truy cập), hãy sử dụng địa chỉ IP thay vì tên máy dưới dạng: '\\<host ip address>\<sharename>'.

Trong Linux, tên các tệp tin có phân biệt chữ hoa và chữ thường (case sensitive), trong Windows thì không. Do đó để nhận được chính xác tên tệp tin chúng ta phải sửa đổi một số tuỳ chọn sau:

```
; Mangle case = yes có nghĩa là cung cấp đúng tên tệp cho
; Win95/98/NT.
mangle case = yes
```

```
;không phân biệt chữ hoa khi tìm kiếm tên tệp
case sensitive = no
;Tên tệp mới tạo ra mặc định là chữ thường
default case = lower
;Giữ nguyên chữ hoa, chữ thường cho tất cả các tên tệp
preserve case = yes
;Giữ nguyên chữ hoa, chữ thường cho tên tệp kiểu dos (8.3)
short preserve case = no
```

Còn có nhiều tuỳ chọn khác nữa cho các thư mục được chia sẻ. Các tuỳ chọn trên là đủ cho bước đầu sử dụng Samba. Để sử dụng dịch vụ này ở mức độ sâu hơn, ta có thể tham khảo tài liệu của Samba, hoặc địa chỉ **http://www.samba.org**.

5. Chia sẻ các tệp tin Windows cho Linux

Hiện có hai phiên bản của chương trình Samba client. chương trình smbclient hoạt động theo kiểu FTP. Chương trình này cung cấp giao diện làm việc giống như giao diện của chương trình *ftp*. Trong đó máy Windows đóng vai trò như một "FTP server", và ta sử dụng tiện ích này để truyền tệp giữa nó với một trạm Linux.

Một tiện khác là samfs bao gồm 2 công cụ là *smbmount* và *smbumount* cho phép thiết lập các thư mục chia sẻ trên các máy Windows giống như các lệnh mount và *umount*. Để sử dụng được các công cụ này ta cần biên dịch hạt nhân có tuỳ chọn hỗ trợ smbfs.

Trong phần dưới đây ta chỉ trình bày cách sử dụng *smbclient*. Để biết được trên một máy Windows có những tài nguyên nào được chia sẻ ta dùng lệnh:

```
$ /usr/sbin/smbclient -L host
```

Trong đó *host* là tên của máy mà ta muốn kiểm tra. Nếu Samba server được cấu hình bảo mật, nó sẽ hỏi mật khẩu, ta có thể nhập mật khẩu của "guest" hoặc mật khẩu của mình trên server đó. Ví dụ:

```
$ smbclient -L zimmerman
```

Kết quả của lệnh trên có dạng như sau:

```
Server time is Sat Aug 10 15:58:27 1996
Timezone is UTC+10.0
```

```
Password:
Domain=[ WORKGROUP] OS=[ Windows NT 3.51] Server=[ NT LAN Manager 3.51]
Server=[ ZIMMERMAN] User=[ ] Workgroup=[ WORKGROUP] Domain=[ ]
        Sharename       Type      Comment
        ---------       ----      -------
        ADMIN$          Disk      Remote Admin
        public          Disk      Public
        C$              Disk      Default share
        IPC$            IPC       Remote IPC
        OReilly         Printer   OReilly
        print$          Disk      Printer Drivers
This machine has a browse list:
        Server               Comment
        ---------            -------
        HOPPER               Samba 1.9.15p8
        KERNIGAN             Samba 1.9.15p8
        LOVELACE             Samba 1.9.15p8
        RITCHIE              Samba 1.9.15p8
        ZIMMERMAN
```

Danh sách trên cho biết các Samba server khác có tài nguyên được chia sử trên mạng. Để sử dụng Samba client, ta gõ lệnh:

```
$ /usr/sbin/smbclient service <password>
```

Trong đó "service" là tên máy và tên tài nguyên được chia sẻ. Ví dụ nếu ta muốn đọc một thư mục được chia sẻ tên là *public* trên máy **zimmerman**, service sẽ có dạng **\\\\zimmerman\public**. password là mật khẩu của người dùng .

```
$ /usr/sbin/smbclient \\\\zimmerman\\public mypasswd
```

Chúng ta sẽ nhận được dấu nhắc của smbclient:

```
Server time is Sat Aug 10 15:58:44 1996
Timezone is UTC+10.0
Domain=[ WORKGROUP] OS=[ Windows NT 3.51] Server=[ NT LAN Manager 3.51]
smb: \>
```

Gõ h để xem giúp đỡ:

```
smb: \> h
ls              dir             lcd             cd              pwd
get             mget            put             mput            rename
more            mask            del             rm              mkdir
md              rmdir           rd              prompt          recurse
translate       lowercase       print           printmode       queue
cancel          stat            quit            q               exit
newer           archive         tar             blocksize       tarmode
setmode         help            ?               !
smb: \>
```

Chúng ta có thể sử dụng các lệnh như trong các phiên làm việc của *ftp* để truyền và nhận tệp. Nếu chưa thành thạo với *ftp*, cần tham khảo tài liệu của *smbclient* bằng lệnh "*man smbclient*".

Chương V

KHAI THÁC SHELL VÀ KERNEL

I. THAY ĐỔI CẤU HÌNH HẠT NHÂN

Nhìn chung, ta không nên động đến hạt nhân Linux trừ phi cần thực hiện các công việc nâng cấp, cài đặt mạng (như NFS hay NIS) hoặc các trình điều khiển thiết bị mới có các yêu cầu đặc biệt về hạt nhân, vì một sai sót trong việc sửa mã nguồn hạt nhân có thể làm ảnh hưởng đến toàn bộ hệ thống tệp. Do đó, trước khi làm việc với hạt nhân hãy tạo các đĩa khởi động và sao lưu hệ thống để tránh những sai sót đáng tiếc không thể phục hồi được xảy ra.

Các phiên bản Linux nói chung thường không hoàn toàn đồng nhất. Vì vậy, các lệnh sau đây có thể không đúng với một số phiên bản Linux, nhưng cách tiếp cận vấn đề là như nhau, có thể chỉ có tên thư mục và tên tiện ích là khác nhau.

Hạt nhân Linux được dịch bằng chương trình dịch C (là một phần của Linux). Trong phần này ta cũng xem xét đôi chút về trình dịch C, những gì có thể liên quan đến việc thay đổi cấu hình hạt nhân.

2. Nâng cấp và cài đặt phần mềm hạt nhân mới

Linux là một hệ điều hành linh hoạt. Các phiên bản mới của hạt nhân hay các phần của hệ điều hành có thể được liên kết vào hạt nhân luôn sẵn sàng cho mọi người dùng . Ta có thể sẽ phải biên dịch lại hạt nhân khi thêm phần mềm mới, hoặc không chỉ khi phần mềm đó được nạp dưới dạng tiện ích hoặc trình điều khiển thiết bị.

Nên tránh thường xuyên cập nhật hệ thống với mọi phiên bản mới, vì như vậy có thể xảy ra vấn đề không tương thích với hệ thống hiện tại, hoặc phải thường xuyên cấu hình lại hệ thống do việc cập nhật hay xoá mất các thông tin cấu hình. Việc cập nhật khiến ta phải tiêu phí thời gian vào việc dịch lại hạt nhân, do đó nên cân nhắc xem phiên bản đó có đáng giá với thời gian cài đặt và các vấn đề có thể nảy sinh hay không.

Cũng nên biết rằng khi nâng cấp cũng không bắt buộc phải nâng cấp mọi thứ. Nhiều phiên bản mới của Linux chỉ thay đổi khoảng 5% so với hệ thống cũ và chỉ các module chính mới được nâng cấp. Do vậy, cũng chỉ nên nâng cấp những phần có ảnh hưởng nhất định như chương trình dịch, thư viện, các tiện ích thường dùng. Phương pháp này tiết kiệm thời gian và việc cấu hình lại.

2. Dịch lại hạt nhân từ mã nguồn

Mọi việc thay đổi cấu hình hạt nhân : nâng cấp, thay thế, thêm mã mới vào hạt nhân đều được thực hiện theo cách : lấy mã nguồn của hạt nhân, thay đổi cấu hình, dịch lại và đặt nó vào đúng vị trí trong hệ thống tệp để hệ thống lại chạy một cách đúng đắn (quá trình này thường được tự động bằng các chương trình).

2.1. Tìm mã nguồn hạt nhân

Các chương trình nguồn hạt nhân Linux thường có trên CD-ROM, trên các địa chỉ FTP, các nhóm người dùng,... Hầu hết các phiên bản hạt nhân được đánh số bằng tên phiên bản và một mức patch. Ví dụ một hạt nhân tên là 1.12.123, trong đó 1 là số phiên bản chính, 12 là số phiên bản phụ, 123 là số patch. Hầu hết các nơi có các chương trình nguồn hạt nhân đều chứa vài phiên bản một lúc.

Đa số các chương trình nguồn hạt nhân đều được đặt dưới một tệp nén. Hãy tời tệp vào một thư mục con của */usr/src*, đây cũng là thư mục chứa hầu hết các chương trình nguồn của Linux.

2.2. Dùng các chương trình nguồn hạt nhân mới

Với chương trình nguồn hạt nhân mới, nhiệm vụ của ta là tời chương trình này, sau đó dịch và cấu hình lại.

Để tời chương trình này vào thư mục */usr/src*, cần phải tạo thư mục

/usr/src/linux, nó sẽ đè lên chương trình nguồn hạt nhân cũ. Để an toàn, trước tiên ta nên lưu lại chương trình hạt nhân cũ này để có một bản sao lưu phòng bất chắc.

Sau đó tạo 2 liên kết biểu tượng đến thư mục */usr/include* như sau :

```
ln -sf /usr/src/linux/include/linux /usr/include/linux
ln -sf /usr/src/linux/include/asm /usr/include/asm
```

Nếu không có 2 liên kết này quá trình nâng cấp sẽ không thực hiện được.

Sau khi đã tới chương trình nguồn và đặt các liên kết biểu tượng, ta có thể bắt đầu quá trình dịch. Để dịch ta dùng *gcc* hay *g++* (chương trình dịch GNU C và C++) hoặc các chương trình tương thích khác, nên xem xem chương trình nguồn mới có được hỗ trợ bởi *gcc* hay *g++* không.

Tìm tệp *Makefile* (thường trong thư mục */usr/src/linux*), tệp này thường có một dòng định nghĩa ROOT_DEV chỉ ra hệ thống tệp gốc khi Linux khởi động. hường là

```
ROOT_DEV = CURRENT
```

Nếu ta có các giá trị khác, cần đảm bảo là nó đúng với cấu hình hệ thống tệp. Nếu *Makefile* không có giá trị, hãy đặt cho nó như dòng trên.

Quá trình dịch bắt đầu bằng việc chuyển đến thư mục */usr/src/linux* và đưa ra lệnh :

```
$ make config
```

nó gọi tiện ích *make* của chương trình dịch C và đưa ra một loạt các câu hỏi cấu hình yêu cầu ta trả lời trước khi dịch, đó là các câu hỏi về kiểu đĩa đang dùng, CPU, các phân vùng, hay các thiết bị khác như CD-ROM. Nếu ta không biết chắc các thông tin này hãy dùng giá trị mặc định.

Tiếp theo là đặt các phụ thuộc nguồn (source dependencies). Dùng lệnh sau :

```
$ make dep
```

Nếu phần mềm đang cài đặt không có tệp *dep*, kiểm tra lại hướng dẫn cài đặt xem các phụ thuộc có phải được đặt trong các bước khác hay không.

Cuối cùng ta dịch hạt nhân mới. Dùng lệnh :

```
$ make Image
```

Lệnh này sẽ dịch hạt mã nguồn nhân mới và đặt tệp ảnh hạt nhân trong thư mục hiện tại (như ở đây là */usr/src/linux*). Nếu muốn có một ảnh nén hạt nhân, ta dùng lệnh :

```
$ make zimage
```

Tuy vậy không phải phiên bản nào cũng hỗ trợ dịch ảnh nén.

Bước cuối cùng, sao tệp ảnh hạt nhân mới vào đĩa khởi động hay đĩa mềm khởi động. Để đặt vào đĩa mềm khởi động dùng lệnh :

```
$ cp Image /dev/fd0
```

Bằng cách khác, nếu muốn dùng LILO để khởi động hệ điều hành, ta cài đặt hạt nhân mới bằng cách chạy chương trình cài đặt hoặc */usr/lilo/lilo*. Nhưng đừng chép đè hạt nhân mới lên hạt nhân cũ của đĩa khởi động.

Sau đó, chỉ cần khởi động lại hệ thống. Nếu có vấn đề gì, hãy khởi động lại bằng đĩa mềm và khôi phục lại hạt nhân cũ.

3. Thêm các trình điều khiển thiết bị vào hạt nhân

Đôi khi ta muốn liên kết một trình điều khiển thiết bị mới (chẳng hạn khi thêm một ổ đĩa quang vào hệ thống) hay một phần mềm nào đó vào hạt nhân mà không phải thực hiện quá trình nâng cấp hạt nhân. Các phần mềm ghép hạt nhân (add-in kernel) thường có các hướng dẫn cài đặt, nhưng quá trình nói chung là :

- Đặt mã nguồn vào trong thư mục mà tiến trình dịch lại hạt nhân có thể tìm được (thư mục */usr/src*).
- Sửa tệp *Makefile* để yêu cầu *make* thêm mã mới vào hạt nhân. Việc này được thực hiện bởi script cài đặt hoặc ta tự làm lấy. Một số phần mềm có tệp *Makefile* riêng phục vụ việc này.

Sau đó ta bắt đầu dịch lại hạt nhân, tương tự như đã chỉ ra trong phần trên.

4. Nâng cấp thư viện

Đa số các phần mềm trên hệ thống Linux đều dùng các thư viện chia sẻ. Đôi khi sau khi nâng cấp hệ thống chúng ta gặp phải thông báo

```
Incompatible library version
```

Điều đó có nghĩa là thư viện đã được nâng cấp nhưng chưa được dịch lại. Nói chung các thư viện đều tương thích ngược, vì vậy các phần mềm hiện tại sẽ làm việc tốt sau khi nâng cấp thư viện.

Thư viện ít khi được nâng cấp hơn là hạt nhân. Các chương trình thư viện nâng cấp cũng hay đặt cùng chỗ với các chương trình hạt nhân nâng cấp. Thường thì có các tài liệu hướng dẫn chỉ ra đâu là thư viện mới nhất, hoặc các thư viện nào là cần thiết cho phiên bản mới của hạt nhân. Các bản nâng cấp thư viện cũng là các tệp nén, quá trình tời chúng cũng giống như tời chương trình nguồn hạt nhân, chỉ khác thư mục đích là */lib*, */usr/lib*, và */usr/include*. Thường các tệp *.a* hay *.aa* sẽ vào thư mục */usr/lib*, các tệp *libc.so.version* vào */lib*.

Cần phải chuyển các liên kết biểu tượng của hệ thống tệp để trỏ đến phiên bản mới nhất của thư viện, ví dụ ta đang chạy phiên bản thư viện *libc.so.4.4.1* và nâng cấp lên *libc.so.4.4.2*, ta phải thay liên kết biểu tượng đặt trong */lib* đến tệp này.

```
$ ln -sf /lib/libc/so/4/4/2 /lib/libc.so.4
```

Trong đó tham số cuối cùng của lệnh trên là tên tệp thư viện hiện tại trong /lib.

Ta cũng phải đổi liên kết biểu tượng đến tệp libm.so.version theo cách đó. Lưu ý là không được xoá các liên kết biểu tượng, nếu không tất cả các chương trình phụ thuộc vào thư viện (gồm cả *ls*) sẽ không hoạt động được.

5. Sử dụng chương trình dịch C

Để dịch hạt nhân và hầu hết các tiện ích, Linux dùng chương trình dịch C. Chương trình dịch C có sẵn đối với mọi phiên bản của Linux là trình dịch GNU C gọi tắt là *gcc*, nó tương thích hoàn toàn với ANSI C. Nếu đã quen với một trình dịch C trên một hệ điều hành khác rồi, bạn sẽ học *gcc* rất nhanh.

Cú pháp cơ bản của gcc như sau

```
$ gcc [ cac-lua-chon] [ cac-ten-tep ]
```

Trong đó `cac-ten-tep` là tên các tệp cần dịch, `cac-lua-chon` là các tuỳ chọn của lệnh *gcc* sẽ được áp dụng trên mọi tệp cần dịch.

5.1. Các lựa chọn của chương trình dịch

Nhiều tuỳ chọn của *gcc* có nhiều hơn một kí tự, do đó các tuỳ chọn này nhất thiết phải đặt sau dấu "-", và cũng vì vậy ta không thể gộp các tuỳ chọn như vẫn làm với các lệnh Linux khác. Ví dụ sau đây là 2 lệnh có ý nghĩa khác nhau:

```
$ gcc -p -g test.c
$ gcc -pg test.c
```

Khi dịch một chương trình bằng *gcc* mà không có tuỳ chọn nào trên dòng lệnh, ta sẽ thu được một tệp thực thi tên là *a.out*. Để đặt cho tệp thực thi tên khác phải dùng tuỳ chọn -o. Tên tệp thực thi được đặt ngay sau tuỳ chọn -o. Ví dụ dịch tệp *count.c* thành tệp thực thi *count* :

```
$ gcc -o count count.c
```

Các tuỳ chọn khác cho phép chỉ ra mức độ dịch mà ta muốn. Tuỳ chọn -c yêu cầu gcc dịch mã nguồn đến dạng object mà không đến giai đoạn dịch sang mã asembler và giai đoạn liên kết. Tuỳ chọn này thường được dùng vì nó làm cho quá trình dịch các chương trình C gồm nhiều tệp nhanh hơn và dễ quản lí. Tệp object có đuôi mặc định là *.o*.

Tuỳ chọn -S yêu cầu *gcc* ngừng dịch sau khi đã sinh ra tệp asembler (mặc định đuôi tệp asembler là *.s*).

Tuỳ chọn -E yêu cầu chương trình dịch hiển thị các giai đoạn tiền xử lí dịch trên tệp đầu vào. Khi đặt tuỳ chọn này đầu ra của tiền xử lí được chuyển ra đầu ra chuẩn (mặc định là màn hình) chứ không được lưu trong tệp.

Khi dịch bằng *gcc*, *gcc* sẽ cố gắng dịch mã nguồn trong thời gian ngắn nhất, dễ gỡ rối nhất (tức là mã sau dịch có thứ tự như mã nguồn) mà không thực hiện một tối ưu hoá nào cả. Chương trình *gcc* cũng cung cấp nhiều tuỳ chọn để yêu cầu dịch ra các chương trình thực thi nhỏ hơn, nhanh hơn, đó là các tuỳ chọn -O và -O2.

Tuỳ chọn -O yêu cầu *gcc* thực hiện các tối ưu cơ bản trên mã nguồn làm cho chương trình chạy nhanh hơn trong đa số các trường hợp. Tuỳ chọn -O2 yêu cầu tạo ra mã nhỏ nhất và chạy nhanh nhất có thể. Tuỳ chọn -O2 làm cho tốc độ dịch chậm hơn tuỳ chọn -O nhưng kết quả là mã dịch được chạy nhanh hơn.

5.2. Các tuỳ chọn gỡ rối và profile

gcc cung cấp một số tuỳ chọn gỡ rối và profile. Hai trong số các tuỳ chọn này thường được dùng nhất là : *-g* và *-pg*.

Tuỳ chọn *-g* yêu cầu *gcc* sinh ra các thông tin gỡ rối để trình gỡ rối của GNU (*gdb*) dùng giúp ta gỡ rối chương trình. Với *gcc* có thể kết hợp cả hai tuỳ chọn -O và -g, khi đó vừa có thể sinh ra mã tối ưu, vừa có thể thực hiện gỡ rối mã gần nhất với mã được sinh ra cuối cùng. Khi dùng hai tuỳ chọn này kết hợp với nhau, có thể *gcc* sẽ thay đổi một chút mã chương trình đã viết.

Tuỳ chọn -pg yêu cầu *gcc* thêm mã mở rộng vào chương trình, khi đã thực thi các mã mở rộng này sẽ sinh ra các thông tin sơ lược mà chương trình *gprof* dùng để hiển thị thông tin về chương trình.

5.3. Chương trình gỡ rối gdb

Linux có chương trình gỡ rối của GNU gọi là *gdb*, được dùng để gỡ rối một chương trình C hay C++. Chương trình này cho phép thấy được cấu trúc bên trong hay bộ nhớ mà một chương trình dùng khi đang thực thi.

Có thể dùng *gdb* với một số các tuỳ chọn trên dòng lệnh. Dạng thường dùng của *gdb* như sau :

```
$ gdb ten-tep
```

Trong đó `ten-tep` là tên tệp cần thực hiện gỡ rối. Ngoài ra *gdb* còn cung cấp một số tuỳ chọn khác. Chi tiết các tuỳ chọn có thể được xem này bằng lệnh *man gcc*.

Để *gdb* làm việc đúng, cần dịch chương trình theo cách mà chương trình dịch sẽ sinh ra các thông tin gỡ rối, tức là dùng tuỳ chọn -g khi dịch. Thông tin gỡ rối được sinh ra bao gồm kiểu mỗi biến trong chương trình, ánh xạ giữa các địa chỉ trong chương trình thực thi với các số dòng trong mã nguồn. Chương trình *gdb* sẽ dùng các thông tin này để liên hệ mã thực thi với mã nguồn.

II. LẬP TRÌNH VỚI SHELL

Lập trình shell là một công cụ đắc lực cho các người quản trị hệ thống. Khả năng viết các chương trình ngắn nhưng lại thực hiện được các công việc tốn thời gian được người quản trị coi là mạnh hơn hẳn các công cụ khác. Phần này sẽ nói về một số khái niệm cơ bản nhưng hữu ích cho công việc quản trị hàng ngày.

1. Shell và các loại shell

1.1. Shell là gì

Vai trò của shell là thực hiện chuyển đổi các lệnh được người dùng gõ vào thành các lệnh đối với hệ điều hành ví dụ dùng lệnh sau :

```
$ sort -n phonelist > phonelist.inorder
```

có nghĩa là sắp xếp các dùng trong tệp *phonelist* theo thứ tự số và đặt kết quả trong tệp *phonelist.inorder*.

Hoạt động của shell với lệnh nhận được như sau :

- Cắt dòng lệnh nhận được thành các đoạn nhỏ, như ở đây là : `sort`, `n`, `phonelist`, `>`, `phonelist.inorder`. Mỗi phần nhỏ này được gọi là một từ.
- Xác định vai trò của mỗi từ, ví dụ `sort` là tên lệnh, `n` và `phonelist` là các tham số ...
- Xác định đầu vào và ra thông qua các dấu hiệu định hướng. Ví dụ, căn cứ vào `>phonelist.inorder` thì đầu ra là tệp có tên `phonelist.inorder`
- Tìm lệnh, ví dụ `sort` trong các tệp nhị phân và thực hiện nó với tuỳ chọn `-n`

1.2. Các loại shell

Do thị trường UNIX hoàn toàn tự do mà có rất nhiều chương trình cùng kiểu được viết ra như các shell, các trình soạn thảo, các công cụ cho người quản trị. Những phiên bản tốt được phổ biến rộng rãi còn những phiên bản khác dần dần tự biến mất. Shell cũng chịu ảnh hưởng của tình trạng đó, do đó mà cũng có khá nhiều bản shell khác nhau. Tuy vậy sự khác nhau của các bản shell này cũng

Không lớn lắm, sự khác biệt có thể thấy rõ nhất ở các cấu trúc điều khiển như vòng lặp, if ... then...

Hiện tại có một số shell chính chạy dưới Linux như sau :

Bourne Again shell (bash)

Hiện nay được dùng nhiều nhất gần đây dưới UNIX, nó tương thích *với* Bourne shell và hơn nữa lại thừa các điểm hay nhất của Korn và C shell đồng thời có những điểm mạnh riêng của nó.

Trước hết phải kể đến một điểm mà *bash* hấp dẫn mọi người dùng là chế độ soạn thảo các dòng lệnh của nó. Khả năng này cho phép người dùng dễ dàng sửa chữa các lỗi khi gõ lệnh hoặc gọi lại các lệnh trước đó. Cơ chế gọi lại lệnh trước của C shell khó sử dụng còn Bourneshell thì không cho phép các xử lí này.

Một ưu điểm nữa của bash là nó cho phép thực hiện đồng thời các lệnh chỉ trong một cửa sổ *terminal*, đây là một chức năng thừa kế của C shell .

Đối với người lập trình và người thích tạo ra môi trường làm việc riêng của mình thì phải kể đến việc *bash* cung cấp nhiều tuỳ chọn cấu hình và khả năng lập trình được mở rộng đáng kể bao gồm định nghĩa các hàm, các cấu trúc điều khiển, tính toán số học trên số nguyên, vào ra ...

Bourne shell (sh)

Bourne shell được coi như trình dịch lệnh chuẩn, nó là shell gốc. Nó không cung cấp khả năng sửa đổi trên dòng lệnh như *bash*.

C shell (csh)

C shell được phát triển tại trường đại học Berkeley. Nó tương thích với cả Bourne shell cho sử dụng tương tác, nhưng lại có giao diện lập trình rất khác. Nó không cho phép soạn thảo (thực hiện sử đổi) trên dòng lệnh nhưng bù lại lại cho phép thao tác với các lệnh đã thực hiện trước nhờ cơ chế lưu lại các lệnh đã thực hiện.

Korn shell (ksh)

Đây có lẽ là loại shell phổ biến nhất trong các hệ thống UNIX, Korn shell là shell đầu tiên đưa ra các kĩ thuật hiện đại (một số trong đó mượn của C shell) trong Bourne shell (nó tương thích với cả loại shell này). Nó cho phép soạn thảo trên dòng lệnh.

tcsh

Là C shell đã được cải tiến. Cho phép soạn thảo trên dòng lệnh, ngoài ra còn phát triển một số tính chất khác. Những tính chất mà bash có được của Bourne shell thì C shell cũng có.

zsh

Z shell , là loại mới nhất, nhỏ. Nó tương thích với Bourne shell , cung cấp khả năng soạn thảo trên dòng lệnh.

pdksh (Public Domain Korn shell)

Là loại Korn shell được phổ biến rộng rãi trên Internet (miễn phí). Nó có tất cả các tính chất của Bourne shell thêm một số mở rộng của POSIX shell (POSIX : Portable Operating System Interface) và một số tính chất riêng của nó. Điểm mạnh nổi bật của pdksh là mã thực thi của nó chỉ nhỏ cỡ bằng 1/3 của *bash* và thực hiện nhanh hơn đáng kể. Nhưng ngược lại nó không tuân theo POSIX do đó ít có ứng dụng được phát triển trên nó, và nó không bị điều khiển khắt khe như bash.

Cuối cùng thì pdksh là sự lựa chọn tốt cho những ai không muốn dùng *bash* mà lại không thể có được Korn shell .

Các phần sau sẽ trình bày về lập trình shell với các ví dụ đưa ra trên 3 loại shell : *bash*, *tcsh*, *pdksh*.

1.3. Viết và chạy các chương trình shell

Ở mức đơn giản nhất, chương trình shell chỉ là một tệp chứa một số shell hay lệnh Linux. Người quản trị dùng chương trình này để đơn giản hoá các công việc lặp đi lặp lại, để thay các lệnh thường được thực hiện cùng nhau bằng một lệnh duy nhất, hay để tự động hoá việc cài đặt một chương trình, hoặc để viết các ứng dụng tương tác đơn giản.

Để tạo một chương trình shell, trước hết phải tạo một tệp và dùng một trình soạn thảo văn bản (text) để viết các lệnh Linux hay các shell mà ta muốn thực hiện vào đó. Sau đây ta sẽ xem một ví dụ một chương trình shell đơn giản.

Chẳng hạn ta có ổ CD-ROM được thiết lập với hệ thống tệp Linux ngay khi hệ thống khởi động. Nếu sau đó muốn đổi đĩa CD trong ổ, để hệ thống đọc nội dung đĩa CD mới, ta phải bỏ đĩa CD mới vào ổ, thực hiện bỏ nối đĩa CD (bằng lệnh

umount) rồi lại nối lại ổ CD (bằng lệnh mount). Điều đó được thực hiện bằng các lệnh sau :

```
$ umount /dev/cdrom

$ mount /dev/cdrom /cdrom
```

Để thay cho công việc nhàm chán là thực hiện hai lệnh này mỗi khi muốn đổi đĩa CD, ta sẽ tạo một chương trình shell để nó thực hiện hai lệnh này cho ta. Chương trình shell sẽ chỉ gồm 2 dòng lệnh viết đúng như trên và được ghi vào tệp có tên là *remount*. Như vậy để thực hiện công việc mong muốn ta chỉ việc thực hiện chương trình *remount* vừa viết. Có vài cách để thực hiện chương trình này.

Cách thứ nhất, chạy *remount* từ dòng lệnh. Trước hết phải làm cho tệp *remount* có thể thực thi được bằng lệnh :

```
$ chmod +x remount
```

Để chạy chương trình, ta chỉ việc gõ *remount* trên dòng lệnh. Lưu ý là tệp *remount* phải được đặt trong các thư mục đã chỉ ra trong biến môi trường PATH, nếu không shell sẽ không tìm thấy nó để thực thi.

Cách thứ hai, chạy chương trình shell đã viết bằng cách chuyển nó cho shell như một tham số. Nếu ta dùng chương trình tcsh thì:

```
$ tcsh remount
```

Lệnh này sẽ kích hoạt một shell mới và yêu cầu nó thực hiện các lệnh trong tệp remount. Nhưng nếu dùng *tcsh*, ta phải đặt đầu chương trình shell mà ta viết một dấu # để tcsh nhận ra đấy là một chương trình của nó. Thực tế nên đưa dòng chỉ chương trình dịch shell (ví dụ #!/bin/sh với Bourne shell) vào đầu chương trình shell để chỉ chính xác shell nào được dùng để dịch, tránh tình trạng C shell lại dịch chương trình Bourne shell.

Cách thứ ba để thực hiện chương trình shell là dùng lệnhs. (chấm) với cả *pdksh* và *bash*, hoặc lệnh *source* trong *tcsh*. Các lệnh này yêu cầu shell thực hiện tệp được chỉ ra như một tham số. Ví dụ yêu cầu *pdksh* hay *bash* thực hiện *remount*.

```
. remount
```

hoặc với *tcsh* :

```
. source remount
```

2. Dùng các biến

Như hầu hết các ngôn ngữ, sử dụng biến là một khía cạnh quan trọng trong chương trình shell. Có một số biến dùng chung như PATH hay TERM, đây là các biến trong của shell được shell định nghĩa. Trong phần này chúng ta sẽ xem xét cách tạo ra các biến shell của riêng mình và sử dụng chúng trong chương trình.

2.1. Gán giá trị cho biến

Với mọi shell được cung cấp cùng với Linux (Bourne, Korn và C shell) Để gán giá trị cho các biến, ta viết tên biến, tiếp theo là dấu bằng và giá trị mà ta muốn gán cho biến. Ví dụ với *bash* hay *pdksh* ta dùng lệnh sau để gán cho biến count giá trị bằng 5 :

```
count=5
```

Với *tcsh* ta gõ lệnh

```
set count = 5
```

Cần lưu ý là với *bash* và *pdksh* hai bên dấu bằng không được có dấu cách còn với *tcsh* thì không ảnh hưởng gì.

Vì shell là ngôn ngữ dịch không định kiểu, do đó không phải khai báo kiểu của biến. Cũng có thể dùng một biến để lưu một chuỗi kí tự hay một số nguyên. Cách gán một xâu kí tự cho biến cũng giống như cách gán một số nguyên cho biến, tuy nhiên chuỗi kí tự đó không thể có dấu cách ở giữa (nếu không ta phải dùng các kí tự đặc biệt để ẩn đi dấu cách này).

```
name=Garry              ( với pdksh và bash)

set name = Garry        (với tcsh)
```

Giá trị của một biến được biểu diễn bằng tên biến đó có dấu $ đứng liền đằng trước. Ví dụ, giá trị của biến count được biểu diễn bằng $count. Để in ra màn hình giá trị biến count, ta đưa ra lệnh :

```
$ echo $count
```

Lệnh echo sẽ đưa tham số của nó ra dòng ra chuẩn. Nếu ta quên dấu $ đằng trước tên biến shell sẽ hiểu count trong lệnh là một xâu kí tự và sẽ in ra màn hình chữ count.

2.2. Các tham số vị trí và các biến trong của shell

Khi chạy một chương trình shell có một số tham số trên dòng lệnh mỗi tham số này được lưu trong các biến đặc biệt. Tham số đầu tiên lưu trong biến có tên là 1, tham số thứ hai lưu trong biến có tên là 2, và cứ tiếp tục như vậy (vì thế ta cũng không thể dùng các số này để đặt tên biến cho mình). Do đó khi muốn lấy giá trị của các tham số trong các biến ta cũng thêm dấu $ vào đầu các tên biến này. Ví dụ một chương trình program thực hiện in ra hai tham số trên dòng lệnh theo thứ tự đảo ngược, tham số thứ hai in trước, rồi đến tham số thứ nhất, chương trình được viết như sau :

```
echo "$2"
echo "$1"
```

Nếu ta gọi chương trình này như sau :

```
reverse hello there
```

thì chương trình sẽ đưa ra kết quả trên màn hình:

```
there hello
```

Bảng 5.1. Một số biến trong quan trọng của shell

Biến	ý nghĩa
$#	Số tham số trên dòng lệnh được chuyển cho chương trình shell
$?	Lưu mã trả lại của lệnh cuối cùng được thực hiện
$0	Từ đầu tiên của lệnh được gõ, đó chính là tên chương trình shell.
$*	Lưu tất cả các tham số trên dòng lệnh được viết kề nhau ("$1 $2 ..."
"$@"	Lưu tất cả các tham số dòng lệnh trên các xâu riêng biệt ("$1" "$2" ...)

3. Dùng các dấu

Shell dùng nhiều dấu khác nhau cho các mục đích đặc biệt. Dấu ngoặc kép (""), dấu nháy đơn (')và dấu gạch chéo (\) được shell sử dụng để che giấu các kí tự đặc biệt. Dấu nháy đơn có một ý nghĩa đặc biệt đối với shell và không được dùng để đóng khung các xâu kí tự. Mỗi phương pháp cho phép ẩn đi các kí tự đặc biệt trong shell ở các mức độ khác nhau.

Dấu ngoặc kép khi được dùng để đóng khung một xâu kí tự sẽ làm cho các

dấu cách được ẩn đi đối với shell. Tuy vậy các kí tự đặc biệt khác vẫn được dịch như thường. Điều này thường được dùng khi gán một xâu kí tự có nhiều hơn một từ cho một biến. Ví dụ gán xâu `hello there` cho biến tên là greeting.

```
greeting="hello there"              (trong bash hay pdksh)
set greeting = "hello there"        (trong tcsh)
```

Nếu ta không để xâu hello there trong ngoặc kép thì *bash* hay *pdksh* sẽ không hiểu lệnh, còn với tcsh thì chỉ từ `hello` được gán cho biến greeting.

Dấu nháy đơn là dấu mạnh nhất, nó che dấu tất cả các kí tự đặc biệt của shell. Dấu này hay được dùng khi lệnh đưa ra là dành cho một chương trình khác chứ không phải là shell (vì như vậy lệnh đưa ra hoàn toàn không được shell dịch). Ta cũng có thể dùng nó để đóng khung một xâu khi gán cho biến như đã làm ở trên bằng dấu ngoặc kép. Tuy vậy, đôi khi dùng dấu nháy đơn lại không tốt, nhất là khi ta muốn kèm theo một biến vào xâu kí tự cần gán. Ví dụ , ta muốn gán xâu gồm cả tên người dùng cho biến greeting :

```
greeting="hello there $LOGNAME"
```

khi đó `greeting` sẽ chứa `hello there` tiếp theo là tên người dùng đang đăng nhập hệ thống (được chứa trong biến `LOGNAME`).

Còn nếu

```
greeting='hello there $LOGNAME'
```

Thì `greeting` sẽ chứa đúng xâu `hello there` $LOGNAME.

Dùng dấu gạch chéo \ là cách thứ ba để che dấu các kí tự đặc biệt, nhưng khác với dấu nháy đơn che cả một nhóm kí tự, dấu gạch chéo chỉ che được một kí tự đứng ngay sau nó. Ví dụ thay vì dùng dấu ngoặc kép ta có thể dùng dấu gạch chéo như sau :

```
greeting=hello\ there          (trong bash và pdksh)
```

Trong đó dấu \ đã làm cho shell không nhận ra dấu cách giữa chữ `hello` và chữ `there`, nhờ đó mà xâu `hello there` vẫn được gán cho `greeting`. Dấu \ rất hay được dùng để ẩn đi một kí tự đặc biệt đối với shell, nhất là khi ta kí tự đó xuất hiện trong một xâu. Chẳng hạn, ta cần lưu giá của một đĩa trong một biến `disk_price`, giá này được tính bằng đô la do đó nó có dấu đô la ở đầu, ta dùng lệnh :

```
disk_price=\$5.00              (với bash và pdksh)
```

Như vậy biến disk_price chứa xâu $5.00. Nếu không có dấu \ ở đầu thì shell sẽ cố tìm biến có tên là 5 để gán giá trị của nó cho disk_price, nếu không thấy, shell sẽ gán cho disk_price giá trị .00.

Dấu nháy ngược (` `) lại có chức năng khác. Ta dùng chúng khi muốn dùng kết quả của một lệnh cho một lệnh khác. Ví dụ để gán cho biến content danh sách các tệp trong thư mục hiện tại, ta viết :

```
contents=`ls`
```

Như vậy là lệnh ls sẽ được thực hiện và kết quả của nó được lưu trong biến content.

4. Dùng lệnh test

Trong bash và pdksh, lệnh *test* được dùng để đánh giá một biểu thức điều kiện. Ta thường dùng nó để đánh giá một điều kiện trong một câu lệnh điều kiện hay đánh giá điều kiện vào hay ra của một câu lệnh lặp. Lệnh *test* có cú pháp như sau :

```
test expression
```

hay

```
[ expression ]
```

Ta có thể dùng một số toán tử của shell với lệnh *test*. Các toán tử này được chia làm 4 nhóm: toán tử trên xâu kí tự, toán tử trên số nguyên, toán tử trên tệp, toán tử logic.

Ta dùng các toán tử trên xâu kí tự để đánh giá các biểu thức xâu kí tự, sau đây là bảng toán tử trên xâu kí tự :

Bảng 5.2. Các toán tử trên xâu kí tự cho lệnh *test*

Toán tử	Ý nghĩa
str1 = str2	Trả lại true nếu str1 đồng nhất (giống hệt) str2
str1 != str2	Trả lại true nếu str1 không đồng nhất với str2
str	Trả lại true nếu str khác null
-n str	Trả lại true nếu độ dài của xâu str lớn hơn 0
-z str	Trả lại true nếu độ dài xâu str bằng 0

Các toán tử của shell trên số nguyên cũng tương tự như trên xâu kí tự chỉ có khác là chúng thực hiện trên số nguyên.

Bảng 5.3. Các toán tử số nguyên cho lệnh test

Toán tử	Ý nghĩa
int1 -eq int2	Trả lại true nếu int1 bằng int2
int1 -ge int2	Trả lại true nếu int1 lớn hơn hoặc bằng int2
int1 -gt int2	Trả lại true nếu int1 lớn hơn int2
int1 -le int2	Trả lại true nếu int1 nhỏ hơn hoặc bằng int2
int1 -lt int2	Trả lại true nếu int1 nhỏ hơn int2
int1 -ne int2	Trả lại true nếu int1 không bằng int2

Ta dùng các toán tử trên tệp để kiểm tra xem một tệp có tồn tại hay không, thuộc kiểu nào. Tên tệp được chuyển cho phép toán dưới dạng tham số.

Bảng 5.4. Các toán tử trên tệp cho lệnh *test*

Toán tử	Ý nghĩa
-d tệp	Trả lại true nếu tệp là thư mục
-f tệp	Trả lại true nếu tệp là một tệp thông thường
-r tệp	Trả lại true nếu tệp là đọc được đối với tiến trình đưa ra phép toán này
-s tệp	Trả lại true nếu tệp có kích thước khác 0
-w tệp	Trả lại true nếu tệp là ghi được đối với tiến trình đưa ra phép toán này
-x tệp	Trả lại true nếu tệp là thực thi được đối với tiến trình đưa ra phép toán này

Toán tử logic để kết hợp các phép toán trên xâu kí tự, số nguyên, tệp hay để lấy phủ định của một phép toán trên xâu, số nguyên, tệp.

Bảng 5.5. Các toán tử logic cho lệnh test

Toán tử	Ý nghĩa
! expr	Trả lại true nếu biểu thức là không đúng
expr1 -a expr2	Trả lại true nếu expr1 và expr2 đều đúng.
expr1 -o expr2	Trả lại true nếu expr1 hoặc expr2 đúng.

Tcsh shell không có lệnh *test*, nhưng các biểu thức trong *tcsh* thực hiện các chức năng tương tự. Các phép toán trong *tcsh* giống như trong C.

Bảng 5.6. Các toán tử số nguyên trong biểu thức *tcsh*

Toán tử	Ý nghĩa
int1 <= int2	Trả lại true nếu int1 nhỏ hơn hoặc bằng int2
int1 >= int2	Trả lại true nếu int1 lớn hơn hoặc bằng int2
int1 < int2	Trả lại true nếu int1 nhỏ hơn int2
int1 > int2	Trả lại true nếu int1 lớn hơn int2

Bảng 5.7. Các toán tử trên xâu trong biểu thức *tcsh*

Toán tử	Ý nghĩa
str1 == str2	Trả lại true nếu str1 bằng str2
str1 != str2	Trả lại true nếu str1 không bằng str2

Bảng 5.8. Các toán tử tệp cho biểu thức *tcsh*

Toán tử	Ý nghĩa
-r tệp	Trả lại true nếu tệp là đọc được
-w tệp	Trả lại true nếu tệp là ghi được
-x tệp	Trả lại true nếu tệp là thực thi được
-e tệp	Trả lại true nếu tệp tồn tại
-o tệp	Trả lại true nếu tệp là sở hữu của người dùng hiện tại
-z tệp	Trả lại true nếu tệp có kích thước là 0
-f tệp	Trả lại true nếu tệp là tệp thông thường
-d tệp	Trả lại true nếu tệp là một thư mục

Bảng 5.9. Các toán tử logic trong biểu thức *tcsh*

Toán tử	Ý nghĩa
exp1 || exp2	Trả lại true nếu một trong exp1 hoặc exp2 là đúng
exp1 && exp2	Trả lại true nếu cả exp1 và exp2 là đúng
! exp	Trả lại true nếu exp là không đúng.

5. Câu lệnh điều kiện

Shell có 2 loại câu lệnh điều kiện, lệnh if và lệnh case. Ta dùng các câu lệnh này để thực hiện các phần chương trình khác nhau nếu điều kiện đúng. Cú pháp của các lệnh này có thể hơi khác nhau với các shell khác nhau.

5.1. Lệnh if

Cả 3 loại shell *bash*, *pdksh*, *tcsh* đều hỗ trợ lệnh `if-then-else` lồng nhau. Câu lệnh này cho phép thực hiện các kiểm tra điều kiện phức tạp trong chương trình shell của ta. Cú pháp của lệnh với *bash* hay *pdksh* là:

```
if [ biểu_thức_1 ]
then
  các lệnh
elif [ biểu_thức_2 ]
  các lệnh
else
  các lệnh
fi
```

Nói chung, tên viết ngược của các câu lệnh thường được dùng để đóng câu lệnh trong các trường hợp lệnh phức tạp. Ở câu lệnh trên `fi` được dùng để đóng câu lệnh `if`.

`Elif` và `else` đều là các phần không bắt buộc trong câu lệnh `if`. `Elif` là dạng viết tắt của `else if`. Câu lệnh này chỉ thực hiện khi không có một biểu thức điều kiện nào của lệnh `if` trước hay cuả các lệnh `elif` trước là đúng. Các lệnh của câu lệnh `else` chỉ thực hiện khi không một biểu thức điều kiện nào của lệnh `if` hay của các `elif` trước đó là đúng.

Trong *tcsh* câu lệnh `if` có 2 dạng khác. Dạng thứ nhất có chức năng như trong *bash* :

```
if (biểu_thức_1) then
các lệnh
else if (biểu_thức_2) then
các lệnh
```

```
else
các lệnh
endif
```

Trong đó `else if` cũng giống như `elif` với *bash* và cũng có thể viết là `elif`, `endif` cũng giống như `fi` với *bash*. Để chỉ rõ là chương trình được dịch bằng *tcsh* , dòng đầu của chương trình nên ghi rõ chương trình shell dịch :

```
#!/bin/sh
```

Dạng thứ hai là dạng đơn giản của lệnh `if`, nó chỉ có một câu lệnh `if` đầu tiên với một biểu thức để đánh giá và một câu lệnh thực hiện. Nếu biểu thức đúng, nó thực hiện câu lệnh, nếu sai thì sẽ không có chuyện gì xảy ra cả, nó thực hiện các dòng lệnh tiếp theo. Cú pháp câu lệnh :

```
if (expression) command
```

Sau đây là ví dụ lệnh `if` trong *bash*. Lệnh này kiểm tra xem có tệp *.profile* trong thư mục hiện tại hay không.

```
if [ -f .profile ]
then
 echo "Có tệp .profile trong thư mục hiện tại."
else
 echo "Không tìm thấy tệp .profile."
fi
```

Các câu lệnh tương tự được viết cho *tcsh*

```
#
if ( { -f .profile } ) then
echo "Có tệp .profile trong thư mục hiện tại."
else
echo "Không tìm thấy tệp .profile."
endif
```

Dấu # ở dòng đầu để chỉ ra tệp chứa chương trình này là tệp chứa một script của tcsh.

5.2. *Câu lệnh case*

Câu lệnh case cho phép ta so sánh một mẫu với một số mẫu khác và thực hiện khối chương trình tương ứng với mẫu phù hợp. Câu lệnh case trong shell mạnh hơn trong các ngôn ngữ lập trình khác, nó cho phép ta so sánh các xâu có kí tự thay thế trong đó (với C hay Pascal ta chỉ có thể so sánh các kiểu liệt kê (enumerated type) hay các giá trị nguyên).

Cú pháp lệnh trong bash hay pdksh :

```
case string1 in
 str1)
  các lệnh;;
 str2)
các lệnh;;
 *)
các lệnh;;
esac
```

string1 được so sánh với str1 rồi str2. Nếu một trong hai xâu này phù hợp, các lệnh tương ứng cho đến dấu ;; sẽ được thực hiện. Nếu str1 và str2 đều không phù hợp với string1, các lệnh tương ứng với dấu * sẽ được thực hiện. Các lệnh ứng với dấu * là các lệnh mặc định vì * phù hợp với mọi xâu kí tự.

Trong *tcsh*, cú pháp câu lệnh case như sau, giống như trong C :

```
switch (string1)
 case str1:
các câu lệnh
breaksw
 case str2:
 các câu lệnh
breaksw
 default:
 các câu lệnh
 breaksw
endsw
```

Nó thực hiện hoàn toàn giống như trong *bash*, cũng so sánh `string1` với `str1`, `str2`. `breaksw` có ý nghĩa như dấu `;;` ở trên, `default` có giá trị như * ở trên.

Sau đây là một ví dụ câu lệnh `case` trong bash. Đoạn mã sau kiểm tra tuỳ chọn đầu tiên trên dòng lệnh có phải là `-i` hay `-e` hay không. Nếu là `-i`, chương trình tính số dòng trong tệp được chỉ ra trong tuỳ chọn thứ hai trên dòng lệnh mà các dòng này bắt đầu bằng chữ `i`. Nếu là `-e`, chương trình tính số dòng trong tệp chỉ ra trong tuỳ chọn thứ hai trên dòng lệnh mà các dong này bắt đầu bằng chữ `e`. Nếu tuỳ chọn đầu tiên trên dòng lệnh không phải là `-i` hay `-e`, chương trình sẽ đưa ra thông báo lỗi trên màn hình.

```
case $1 in
 -i)
 count=`grep ^i $2 | wc -l`
 echo "The number of lines in $2 that start with an i is
$count"
 ;;
 -e)
 count=`grep ^e $2 | wc -l`
 echo "The number of lines in $2 that start with an e is
$count"
 ;;
 * )
 echo "That option is not recognized"
 ;;
esac
```

Tương tự ta cũng có thể viết trong tcsh.

6. Câu lệnh lặp

Shell cung cấp một số kiểu câu lệnh lặp khác nhau. Ta dùng các câu lệnh này khi phải thực hiện các hành động lặp đi lặp lại, như khi ta xử lí một danh sách các tệp.

6.1. Câu lệnh for

Câu lệnh for thực hiện các lệnh trong nó một số lần định trước. Lệnh `for` trong *bash* và *pdksh* có hai loại khác nhau :

```
for var1 in list
do
các lệnh
done
```

Theo dạng này các lệnh bên trong câu lệnh `for` được thực hiện một lần với mỗi phần tử trong danh sách `list`. `List` có thể là một biến hoặc một danh sách các giá trị được gõ trực tiếp vào câu lệnh. Mỗi lần lặp biến `var1` được gán cho phần tử hiện tại của `list` cho đến khi phần tử cuối cùng được gán.

Dạng thứ hai có cú pháp như sau :

```
for var1
do
các câu lệnh
done
```

Dưới dạng này câu lệnh thực hiện một lần với mỗi phần tử trong biến `var1`. Trong đó shell giả thết rằng `var1` chứa toàn bộ các tham số trên dòng lệnh được chuyển cho chương trình shell. Dạng này tương đương với câu lệnh `for` sau :

```
for var1 in "$@"
do
các câu lệnh
done
```

Tương tự, ta có câu lệnh `for` cho *tcsh* gọi là `foreach`, cũng giống như trong *bash* và *pdksh*. Cú pháp như sau :

```
For each name (list)
các lệnh
end
```

Sau đây là ví dụ câu lệnh `for` trong *bash* hay *pdksh*. Ví dụ này lấy các tệp có tên được chỉ ra trên dòng lệnh, chuyển nội dung các tệp này từ các chữ thường ra

chữ hoa rồi ghi lại vào tệp cùng tên nhưng với đuôi là *.caps*.

```
for file
do
tr a-z A-Z < $tệp >$tệp.caps
done
```

Chương trình tương tự viết trong *tcsh*

```
#
for each file ($*)
 tr a-z A-Z < $tệp >$tệp.caps
end
```

6.2. Câu lệnh while

Một câu lệnh lặp khác của shell là `while`. Câu lệnh này làm cho một khối mã được thực hiện chừng nào điều kiện đưa ra còn đúng. Cú pháp của nó trong *bash* và *pdksh* như sau :

```
while biểu_thức_điều_kiện
do
các câu lệnh
done
```

Trong tcsh :

```
while (biểu_thức_điều_kiện)
các câu lệnh
end
```

Sau đây là một ví dụ về câu lệnh while trong *bash* hay *pdksh*. Chương trình này liệt kê các tham số được truyền cho chương trình cùng với số của tham số.

```
count=1
while [ -n "$*" ]
do
 echo "Dây là tham số thứ $count $1"
  shift
 count=`expr $count + 1`
done
```

Lệnh shift dịch các tham số trên dòng lệnh sang trái một số, khi đó tham số số 2 sẽ trở thành tham số số 1, tham số số 3 sẽ trở thành tham số số 2, ...

`expr $count + 1` là xâu ghi biểu thức tăng count lên 1, khi lấy giá trị của biểu thức này, nó sẽ thực hiện nội dung biểu thức tức là thực hiện cộng thêm 1 vào count.

Cũng chương trình trên viết trong *tcsh* :

```
#
set count = 1
while ( "$*" != "" )
 echo "Đây là tham số thứ $count $1"
 shift
 set count = `expr $count + 1`
end
```

6.3. Câu lệnh until

Câu lệnh until cũng có cú pháp và chức năng tương tự như lệnh while, chỉ có khác là câu lệnh thực hiện khối mã cho đến khi biểu thức điều kiện sai. Cú pháp của nó trong *bash* và *pdksh* như sau :

```
until biểu_thức_điều_kiện
do
các lệnh
done
```

Cũng với ví dụ đã viết với câu lệnh while ta viết bằng câu lệnh until như sau :

```
count=1
until [ -z "$*" ]
do
echo "Đây là tham số thứ $count $1"
shift
count=`expr $count + 1`
done
```

Điểm khác biệt khi ta sử dụng câu lệnh `while` và `until` cho ví dụ này là ở chỗ, trong câu lệnh `while` ta dùng điều kiện kiểm tra `-n`, có nghĩa là độ dài xâu khác 0, trong câu lệnh `until` ta dùng điều kiện kiểm tra `-z` tức là chiều dài xâu bằng 0. Câu lệnh `until` hoàn toàn có thể viết được nhờ câu lệnh `while` nên trong thực tế nó ít khi được dùng.

6.4. Lệnh shift

Lệnh `shift` dịch các tham số vị trí trên dòng lệnh (các tham số mà ta gõ trên dòng ngay khi gọi lệnh, được lưu trong các biến có tên là các số 1, 2,...) một vị trí sang trái. Ví dụ nếu các giá trị hiện tại của các tham số vị trí trên dòng lệnh là :

```
$ command  file0 file1 file2
```

Sau khi ta thực hiện lệnh shift

```
shift
```

thì các tham số vị trí trên dòng lệnh sẽ là :

```
$1 = file1 $2 = file2
```

Ta cũng có thể dịch nhiều hơn một vị trí bằng cách chỉ ra số vị trí muốn dịch sau lệnh `shift`. Ví dụ để dịch hai vị trí :

```
shift 2
```

Lệnh này rất có ích khi ta có một chương trình shell cần phân tích các tham số hay các tuỳ chọn trên dòng lệnh. Các tham số hay các tuỳ chọn này thường được xử lí bằng một vòng lặp nào đó, ta thường muốn bỏ qua một tham số tiếp theo khi đã biết tham số tiếp theo là tham số nào. Ví dụ, chương trình shell sau đây chờ một trong hai loại tuỳ chọn trên dòng lệnh, `-i` và tham số tiếp theo sẽ là tên tệp đầu vào, và tuỳ chọn `-o` với tham số tiếp theo sẽ là tên tệp đầu ra. Chương trình sẽ đọc tệp đầu vào, biến các chữ hoa thành chữ thường và ghi ra tệp đầu ra.

```
while [ "$1" ]
do
 if [ "$1" = "-i" ] then
 infile="$2"
 shift 2
 else if [ "$1" = "-o" ] then
```

```
    outfile="$2"
    shift 2
    else
    echo "Program $0 does not recognize option $1"
    fi
done
tr a-z A-Z <$infile >$outfile
```

6.5. Câu lệnh select

Câu lệnh `select` chỉ có trong shell pdksh. Khác với các câu lệnh lặp khác, nó không thực hiện một khối lệnh khi biểu thức điều kiện đúng hay sai. Mục tiêu của câu lệnh này chỉ để tạo thực đơn dạng text đơn giản cho chương trình. Cú pháp của câu lệnh như sau :

```
select menuitem [ in list_of_items]
do
các câu lệnh
done
```

Khi thực hiện câu lệnh `select`, pdksh tạo một số các mục trong thực đơn cho mỗi phần tử của danh sách `list_of_item`. `list_of_item` có thể là một biến bao gồm nhiều hơn một phần tử như `choice1 choice2` hay là danh sách các mục được gõ ngay trong lệnh như sau :

```
select menuitem in choice1 choice2 choice3
```

Nếu không đưa ra `list_of_item`, câu lệnh select sẽ dùng các tham số vị trí như câu lệnh `for` vẫn dùng.

Khi người dùng chương trình có câu lệnh `select` chọn một mục trong thực đơn đã tạo bằng cách gõ số tương ứng với mục đó, câu lệnh select sẽ lưu gía trị của mục được chọn trong biến `menuitem`. Các câu lệnh trong khối do có thể thực hiện các hành động tương ứng với mục đó trong thực đơn.

Sau đây là ví dụ thực hiện câu lệnh `select`. Ví dụ này hiển thị một thực đơn gồm 3 mục. Khi người dùng chọn một mục, chương trình sẽ hỏi có phải định chọn mục đó không. Nếu người dùng gõ y hoặc Y, chương trình sẽ hiển thị lại thực đơn.

```
select menuitem in pick1 pick2 pick3
do
 echo "Có phải bạn muốn chọn mục $menuitem"
 read res
 if [ $res = "y" -o $res = "Y" ]
 then
 break
 fi
done
```

Trong này có một số lệnh mới, lệnh `read` để đọc đầu vào mà người dùng gõ vào và lưu vào biến mà lệnh `read` chỉ ra. Lệnh `break` để thoát khỏi các câu lệnh `while`, `for` hay `select`.

6.6. Câu lệnh repeat

Tcsh shell có câu lệnh lặp mà *bash* và *pdksh* không có câu lệnh tương tự (nhưng ta hoàn toàn có thể viết được nó từ câu lệnh *while*), đó là lệnh *repeat*. Câu lệnh *repeat* thực hiện một lệnh duy nhất một số lần. Cú pháp của nó :

```
repeat số_lần lệnh
```

Ví dụ sau đây đọc các một tập các số từ các tham số trên dòng lệnh và in ra số các chấm bằng số đó trên màn hình (giống như một chương trình đồ hoạ).

```
#
foreach num ($*)
 repeat $num echo -n "."
 echo ""
end
```

7. Sử dụng chương trình con

Shell cho phép ta định nghĩa các hàm của riêng mình, các hàm này cũng được đối xử như các hàm trong C và các ngôn ngữ lập trình khác, cũng như trong các ngôn ngữ này các hàm làm cho chương trình shell trở nên sáng sủa hơn và cũng tránh phải viết đi viết lại các đoạn mã lặp lại.

Cú pháp để tạo một hàm trong *bash* và *pdksh* :

```
Tên_hàm() {
Các câu lệnh shell
}
```

Ngoài ra còn có thể viết (ý nghĩa hoàn toàn như nhau) :

```
function tên_hàm {
Các câu lệnh shell
}
```

Sau khi đã tạo ta gọi các hàm như sau :

```
fname [ parm1 parm2 parm3 ...]
```

Khi các tham số được chuyển cho hàm, hàm coi chúng là các tham số vị trí như các chương trình shell vẫn chuyển tham số của chúng trên dòng lệnh. Ví dụ sau đây chứa một vài hàm, mỗi hàm thực hiện một nhiệm vụ riêng liên quan tới một tham số trên dòng lệnh của chương trình. Chương trình đọc tất cả các tệp được chỉ ra trên dòng lệnh, tuỳ vào tham số được chọn mà chương trình sẽ viết ra tệp nội dung với các kí tự được viết dưới dạng chữ hoa hay chữ thường hay in ra nội dung của tệp.

```
upper() {
shift
for i   #thực hiện lần lượt với từng tệp được chỉ ra trên
dòng lệnh
do
 #đọc tệp có tên trong $1 đổi chữ hoa thành chữ thường, ghi
ra tệp $1.out
 tr a-z A-Z <$1 >$1.out
 rm $1          # xoá tệp $1
 mv $1.out $1        #chuyển tệp $1.out thành tệp $1
 shift
done; }
lower () {
shift
for i   # thực hiện lần lượt với từng tệp được chỉ ra trên
dòng lệnh
do
 #đọc tệp $1 đổi chữ hoa thành chữ thường, ghi ra tệp $1.out
```

```
  tr A-Z a-z <$1 >$1.out
  rm $1                # xoá tệp $1
       mv $1.out $1          #chuyển tệp $1.out thành tệp $1
  shift
done; }

print () {
shift
for i
do
       lpr $1
  shift
done; }

usage_error () {
echo "$1 syntax is $1 <option> <input files>"
echo ""
echo "where option is one of the following"
echo "p -- to print frame files"
echo "u -- to save as uppercase"
echo "l -- to save as lowercase"; }
case $1
in
       p | -p) print $@;;
  u | -u) upper $@;;
  l | -l) lower $@;;
  *) usage_error $0;;
esac
}
```

Một số script hữu ích trong quản trị hệ thống

Trong phần cuối, chúng tôi giới thiệu với bạn đọc một số script tiện ích thường được dùng trong việc quản trị hệ thống. Đây là các script khởi tạo các dịch vụ phổ biến cho một máy tính cũng như các máy chủ. Các dịch vụ này bao gồm cả dịch vụ người dùng như dịch vụ máy chủ Web, máy chủ lưu trữ FTP và dịch vụ hệ thống như dịch vụ syslog để ghi nhật ký hoạt động của hệ thống.

```
#!/bin/bash

. /etc/rc.d/init.d/functions

if [ -f /etc/sysconfig/httpd ]; then
        . /etc/sysconfig/httpd
fi

INITLOG_ARGS=""

# Đặt HTTPD=/usr/sbin/httpd.worker trong
# /etc/sysconfig/httpd để có khởi tạo daemon với một
# tuyến (thread) cơ sở là worker MPM. Lưu ý có một số
# module của apache server không làm việc tốt khi chạy ở
# chế độ này

# Đường dẫn trỏ tới script apchectl
apachectl=/usr/sbin/apachectl
httpd=${HTTPD-/usr/sbin/httpd}
prog=httpd
RETVAL=0

# Kiểm tra cấu hình theo khuôn dạng phiên bản 1.3
check13 () {
  CONFFILE=/etc/httpd/conf/httpd.conf
  GONE="(ServerType|BindAddress|Port|AddModule|ClearModuleList|
"
  GONE="${GONE}AgentLog|RefererLog|RefererIgnore|FancyIndexing|
"
  GONE="${GONE}AccessConfig|ResourceConfig)"
  if grep -Eiq "^[[:space:]]*($GONE)" $CONFFILE; then
        echo
        echo 1>&2 " Apache 1.3 configuration directives found"
        echo 1>&2 " please read /usr/share/doc/httpd-
2.0.47/migration.html"
        failure "Apache 1.3 config directives test"
```

```
        echo
        exit 1
  fi
}

# Luu y hai apachectrl quy dinh rang, viec kich hoat dich
# vu trong khi dich vu da duoc kich hoat tu truoc se gay
# loi. Tuong tu, viec tat dich vu khi dich vu khong hoat
# dong cung duoc xem la loi. Do vay, cac viec khoi tao
# cung nhu tat dich vu se khong goi truc
# tiep apachectrl ma thong qua cac ham sau.

start() {
        echo -n $"Starting $prog: "
        check13 || exit 1
        daemon $httpd $OPTIONS
        RETVAL=$?
        echo
        [ $RETVAL = 0 ] && touch /var/lock/subsys/httpd
        return $RETVAL
}
stop() {
  echo -n $"Stopping $prog: "
  killproc $httpd
  RETVAL=$?
  echo
  [ $RETVAL = 0 ] && rm -f /var/lock/subsys/httpd
/var/run/httpd.pid
}
reload() {
  echo -n $"Reloading $prog: "
  check13 || exit 1
  killproc $httpd -HUP
  RETVAL=$?
  echo
}

# Kiem tra yeu cau cua nguoi dung. Kich hoat (start), tat
# (stop)hoac khoi dong lai server.
case "$1" in
  start)
  start
  ;;
  stop)
  stop
  ;;
  status)
        status $httpd
  RETVAL=$?
  ;;
  restart)
  stop
```

```
  start
  ;;
  condrestart)
  if [ -f /var/run/httpd.pid ] ; then
        stop
        start
  fi
  ;;
  reload)
        reload
  ;;
  graceful|help|configtest|fullstatus)
  $apachectl $@
  RETVAL=$?
  ;;
  *)
  echo $"Usage: $prog
{start|stop|restart|condrestart|reload|status|fullstatus|gracef
ul|help|configtest}"
  exit 1
esac

exit $RETVAL
```

Script killall

```
#! /bin/bash

# Tắt tất cả các dịch vụ đang chạy nhưng không cần thiết.
# Trước khi tắt, các dịch vụ đều được kiểm thử. Do vậy
# không phải bất cứ dịch nào cũng bị tắt.

case "$1" in
   *start)
  ;;
   *)
  echo $"Usage: $0 {start}"
  exit 1
  ;;
esac
for i in /var/lock/subsys/* ; do
  # Kiểm tra xem các script trong các dịch vụ có tồn tại
  # hay không
  [ -f "$i" ] || continue

  # Lấy tên dịch vụ
  subsys=${i#/var/lock/subsys/}

  # Kiểm tra xem mạng có hoạt động hay không
  [ $subsys = network ] && continue
```

```
    # Tắt dịch vụ
    if [ -f /etc/init.d/$subsys.init ]; then
           /etc/init.d/$subsys.init stop
    elif [ -f /etc/init.d/$subsys ]; then
           /etc/init.d/$subsys stop
    else
           rm -f "$i"
    fi
done
```

Script ghi nhật ký hệ thống

```
#!/bin/bash
#
# syslog        Kích hoạt syslogd/klogd.
#
#
# chkconfig: 2345 12 88
# Mô tả: Đây là dịch vụ giúp các dịch vụ khác có thể ghi
# lại nhật ký hoạt động. Nhật ký hoạt động được lưu trữ
# dưới dạng tệp tin nhật ký hệ thống. Đây là một dịch vụ
# rất hữu ích, giúp người quản trị mạng có thể tìm được
# lỗi khi hệ thống có vấn đề cũng như
# giúp hệ thống hoạt động hiệu quả hơn.

### BEGIN INIT INFO
# Provides: $syslog
### END INIT INFO

# Đường dẫn trỏ tới thư viện các hàm cơ bản
. /etc/init.d/functions

[ -f /sbin/syslogd ] || exit 0
[ -f /sbin/klogd ] || exit 0

# Đường dẫn chỉ tới tệp tin lưu cấu hình
if [ -f /etc/sysconfig/syslog ] ; then
  . /etc/sysconfig/syslog
else
  SYSLOGD_OPTIONS="-m 0"
  KLOGD_OPTIONS="-2"
fi

RETVAL=0

umask 077

start() {
```

```
    echo -n $"Starting system logger: "
    daemon syslogd $SYSLOGD_OPTIONS
    RETVAL=$?
    echo
    echo -n $"Starting kernel logger: "
    daemon klogd $KLOGD_OPTIONS
    echo
    [ $RETVAL -eq 0 ] && touch /var/lock/subsys/syslog
    return $RETVAL
}
stop() {
    echo -n $"Shutting down kernel logger: "
    killproc klogd
    echo
    echo -n $"Shutting down system logger: "
    killproc syslogd
    RETVAL=$?
    echo
    [ $RETVAL -eq 0 ] && rm -f /var/lock/subsys/syslog
    return $RETVAL
}
rhstatus() {
    status syslogd
    status klogd
}
restart() {
    stop
    start
}

case "$1" in
  start)
        start
  ;;
  stop)
        stop
  ;;
  status)
        rhstatus
  ;;
  restart|reload)
        restart
  ;;
  condrestart)
        [ -f /var/lock/subsys/syslog ] && restart || :
  ;;
  *)
  echo $"Usage: $0 {start|stop|status|restart|condrestart}"
  exit 1
esac

exit $?
```

Script khởi tạo các dịch vụ mạng

```
#!/bin/bash
#
# xinetd        Kích hoạt và tắt xinetd.
#
# chkconfig: 345 56 50
# Mô tả: xinetd được sử dụng thay thế cho inetd trong các
# hệ thống trước kia
# xinetd:
# - có cơ chế điều khiển truy nhập,
# - được mở rộng khả năng ghi nhật ký
# - Cho phép các dịch vụ được kích hoạt dựa trên thời gian
# - Có khả năng hạn chế số lượng dịch vụ được kích hoạt
# - Và nhiều ưu điểm khác
#
# Tên tiến trình: /usr/sbin/xinetd
# Các tệp tin cấu hình:
#                               /etc/sysconfig/network
#                               /etc/xinetd.conf
# Tệp tin ghi định danh tiến trình: /var/run/xinetd.pid
# PATH=/sbin:/bin:/usr/bin:/usr/sbin

# Đường dẫn trỏ tới thư viện chứa các hàm cơ bản
# . /etc/init.d/functions

# Đọc cấu hình về mạng
test -f /etc/sysconfig/network && . /etc/sysconfig/network

# Đọc cấu hình về các dịch vụ

test -f /etc/sysconfig/xinetd && . /etc/sysconfig/xinetd

# Kiểm tra người yêu cầu kích hoạt có phải là root ...
# Nếu sai, script dừng tại đây.
[ `id -u` = 0 ] || exit 1

# Kiểm tra hệ thống mạng có hoạt động hay không.
[ "${NETWORKING}" = "yes" ] || exit 0

[ -f /usr/sbin/xinetd ] || exit 1
[ -f /etc/xinetd.conf ] || exit 1

RETVAL=0

prog="xinetd"

start(){
    echo -n $"Starting $prog: "
# Thêm các thông tin về múi giờ, đặc tả về khuôn dạng hiển thị
# trong  /etc/synconfig/xinetd
```

```
    if [ -z "$XINETD_LANG" -o "$XINETD_LANG" = "none" -o
"$XINETD_LANG" = "NONE" ]; then
        unset LANG LC_TIME LC_ALL LC_MESSAGES LC_NUMERIC
LC_MONETARY LC_COLLATE
    else
        LANG="$XINETD_LANG"
        LC_TIME="$XINETD_LANG"
        LC_ALL="$XINETD_LANG"
        LC_MESSAGES="$XINETD_LANG"
        LC_NUMERIC="$XINETD_LANG"
        LC_MONETARY="$XINETD_LANG"
        LC_COLLATE="$XINETD_LANG"
        export LANG LC_TIME LC_ALL LC_MESSAGES LC_NUMERIC
LC_MONETARY LC_COLLATE
    fi
    unset HOME MAIL USER USERNAME
    daemon $prog -stayalive -pidfile /var/run/xinetd.pid
"$EXTRAOPTIONS"
    RETVAL=$?
    echo
    touch /var/lock/subsys/xinetd
    return $RETVAL
}

stop(){
    echo -n $"Stopping $prog: "
    killproc $prog
    RETVAL=$?
    echo
    rm -f /var/lock/subsys/xinetd
    return $RETVAL

}

reload(){
    echo -n $"Reloading configuration: "
    killproc $prog -HUP
    RETVAL=$?
    echo
    return $RETVAL
}

restart(){
    stop
    start
}

condrestart(){
    [ -e /var/lock/subsys/xinetd ] && restart
    return 0
}
```

```
# Kiểm tra yêu cầu người dùng.
case "$1" in
    start)
  start
  ;;
    stop)
  stop
  ;;
    status)
  status $prog
  ;;
    restart)
  restart
  ;;
    reload)
  reload
  ;;
    condrestart)
  condrestart
  ;;
    *)
  echo $"Usage: $0
{start|stop|status|restart|condrestart|reload}"
  RETVAL=1
esac

exit $RETVAL
```

Lưu ý rằng, chúng tôi đưa ra các script ra đây không chỉ đơn thuần mang tính chất minh hoạ cho những câu lệnh được giới thiệu trong các phần trên. Chúng tôi muốn nhấn mạnh tới khả năng vượt trội của script trong công việc quản trị hệ thống. Bạn đọc có thể tham khảo thêm các script khác trong thư mục /etc/init.d.

TÀI LIỆU THAM KHẢO

I. Sách

[1] - The Linux System Administrator's Guide -1994

[2] - OLAF KIRCH - The Linux Network Administrator's Guide - 1994

[3] - EVI N. and GARTH S. and SCOTT S. - Unix System Administration Handbook -1989

[4] - Unix, Utilisation et Administration.

[5] - MARK G. SOLBELL - A pratice guide to the Unix system - 1994

2. Các địa chỉ Internet

[5] - **www.linuxdocs.org** - Các tài liệu Linux Howtos

[6] - **www.linux.org** - Các thông tin liên quan đến Linux

[7] - **www.samba.org** - Tài liệu về Samba

NGUYỄN THANH THỦY (Chủ biên),
LÊ QUÂN, NGUYỄN HỒNG SƠN, TRƯƠNG DIỆU LINH

Quản trị

HỆ THỐNG LINUX

Chịu trách nhiệm xuất bản: PGS, TS TÔ ĐĂNG HẢI

Biên tập: ĐẶNG VĂN SỬ

NGUYỄN NGỌC KHUÊ

Trình bày bìa: HƯƠNG LAN

$\frac{6.6T3}{KHKT-2005}$ 546 - 37 - 2005

NHÀ XUẤT BẢN KHOA HỌC VÀ KỸ THUẬT

70 Trần Hưng Đạo - Hà Nội

In 800 cuốn khổ 16 x 24 cm, tại Công ty in Hàng không
Giấy phép xuất bản số : 546 – 57 - 2005..
In xong và nộp lưu chiểu tháng 10/2005.

205240

8 935048 952405

Giá: 33.000[đ]