ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

NHẬP MÔN
HỆ ĐIỀU HÀNH LINUX

Tài liệu khóa tập huấn quản trị mạng theo tài trợ của dự án "Nâng cao chất lượng giáo dục và đào tạo" từ vốn vay của Ngân hàng thế giới

--- Tiểu dự án "A" ---

Thành phố Hồ chí Minh 10/2001

(Lưu hành nội bộ)

MỞ ĐẦU

Với sự phát triển ngày càng mạnh mẽ của mạng tin học toàn cầu Internet xuất hiện ngày càng nhiều nhu cầu về nguồn nhân lực chuyên nghiệp để quản trị hệ thống mạng dùng riêng phức hợp với giao tiếp ra Internet.

Là một đơn vị chịu trách nhiệm quản trị mạng tin học của Đại học quốc gia Tp HCM, chúng tôi đã có nhiều kinh nghiệm trong công tác quản trị một mạng Intranet rộng lớn với hàng ngàn máy tính kết nối và truy cập Internet qua đường dùng riêng (leased-line). Qua giáo trình này, chúng tôi muốn đưa đến bạn đọc những kiến thức cơ bản nhất, cho phép cài đặt và quản trị một hệ thống server Unix cùng với các dịch vụ Internet cơ bản. Các ví dụ thường được dựa trên hệ điều hành (HDH) Linux hay Sun OS, là hai HDH đang được sử dụng rộng rãi trong mạng ĐHQG-HCM. Chúng tôi cũng sẽ đề cập đến giao thức TCP/IP và cách triển khai TCP/IP trên một máy chủ Unix.

Với phương châm "**chỉ nói về những gì chúng tôi đã sử dụng trong thực te**" chúng tôi hy vọng rằng giáo trình rất ngắn gọn này sẽ có ích một cách thiết thực cho những bạn đọc muốn học về hệ điều hành Unix và công nghệ mạng Internet, cũng như các quản trị viên mạng Internet trên cơ sở máy chủ Unix.

Do thời gian rất eo hẹp cho công tác chuẩn bị, chúng tôi chắc rằng sẽ có những thiếu sót, mong bạn đọc góp ý và xin cảm ơn trước các nhận xét của bạn đọc. Mọi ý kiến xin gửi về :

Trịnh Ngọc Minh

3 Công trường Quốc tế, Q.3 Thành phố Hồ chí minh

tnminh@vnuhcm.edu.vn

I. Giới thiệu lịch sử phát triển của Unix và Linux:

i. Vài dòng về lịch sử UNIX:

Giữa năm 1960, AT&T Bell Laboratories và một số trung tâm khác tham gia vào một cố gắng tạo ra một hệ điều hành mới được đặt tên là Multics (Multiplexed Information and Computing Service). Đến năm 1969, chương trình Multics bị bãi bỏ vì đó là một dự án quá nhiều tham vọng. Thậm trí nhiều yêu cầu đối với Multics thời đó đến nay vẫn chứa có được trên các Unix mới nhất. Nhưng Ken Thompson, Dennis Ritchie, và một số đồng nghiệp của Bell Labs đã không bỏ cuộc. Thay vì xây dựng một HĐH làm nhiều việc một lúc, họ quyết định phát triển một HĐH đơn giản chỉ làm tốt một việc là chạy chương trình (run program). HĐH sẽ có rất nhiều các công cụ (tool) nhỏ, đơn giản, gọn nhẹ (compact) và chỉ làm tốt một công việc. Bằng cách kết hợp nhiều công cụ lại với nhau, họ sẽ có một chương trình thực hiện một công việc phức tạp. Đó cũng là cách thức người lập trình viết ra chương trình. Peter Neumann đặt tên Unix cho HĐH đơn giản này. tiếp tục phát triển theo mô hình ban đầu và đặt ra một hệ thống tập tin mà sau này được phát triển thành hệ thống tập tin của UNIX. Vào năm 1973, sử dụng ngôn ngữ C của Ritchie, Thompson đã viết lại toàn bộ HĐH Unix và đây là một thay đổi quan trọng của Unix, vì nhờ đó Unix từ chỗ là HĐH cho một máy PDP-xx trở thành HĐH của các máy khác với một cố gắng tối thiểu để chuyển đổi. Khoảng 1977 bản quyền của UNIX được giải phóng và HDH UNIX trở thành một thương phẩm.

*ii. **Hai dòng UNIX** : System V của AT&T , Novell và Berkeley Software Distribution (BSD) của Đại học Berkeley.*

- ***System V :***

Các phiên bản UNIX cuối cùng do AT&T xuất bản là System III và một vài phát hành (releases) của System V. Hai bản phát hành gần đây của System V là Release 3 (SVR3.2) và Release 4.2 (SVR4.2). Phiên bản SYR 4.2 là phổ biến nhất cho từ máy PC cho tới máy tính lớn.

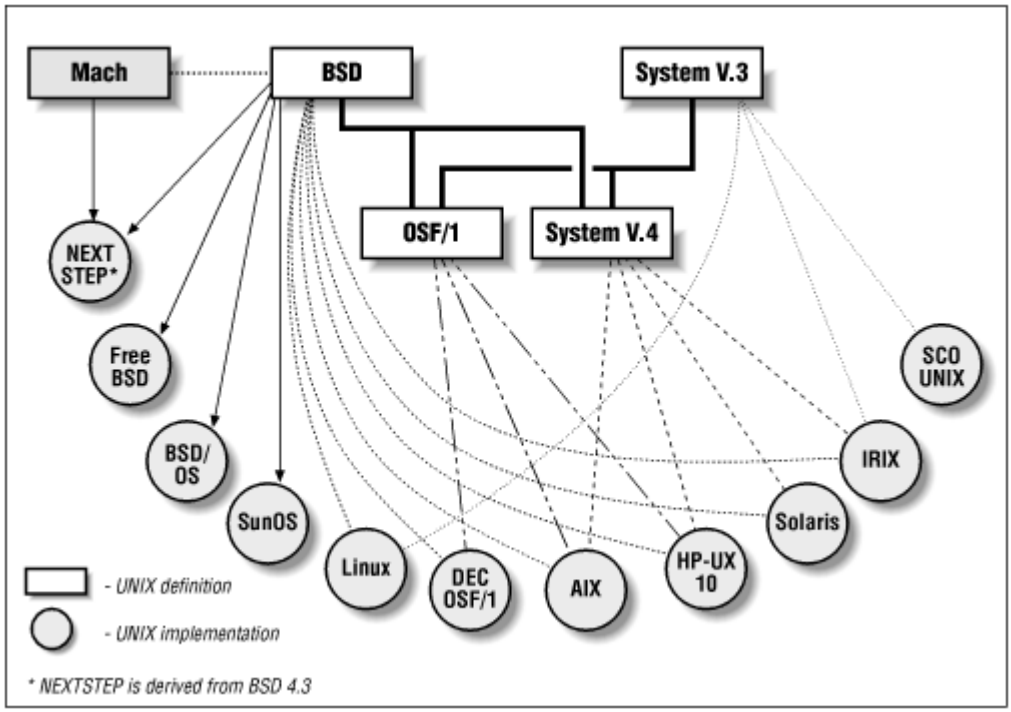
- ***BSD :***

Từ 1970 Computer Science Research Group của University of California tại Berkeley (UCB) xuất bản nhiều phiên bản UNIX, được biết đến dưới tên Berkeley Software Distribution, hay BSD. Cải biến của PDP-11 được gọi là 1BSD và 2BSD. Trợ giúp cho các máy tính của Digital Equipment Corporation VAX được đưa vào trong 3BSD. Phát triển của VAX được tiếp tục với 4.0BSD, 4.1BSD, 4.2BSD, và 4.3BSD

Trước 1992, UNIX là tên thuộc sở hữu của AT&T. Từ 1992, khi AT&T bán bộ phận Unix cho Novell, tên Unix thuộc sở hữu của X/Open foundation. Tất cả các hệ điều hành thỏa mãn một số yêu cầu đều có thể gọi là Unix. Ngoài ra, Institute of Electrical and Electronic Engineers (IEEE) đã thiết lập chuẩn "An Industry-Recognized Operating Systems Interface Standard based on the UNIX Operating System." Kết quả cho ra đời POSIX.1 (cho giao diện C) và POSIX.2 (cho hệ thống lệnh trên Unix)

Kết lại, vấn đề chuẩn hóa UNIX vẫn còn rất xa kết quả cuối cùng. Nhưng đây là quá trình cần thiết có lợi cho sự phát triển của ngành tin học nói chung và sự sống còn của HDH UNIX nói riêng.

Các phiên bản của Unix



c. Lịch sử phát triển của Linux và giới thiệu các phân phối (distribution) Linux ngày nay

Linux là một HDH dạng UNIX (Unix-like Operating System) chạy trên máy PC với bộ điều khiển trung tâm (CPU) Intel 80386 hoặc các thế hệ sau đó, hay các bộ vi xử lý trung tâm tương thích như AMD, Cyrix. Linux ngày nay còn có thể chạy trên các máy Macintosh hoặc SUN Sparc . Linux thỏa mãn chuẩn POSIX.1.

Linux được viết lại toàn bộ từ con số không, tức là không sử dụng một dòng lệnh nào của Unix, để tránh vấn đề bản quyền của Unix, tuy nhiên hoạt động của Linux hoàn toàn dựa trên nguyên tắc của hệ điều hành Unix. Vì vậy nếu một người nắm được Linux, thì sẽ nắm được UNIX. Nên chú ý rằng giữa các Unix sự khác nhau cũng không kém gì giữa Unix và Linux.

Năm 1991 Linus Torvalds, sinh viên của đại học tổng hợp Helsinki, Phần lan, bắt đầu xem xét Minix, một phiên bản của Unix, làm ra với mục đích nghiên cứu cách tạo ra một hệ điều hành Unix chạy trên máy PC với bộ vi xử lý Intel 80386.

Ngày 25/8/1991, Linus cho ra version 0.01 và thông báo trên comp.os.minix của Internet về chương trình của mình.

1/1992, Linus cho ra version 0.12 với shell và C compiler. Linus không cần Minix nữa để recompile HDH của mình. Linus đặt tên HDH của mình là Linux.

1994, phiên bản chính thức 1.0 được phát hành.

Quá trình phát triển của Linux được tăng tốc bởi sự giúp đỡ của chương trình GNU (GNU's Not Unix), đó là chương trình phát triển các Unix có khả năng chạy trên nhiều platform. Đến hôm nay, cuối 2001, phiên bản mới nhất của Linux kernel là 2.4.2-2, có khả năng điều khiển các máy đa bộ vi xử lý và rất nhiều các tính năng khác.

d. Vấn đề bản quyền của GNU project

Các chương trình tuân theo GNU Copyleft or GPL (General Public License) có bản quyền như sau :

1. Tác giả vẫn là sở hữu của chương trình của mình.
2. Ai cũng được quyền bán copy của chương trình với giá bất kỳ mà không phải trả cho tác giả ban đầu.
3. Người sở hữu chương trình tạo điều kiện cho người khác sao chép chương trình nguồn để phát triển tiếp chương trình.

e. ***Tại sao lại sử dụng Linux ?***

Linux là miễn phí (free). Đối với chúng ta hôm nay không quan trọng vì ngay WindowsNT server cũng "free". Nhưng trong tương lai, khi chúng ta muốn hòa nhập vào thế giới, khi chúng ta muốn có một thu nhập chính đáng cho người lập trình, hiện tượng sao chép trộm phần mềm cần phải chấm dứt. Khi đó, "free" là một thông số rất quan trọng để chọn Linux.

Linux rất ổn định. Trái với suy nghĩ truyền thống "của rẻ là của ôi ", Linux từ những phiên bản đầu tiên cách đây 5-6 năm đã rất ổn định. Ngay cả server Linux phục vụ những mạng lớn (hàng trăm máy trạm) cũng hoạt động rất ổn định.

Linux đầy đủ. Tất cả những gì bạn thấy ở IBM, SCO, Sun … đều có ở Linux. C compiler, perl interpeter, shell , TCP/IP, proxy, firewall, tài liệu hướng dẫn ... đều rất đầy đủ và có chất lượng. Hệ thống các chương trình tiện ích cũng rất đầy đủ .

Linux là HDH hoàn toàn 32-bit. Như các Unix khác, ngay từ đầu, Linux đã là một HDH 32 bits. Hiện nay đã có những phiên bản Linux 64 bits chạy trên máy Alpha Digital hay Ultra Sparc.

Linux rất mềm dẻo trong cấu hình. Linux cho người sử dụng cấu hình rất linh động, ví dụ như độ phân dải màn hình Xwindow tùy ý, dễ dàng sửa đổi ngay cả kernel …

Linux chạy trên nhiều máy khác nhau từ PC 386, 486 tự lắp cho đến SUN Sparc.

Linux được trợ giúp. Ngày nay, với các server Linux sử dụng dữ liệu quan trọng, người sử dụng hoàn toàn có thể tìm được sự trợ giúp cho Linux từ các công ty lớn. IBM đã chính thức chào bán IBM server chạy trên Linux. Tài liệu giới thiệu Linux ngày càng nhiều, không thua kém bất cứ một HDH nào khác.

Với nguồn tài liệu phong phú, chương trình từ kernel cho đến các tiện ích miễn phí và bộ mã nguồn mở, Linux là người bạn đồng hành lý tưởng cho những ai muốn đi vào HDH chuyên nghiệp UNIX và công cụ tốt nhất cho công tác đào tạo CNTT trong các trường đại học.

Các phiên bản của Linux. Các phiên bản của HDH Linux được xác định bởi hệ thống số dạng X.YY.ZZ. Nếu YY là số chẵn => phiên bản ổn định. YY là số lẻ => phiên bản thử nghiệm .

Các phân phối (distribution) của Linux quen biết là RedHat, Debian, SUSE, Slakware, Caldera …

Chú ý phân biệt số phiên bản của hệ điều hành (Linux kernel) với phiên bản của các phân phối (ví dụ RedHat 6.0 với kernel Linux 2.2.5-15).

II. Hệ thống tiến trình (process) của Linux. Điều khiển các tiến trình.:

Linux là một HDH đa người sử dụng, đa tiến trình. Linux thực hiện tất cả các công việc của người sử dụng cũng như của hệ thống bằng các tiến trình (process). Do đó, hiểu

được cách điều khiển các tiến trình đang hoạt động trên HDH Linux rất quan trọng, nhiều khi có tính chất quyết định, cho việc quản trị hệ thống.

➢ **Định nghĩa** : Tiến trình (process) là một chương trình đơn chạy trên không gian địa chỉ ảo của nó . Cần phân biệt tiến trình với lệnh vì một dòng lệnh trên shell có thể sinh ra nhiều tiến trình.

➢ **Dòng lệnh :**

nroff -man ps.1 | grep kill | more

sẽ sinh ra 3 tiến trình khác nhau.

Có 3 loại tiến trình chính trên Linux :

- Tiến trình với đối thoại (Interactive processes) : là tiến trình khởi động và quản lý bởi shell, kể cả tiến trình forthground hoặc background.

- Tiến trình batch (Batch processes) : Tiến trình không gắn liền đến bàn điều khiển (terminal) và được nằm trong hàng đợi để lần lượt thực hiện.

- Tiến trình ẩn trên bộ nhớ (Daemon processes) : Là các tiến trình chạy dưới nền (background). Các tiến trình này thường được khởi động từ đầu. Đa số các chương trình server cho các dịch vụ chạy theo phương thức này. Đây là các chương trình sau khi được gọi lên bộ nhớ, đợi thụ động các yêu cầu chương trình khách (client) để trả lời sau các cổng xác định (cổng là khái niệm gắn liền với giao thức TCP/IP BSD socket. Chúng ta sẽ giải thích rõ trong phần TCP/IP). Hầu hết các dịch vụ trên Internet như mail, Web, Domain Name Service … chạy theo nguyên tắc này. Các chương trình được gọi là các chương trình daemon và tên của nó thường kết thúc bằng ký tự "d" như named, inetd … Ký tự "d" cuối được phát âm rời ra như "đê " trong tiếng việt. Ví dụ named được phát âm là "nêm đê".

Cách đơn giản nhất để kiểm tra hệ thống tiến trình đang chạy là sử dụng lệnh ps (process status). Lệnh ps có nhiều tùy chọn (option) và phụ thuộc một cách mặc định vào người login vào hệ thống. Ví dụ :

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

cho phép hiển thị các tiến trình liên quan tới một người sử dụng hệ thống.

Cột đầu tiên là PID (Process IDentification). Mỗi tiến trình của Linux đều mang một số ID và các thao tác liên quan đến tiến trình đều thông qua số PID này. Gạch nối – trước bash để thông báo đó là shell khởi động khi người sử dụng login.

Để hiển thị tất cả các process, ta có thể sử dụng lệnh ***ps –a***. Một người sử dụng hệ thống bình thường có thể thấy tất cả các tiến trình, nhưng chỉ có thể điều khiển dược các tiến trình của mình tạo ra. Chỉ có super-user mới có quyền điều khiển tất cả các tiến trình của hệ thống Linux và của người khác. Lệnh ***ps –ax*** cho phép hiển thị tất cả các tiến trình, ngay cả những tiến trình không gắn liền đến có bàn điều khiển (tty). Chúng ta có thể coi các tiến trình đang chạy cùng với dòng lệnh đầy đủ để khởi đ ộng tiến trình này bằng ***ps –axl.*** Lệnh ***man ps*** cho phép coi các tham số tự chọn khác của lệnh ***ps*** .

✍ **Dừng một tiến trình, lệnh kill :** Trong nhiều trường hợp, một tiến trình có thể bị treo, một bàn phím điều khiển không trả lời các lệnh từ bàn phím, một chương trình server cần nhận cấu hình mới, card mạng cần thay đổi địa chỉ IP …, khi đó chúng ta phải dừng (kill) tiến trình đang có vấn đề . Linux có lệnh **kill** để thực hiện các công tác này. Trước tiên bạn cần phải biết PID của tiến trình cần dừng thông qua lệnh **ps**. Xin nhắc lại chỉ có super-user mới có quyền dừng tất cả các tiến trình, còn người sử dụng chỉ được dừng các tiến trình của mình. Sau đó, ta sử dụng lệnh

kill -9 PID_của_ tiến_trình

Tham số –9 là gửi tín hiệu dừng không điều kiện chương trình. Chú ý nếu bạn logged vào hệ thống như root, nhập số PID chính xác nếu không bạn có thể dừng một tiến trình khác. Không nên dừng các tiến trình mà mình không biết vì có thể làm treo máy hoặc dịch vụ.

Một tiến trình có thể sinh ra các tiến trình con trong quá trình hoạt động của mình. Nếu bạn dừng tiến trình cha, các tiến trình con cũng sẽ dừng theo, nhưng không tức thì . Vì vậy phải đợi một khoảng thời gian và sau đó kiểm tra lại xem tất cả các tiến trình con có dừng đúng hay không. Trong một số hãn hữu các trường hợp, tiến trình có lỗi nặng không dừng được, phương pháp cuối cùng là khởi động lại máy.

Khi dó tiến trình sau lệnh nohup sẽ không bị dừng lại khi bạn logout.

✍ Lệnh **at** : Bên cạnh đó, Linux có các lệnh cho phép thực hiện các tiến trình ở các thời điểm mong muốn. Lệnh **at** cho phép thực hiện một tiến trình vào thời điểm nhập trong dòng lệnh.

$ at 1:23<Return>

lp /usr/sales/reports/*<Return>

echo "Files printed, Boss" | mail boss@company.com<Return>

<^D>

Dấu ^D có nghĩa là cần giữ phím <Ctrl>, sau đó nhấn phím D và bỏ cả 2 phím cùng một lúc.

Sau khi bạn kết thúc lệnh ***at***, dòng thông báo giống như sau sẽ hiện ra màn hình

job 756001.a at Sat Dec 21 01:23:00 2000

Số 756001.a cho phép tham chiếu tới công tác (job) đó, để dùng nếu bạn muốn xóa job đó bởi lệnh

at –r job_number

Lệnh này có thể khác với các phiên bản khác nhau. Ví dụ đối với RedHat 6.2 lệnh xóa một job là ***atrm job_number*** . Trong mọi trường hợp coi manpage để biết các lệnh và tham số cụ thể.

Bạn có thể dùng quy tắc chuyển hướng (redirect) để lập lịch trình cho nhiều lệnh cùng một lúc

at 10:59 < tập_lệnh

trong đó, ***tập_lệnh*** là một tập tin dạng text có các lệnh. Để kiểm tra các tiến trình mà bạn đã nhập vào, dùng lệnh ***at –l***

✍ Lệnh **batch** : Khác với lệnh **at** là tiến trình được thực hiện vào các thời điểm do người sử dụng chọn, lệnh **batch** để cho hệ thống tự quyết định khi nào tiến trình được thực hiện dựa trên mức độ tải của hệ thống. Thường là các tiến trình batch được thi hành khi máy bận dưới 20%. Các tiến trình in ấn, cập nhật dữ liệu lớn … rất thích hợp với kiểu lệnh này. Cú pháp của **batch** như sau :

$ batch<Return>

lp /usr/sales/reports/*<Return>

echo "Files printed, Boss" | mail boss@company.com<Return>

<^D>

Các lệnh **at** và **batch** cho phép lập kế hoạch thực hiện tiến trình một lần. Linux còn cho phép lập kế hoạch có tính chất chu kỳ thông qua lệnh ***cron*** (viết tắt của chronograph) và các tập tin crontabs. Chương trình ***cron*** được khởi động ngay từ đầu với khởi động của hệ thống. Khi khởi động, ***cron*** xem có các tiến trình trong hàng đợi nhập vào bởi lệnh **at**, sau đó xem xét các các tập tin crontabs xem có tiến trình cần phải thực hiện hay không rồi "đi ngủ ":-) . ***Cron*** sẽ "thức dậy" mỗi phút để kiểm tra xem có phải thực hiện tiến trình nào không. Super-user và user đều có thể đặt hàng các tiến trình sẽ được cho phép thực hiện bởi cron. Để làm điều này, bạn cần tạo một tập tin text theo cú pháp của cron như sau.

Phút giờ ngày_của_tháng tháng_của_năm ngày_của_tuần lệnh

0 8 * * 1 /u/sartin/bin/status_report

cho phép /u/sartin/bin/status_report được thực hiện vào 8giờ 00 phút các thứ hai.

Mỗi hàng chứa thời gian và lệnh. Lệnh sẽ được ***cron*** thực hiện tại thời điểm ghi ở trước trên cùng dòng đó. Năm cột đầu liên quan tới thời gian có thể thay thế bằng dấu sao "*" với ý nghĩa là "với mọi". Các giá trị có thể cho các trường là :

minute (0-59)

hour (0-23)

day of month (1-31)

month of year (1-12)

day of week (0-6, 0 is Sunday)

Command (rest of line)

Sau đó dùng lệnh **crontab** để cài đặt tập tin lệnh vào thư mục /usr/spool/cron/crontabs. Mỗi người sử dụng sẽ có một tập tin crontab trùng tên mình (user name) để lưu tất cả các lệnh cần thực hiện theo chu kỳ trong thư mục này. Cú pháp sử dụng **crontab**:

crontab tên_tập_tin_lệnh

Sau khi hiểu rõ cấu trúc các tập tin, người sử dụng có thể tự tạo các tập tin crontab và đặt vào thư mục theo đúng quy định của **cron** mà không cần phải dùng **crontab.** Điều này còn đúng cho đại đa số các dịch vụ khác. Các chương trình của Unix thường tuân theo một quy tắc là có các tập tin cấu hình dạng text. Các tập tin này hoàn toàn có thể được tạo ra bằng các phần mềm soạn thảo văn bản. Các chương trình tiện ích chỉ là công cụ trợ giúp nếu người sử dụng muốn và không mang tính chất bắt buộc. Để có thể trở thành một người quản trị Unix thực thụ, bạn đọc nên tập dần cung cách cấu hình trực tiếp không thông qua các tiện ích.

Lệnh top. Lệnh top cho phép hiển thị sự hoạt động của các tiến trình, đặc biệt là các thông tin về tài nguyên hệ thống cũng như việc sử dụng tài nguyên đó của từng tiến trình. Với lệnh đơn giản top, ta sẽ có

```
11:09am  up 46 days, 17:44,  2 users,  load average: 0.08, 0.03, 0.01
63 processes: 61 sleeping, 2 running, 0 zombie, 0 stopped
CPU states:  0.1% user,  0.0% system,  0.0% nice, 99.8% idle
Mem:   126644K av,  121568K used, 5076K free,  0K shrd,   25404K buff
Swap:  136544K av, 9836K used,  126708K free  36040K cached

  PID USER     PRI  NI  SIZE  RSS SHARE STAT %CPU %MEM   TIME COMMAND
27568 tnminh    11   0  1052 1052   836 R     0.1  0.8   0:00 top
    1 root       0   0   124   72    68 S     0.0  0.0   0:25 init
    2 root       8   0     0    0     0 SW    0.0  0.0   0:00 kevent
```

Số % máy rảnh (idle) in đậm trên là rất quan trọng. Một máy rảnh dưới 50% là một máy quá tải và cần được xem xét. Lệnh top còn cho phép theo dõi xem có tiến trình nào chiếm dụng quá nhiều thời gian CPU cũng như truy cập đĩa không.

Ngoài ra, một số lệnh khác như vmstat. Mpstat, sar, iostat ... cũng cho phép xem xét với các mục đích khác nhau hoạt động của máy chủ.

Inetd và các dịch vụ mạng :

Unix có hai cách để tổ chức các dịch vụ mạng: hoặc là khởi động ngay từ đầu chương trình server dưới dạng daemon, hoặc là để công tác khởi động chương trình dịch vụ theo yêu cầu (khi có yêu cầu kết nối) với sự trợ giúp của một tiến trình daemon khác là **inetd** (đọc là inét đê). Trong trường hợp đầu, ta cần cho mỗi dịch vụ ít nhất một daemon và tài nguyên của hệ thống bị sử dụng ngay cả khi không có yêu cầu kết nối. Còn trong trường hợp sau ta cần một daemon cho tất cả các dịch vụ. Tài nguyên hệ thống chỉ thực sự bị chiếm dụng khi có yêu cầu kết nối. Vì vậy, chương trình server dạng daemon thường trực được dùng cho các dịch vụ có yêu cầu kết nối thường xuyên như DNS, mail, Web ; còn sơ đồ qua inetd dành cho các dịch vụ với tần số thưa như ftp, telnet, secure shell …

Chương trình **inetd,** còn gọi là super-server, được sử dụng để khởi động các daemon phục vụ các dịch vụ mạng. **inetd** đợi các nối mạng sau một số cổng được quy định bởi tập tion cấu hình /etc/inetd.conf. RedHat Linux 7.1 sử dụng tập tin /etc/xinetd.conf và các tập tin trong thư mục /etc/xinet.d. Khi có yêu cầu kết nối, **inetd** sẽ gọi chương trình server tương ứng để thiết lập các kết nối và phục vụ khách hàng. Thông thường, **inetd** được khởi động ngay từ đầu bởi các script dùng cho khởi động máy. **inetd** sẽ đọc file cấu hình /etc/inetd.conf khi được gọi lên bộ nhớ . Sau đây là một vài dòng của tập tin /etc/inetd.conf

```
# <service_name> <sock_type> <proto> <flags> <user> <server_path> <args>
# Echo, discard, daytime, and chargen are used primarily for testing.
# To re-read this file after changes, just do a 'killall -HUP inetd'
#time   stream  tcp    nowait  root    internal
#time   dgram   udp    wait    root    internal
#
```

```
# These are standard services.
#
ftp    stream  tcp    nowait  root   /usr/sbin/tcpd  in.ftpd -l -a
telnet stream  tcp    nowait  root   /usr/sbin/tcpd  in.telnetd
```

Bên cạnh tập tin cấu hình **/etc/inetd.conf**, tập tin **/etc/services** cũng được **inetd** sử dụng để biết các cổng (port) của các chương trình server. Ví dụ một đoạn của tập tin **/etc/services**

```
ftp-data      20/tcp
ftp           21/tcp
fsp           21/udp      fspd
ssh           22/tcp              # SSH Remote Login Protocol
ssh           22/udp      # SSH Remote Login Protocol
telnet        23/tcp
# 24 - private
smtp          25/tcp      mail
# 26 - unassigned
time          37/tcp      timserver
time          37/udp      timserver
rlp           39/udp            resource    # resource location
nameserver    42/tcp      name          # IEN 116
whois         43/tcp      nicname
re-mail-ck    50/tcp      # Remote Mail Checking Protocol
re-mail-ck    50/udp      # Remote Mail Checking Protocol
domain        53/tcp      nameserver    # name-domain server
domain        53/udp      nameserver
```

Hai tập tin **/etc/inetd.conf** và **/etc/services** quan hệ mật thiết với nhau. Cột đầu tiên bao gồm tên các dịch vụ mạng và cần phải giống nhau. Một dịch vụ muốn được hoạt động nhờ inetd phải khai báo cổng mà nó đợi khách hàng thông qua /etc/services và dòng lệnh khởi động nó trong /etc/inetd.conf. Muốn tắt một dịch vụ, ta chỉ cần đặt dấu chú thích # trước dòng miêu tả dịch vụ và khi đó, inetd sẽ không biết và không gọi dịch vụ đó nữa. Như các bạn đọc nhận thấy nội dung của cột <*server_path*> <*args*> cho các dịch vụ là */usr/sbin/tcpd in.telnetd*. Chương trình **tcpd** được **inetd** gọi lên trước để làm một số công tác kiểm tra và ghi log trước khi chương trình dịch vụ thực được gọi lên. Cụ thể là tcpd sẽ sử dụng

Cột <*flags*> cho biết chương trình **inetd** có phải đợi (wait) hay không (nowait) kết nối kết thúc trước khi "tiếp" một kết nối khác. Ví dụ trên với telnet cho thấy nhiều chương trình khách có thể được phục vụ một lúc qua cùng một cổng telnet 23. Tất nhiên chương trình server telnet cũng phải được thiết kế thích hợp với kiểu làm việc đa khách hàng này.

Cột <*user*> quy định quyền của tiến trình khi nó được chạy trên bộ nhớ. Trong trường hợp có nghi ngờ về tính bảo mật của một dịch vụ, ta có thể giảm quyền của nó bằng cách thay đổi nội dung của cột này.

Qua ví dụ trên ta thấy dịch vụ ftp sẽ được **inetd** gọi lên thông qua dòng lệnh ***/usr/sbin/tcpd in.ftpd -l –a*** khi có một chương trình khách hàng dùng giao thức TCP gọi qua cổng 21.

Đọc thêm. **Tiến trình được sinh ra như thế nào?** Trên một máy chủ Unix, thường có hàng chục tiến trình đang đồng thời hoạt động. Trên những máy chủ lớn và bận bịu, có thể có hàng ngàn tiến trình cùng lúc. Vậy tiến trình được hình thành như thế nào ?

Nếu con người được sinh ra bởi con người thì tiến trình cũng sinh ra bởi tiến trình. Chỉ có một điều khác là phải cần 2 người làm cha mẹ mới có trẻ em (trừ những dự định clone người hiện nay), còn tiến trình thì chỉ có một tiến trình cha. Khi hệ thống khởi động, tiến trình đầu tiên là **init**. Sau đó, init sẽ sinh ra các tiến trình khác cần thiết cho sự hoạt động của hệ thống. Ví dụ mỗi khi ta đăng nhập hệ thống, tiến trình login sau khi kiểm tra mật khẩu sẽ sinh ra một tiến trình **shell** để người sử dụng có thể làm việc thông qua các dòng lệnh của **shell**. Có 2 lệnh liên quan tới việc hình thành các tiến trình là lệnh **fork** và **execve**. Lệnh **fork** cho phép hình thành một tiến trình con giống hệt tiến trình cha và cả hai sau đó cùng được song song hoạt động và được HĐH đối xử như nhau. Hai tiến trình này chỉ khác nhau về PID và người ta có thể biết rằng hiện đang ở tiến trình bằng cách xem giá trị trở về của lệnh fork: nếu bằng 0, ta đang ở tiến trình cha, nếu khác 0 thì đó là PID của tiến trình con. Lệnh **execve** thì thay thế một tiến trình bằng một tiến trình khác. Như vậy, nếu ta đang có một tiến trình A, tiến trình B có thể sinh ra từ A bằng cách A fork ra A' rồi trong A' ta dùng lệnh execve để thay thế A' bằng B.

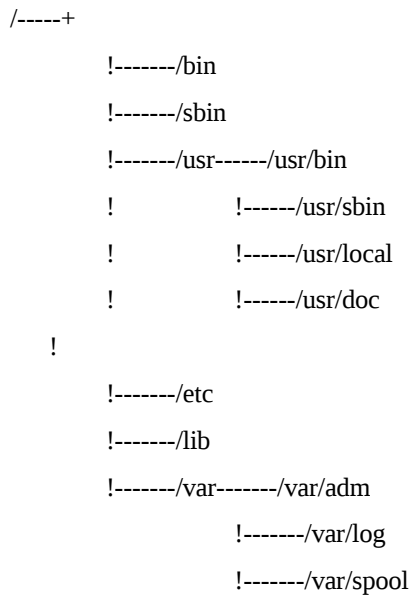
Đoạn chương trình sau cho phép hiểu rõ hơn các miêu tả trên

```
if (fork() == 0) {
        /* I am the child, I will become ls /usr/bin */
        execl("/bin/ls","ls","/usr/bin", (char *) 0);
}
else {
        /* I'm parent, do whatever perent's sopposed to do*/
}
```

Các biến tấu của **execve** tạo thành một họ các hàm exec (exec family). Linux có thêm **clone** để tạo các **threads** (tiểu tiến trình). Trong trường hợp hệ thống quá tải, lệnh **fork** sẽ không thành công do tài nguyên đã bị vét cạn. Khi đó ta sẽ có thông báo lỗi trên màn hình gắn trực tiếp với máy chủ và máy chủ cần được xem xét sửa chữa hoặc nâng cấp.

III. Hệ thống tập tin của Linux :

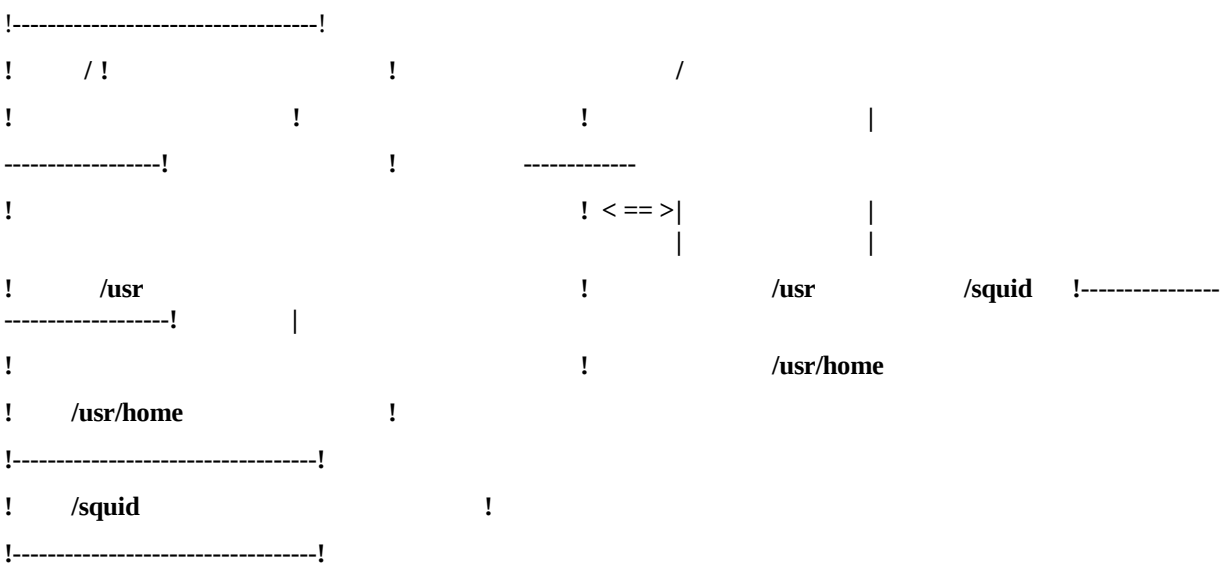
III.1 Cây thư mục của Linux. Đối với hệ điều hành Linux, không có khái niệm các ổ đĩa khác nhau. Sau quá trình khởi động, toàn bộ các thư mục và tập tin được "gắn" lên (mount) và tạo thành một hệ thống tập tin thống nhất, bắt đầu từ gốc '/'



Hình trên là cây thư mục của đa số các Unix. Với cây thư mục trên ta không thể nào biết được số lượng ổ đĩa cứng, các phân mảnh (partition) của mỗi đĩa và sự tương ứng giữa các phân mảnh và thư mục như thế nào.

Chúng ta có thể chia đĩa cứng thành nhiều phân mảnh (partition). Mỗi partition là một hệ thống tập tin (file system) độc lập. Sau đó, các hệ thống tập tin này được ‘gắn ‘ (mount) vào hệ thống tập tin thống nhất của toàn hệ thống. Chúng ta hoàn toàn có thể gắn thêm một đĩa cứng mới, format rồi mount vào hệ thống tập tin dưới tên một thư mục nào đó và tại một điểm (mount point) nào đó. Đối với các chương trình chạy trên Unix, không hề có khái niệm một thư mục nằm ở đĩa nào hay partition nào.

Hình sau đây cho thấy sự tương quan giữa vị trí vật lý trên đĩa và vị trí logic trong cây tập tin.



Thư mục **/usr/home** là thư mục con của **/usr** trong cây thư mục, nhưng trên đĩa vật lý, đây là hai phân mảnh (partition) cạnh nhau.

Hệ thống tập tin được OS Linux mount trong quá trình khởi động tuân theo các thông số ghi trong tập tin **/etc/fstab** (một lần nữa, nếu bạn nắm vững cú pháp của tập tin này, bạn có thể thay đổi nó thông qua một chương trình soạn thảo văn bản text bất kỳ và có một kiểu khởi động hệ thống tập tin như bạn muốn)

```
[tnminh@pasteur tnminh]$ more /etc/fstab
/dev/hda2  /                ext2            defaults  1 1
/dev/hda3  swap             swap            defaults    0 0
/dev/fd0   /mnt/floppy      ext2            noauto      0 0
/dev/cdrom          /mnt/cdrom      iso9660  noauto,ro   0 0
none       /proc    proc     defaults    0 0
none       /dev/pts devpts   mode=0622   0 0
```

Cột 1 (fs_spec) : các trang thiết bị (device) cần mount

- 2 (fs_file) : điểm treo (mount point)
- 3 (fs_vfstype) : Kiểu của hệ thống tập tin,
- 4 (fs_mntops) : các options. Default = mount khi khởi động, ro = read only, user nếu cho phép user mount hệ thống tập tin này ...
- 5 (fs_freq) : hiện thị (dumped) hay không hệ thống tập tin
- 6 (fs_passno) : có cần kiểm tra hay không bởi fsck

Tập tin **/etc/fstab** được sử dụng bởi chương trình **mount** trong quá trình khởi động của Linux. Dòng

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

cho phép ổ CDROM có thể mount theo ý muốn của người dùng (không mount automatic) và gắn vào **/mnt/crdom** với kiểu hệ thống tập tin iso9660 với mục đích chỉ đọc. Nếu không có từ khóa **user** thì chỉ có **root** mới được quyền mount ổ CDROM. Với tập tin **/etc/fstab** như trên thì lệnh mount/unmount ổ CDROM sẽ là :

mount /dev/cdrom hay *umount /dev/cdrom*

Mount không có thông số cho phép hiển nội dung tập tin /etc/mtab = những hệ thống tập tin đã được mounted.

```
[root@pasteur tnminh]# mount
/dev/hda2 on / type ext2 (rw)
none on /proc type proc (rw)
none on /dev/pts type devpts (rw,mode=0622)
/dev/hda1 on /home/tnminh/minh type vfat (rw)
```

So sánh với

```
[root@pasteur tnminh]# more /etc/mtab
/dev/hda2 / ext2 rw 0 0
none /proc proc rw 0 0
none /dev/pts devpts rw,mode=0622 0 0
/dev/hda1  /home/tnminh/minh                vfat                rw                0                0
```

Ở đây chúng ta thấy 2 dòng đặc biệt :

```
none /proc proc rw 0 0
none /dev/pts devpts rw,mode=0622 0 0
```

/dev chứa những tập tin đặc biệt : tập tin thiết bị ngoại vi. Hệ thống Linux sử dụng các tập tin này để truy xuất dữ liệu đến các thiết bị ngoại vi. Như vậy, Linux giao tiếp đến các thiết bị ngoại vi giống như với các tập tin. Ví dụ **/dev/psaux** được dùng để giao tiếp với chuột, **/dev/hda1** để giao tiếp với phân mảnh 1 của đĩa cứng master của controler số 0...

```
brw-rw----  1 root    disk     3,  1 May  6  1998 hda1
crw-rw-r--  1 root    root    10,  1 May  6  1998 psaux
crw-------  1 root    tty      4, 64 Oct  3 15:55 ttyS0
crw-------  1 root    tty      4, 65 May  6  1998 ttyS1
crw-------  1 root    tty      4, 66 May  6  1998 ttyS2
crw-------  1 root    tty      4, 67 May  6  1998 ttyS3
```

Ký tự cột đầu tiên **'b'** để thông báo kiểu giao tiếp block (cho thiết bị như ổ đĩa), **'c'** – giao tiếp kiểu ký tự (cho thiết bị như bàn phím, chuột …).

Tóm lại, ta nhận thấy cây thư mục của Unix cũng giống như cây thư mục của MS DOS hay Windows.

/proc là hệ thống tập tin ảo cho phép đọc các thông tin của các process trên bộ nhớ. Để thực tập , ta có thể dùng **ps** để coi các tiến trình và thấy các tập tin tương ứng trong **/proc** như ví dụ sau :

```
[oracle@appserv]$ ps ax|grep 582
 582 ?        S      0:17 nmbd -D
 8724 pts/5    S      0:00 grep 582
[oracle@appserv]$ cd /proc/582
[oracle@appserv 582]$ ls -l
ls: exe: Permission denied
ls: root: Permission denied
ls: cwd: Permission denied
total 0
-r--r--r--    1 root     root            0 Oct 12 17:39 cmdline
lrwx------    1 root     root            0 Oct 12 17:39 cwd
-r--------    1 root     root            0 Oct 12 17:39 environ
lrwx------    1 root     root            0 Oct 12 17:39 exe
dr-x------    2 root     root            0 Oct 12 17:39 fd
pr--r--r--    1 root     root            0 Oct 12 17:39 maps
-rw-------    1 root     root            0 Oct 12 17:39 mem
lrwx------    1 root     root            0 Oct 12 17:39 root
-r--r--r--    1 root     root            0 Oct 12 17:39 stat
-r--r--r--    1 root     root            0 Oct 12 17:39 statm
-r--r--r--    1 root     root            0 Oct 12 17:39 status
[oracle@appserv 582]$ more cmdline
nmbd-D
[oracle@appserv 582]$
```

Các ký tự in đậm trong ví dụ trên cho phép thấy được mối liên hệ giữa /proc và tiến trình đang chạy. Các thao tác trên có thể thực hiện bởi một user bất kỳ.

III.2 Quyền truy cập, sở hữu tập tin và thư mục của Linux (directory and file permission and ownership) :

Do Linux là một hệ điều hành multitasking và multiuser, nhiều người cùng có thể sử dụng một máy Linux và một người có thể cho chạy nhiều chương trình khác nhau. Có hai vấn đề lớn được đặt ra : quyền sở hữu các dữ liệu trên đĩa và phân chia tài nguyên hệ thống như CPU, RAM ... giữa các process. Chúng ta sẽ bàn về sở hữu các tập tin và các quyền truy xuất tập tin.

Tất cả các tập tin và thư mục của Linux đều có người sở hữu và quyền truy nhập. Bạn có thể đổi các tính chất này cho phép nhiều hay ít quyền truy nhập hơn đối với một tập tin hay thư mục. Quyền của tập tin còn cho phép xác định tập tin có là một chương trình (application) hay không (khác với MSDOS và MSWindows xác định tính chất này qua phần mở rộng của tên tập tin) .Ví dụ với lệnh ***ls –l*** chúng ta có thể thấy

-rw-r—r— 1 fido users 163 Dec 7 14:31 myfile

Cột đầu chỉ ra quyền truy cập tập tin

Cột 2 chỉ số liên kết (link) đối với tập tin hay thư mục

Cột 3, 4 chỉ chủ sở hữu và nhóm sở hữu

Cột 5 chỉ độ dài của tập tin

Cột 6 chỉ thời gian thay đổi cuối cùng

Cột 7 là tên tập tin hay thư mục

Trong ví dụ trên, các ký tự **-rw-r—r—** biểu thị quyền truy cập của tập tin **myfile**. Sở hữu của **myfile** là **fido** và nhóm sở hữu **myfile** là **users** . **Fido** được quyền đọc và ghi vào **myfile**, còn những người sử dụng của nhóm **users** và những người khác chỉ được quyền đọc **myfile.**

Linux cho phép người sử dụng xác định các quyền đọc (read), viết (write) và thực hiện (execute) cho từng đối tượng trong nhóm sau : sở hữu (the owner), nhóm (the group), và những người còn lại ("others" (everyone else)).

Quyền đọc cho phép bạn đọc nội dung của tập tin. Đối thư mục quyền đọc cho phép bạn sử dụng lệnh **ls** để xem nội dung của thư mục.

Quyền viết cho phép bạn thay đổi nội dung hay xóa tập tin. Đối với thư mục, quyền viết cho phép bạn tạo ra, xóa hay thay đổi tên trong thư mục.

Quyền thực hiện cho phép bạn gọi chương trình lên bộ nhớ bằng cách nhập từ bàn phím tên của tập tin. Đối với thư mục, bạn chỉ có thể vào thư mục bởi lệnh **cd** nếu bạn có quyền thực hiện với thư mục .

Xem xét lại ví dụ trên :

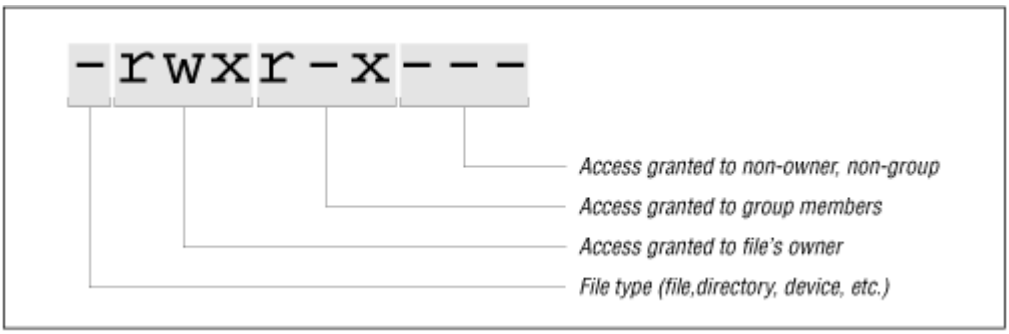
-rw-r—r— 1 fido users 163 Dec 7 14:31 myfile

Ký tự đầu tiên của quyền là ký tự "**–"** ám chỉ rằng đó là một tập tin bình thường. Nếu **myfile** là một thư mục, ta sẽ thấy vào dó ký tự **d**. Ngoài ra còn có **c** cho thiết bị ngoại vi dạng ký tự (như bàn phím), **b** cho thiết bị ngoại vi dạng block (như ổ đĩa cứng).

Chín ký tự tiếp theo chia thành 3 nhóm, cho phép xác định quyền của ba nhóm sở hữu (owner), nhóm (group) và còn lại (other). Mỗi cặp ba này cho phép xác định quyền đọc, viết và thực hiện theo thứ tự kể trên. Quyền đọc viết tắt là "**r**" ở vị trí đầu tiên, quyền viết viết tắt bằng "**w"** ở vị trí thứ hai và vị trí thứ ba là quyền thực hiện ký hiệu bằng chữ "**x"** . Nếu một quyền không được cho, tại vị trí đó sẽ có ký tự "**-"** .

Trong trường hợp của tập tin **myfile**, sở hữu có quyền **rw** tức là đọc và viết. **Myfile** không phải là một chương trình. **Nhóm** cùng với **còn lại** chỉ có quyền đọc tập tin (read-only). Hình sau cho ta thấy rõ hơn cách "đọc" quyền truy cập đối với tập tin.

Quyền truy cập cơ bản của tập tin



Song song với cách ký hiệu miêu tả bằng ký tự như ở trên, quyền thao tác tập tin còn có thể cho dưới dạng 3 số . Đối với **myfile**, quyền đó là 644. Điều quan trọng là phải hiểu cách ký hiệu bằng số vì nó liên quan đến việc thay đổi các quyền sau này. Các số có thể nhận tất cả các giá trị từ 0 đến 7. Số đầu tiên miêu tả quyền của sở hữu, số thứ hai cho nhóm và số thứ ba cho còn lại.

Mỗi số là tổng của các quyền theo quy tắc sau :

read permission	4
Write permission	2
Execute permission	1

Vì vậy, một tập tin với quyền 751 có nghĩa là sở hữu có quyền read, write, và execute bằng 4+2+1=7, Nhóm có quyền read và execute bằng 4+1=5, và còn lại có quyền execute bằng 1.

Nếu chúng ta xem kỹ, chúng ta sẽ thấy mọi số từ 0 đến 7 đều tương ứng với một tổ hợp duy nhất các quyền truy nhập tập tin.

0 or —-: No permissions at all

4 or r—: read-only

2 or -w-: write-only (rare)

1 or —x: execute

6 or rw-: read and write

5 or r-x: read and execute

3 or -wx: write and execute (rare)

7 or rwx: read, write, and execute

Nếu bạn quen với hệ nhị phân, hãy suy nghĩ bằng hệ thống nhị phân. Khi đó, rwx sẽ như số nhị phân 3 bits. Nếu quyền được cho, số nhị phân tương ứng sẽ bằng 1, ngược lại, nó sẽ bằng 0. Ví dụ r-x sẽ là số nhị phân 101, và theo hệ thập phân sẽ là 4+0+1, hay 5. —x sẽ tương ứng 001, hay 0+0+1 = 1 …

Chú ý: Người sử dụng có quyền đọc thì có quyền copy tập tin và tập tin sao chép sẽ thuộc sở hữu người làm copy như minh họa sau

[tnminh@backup tnminh]$ ls -l /etc/passwd

```
-rw-r--r--    1 root     root    1113 Oct 13 12:30 /etc/passwd
[tnminh@backup tnminh]$  cp /etc/passwd ./
[tnminh@backup tnminh]$ ls -l passwd
-rw-r--r--    1 tnminh   admin       1113 Oct 15 10:37 passwd
```

Các quyền d9ịnh khi tạo tập tin. Khi một tập tin hay thư mục được tạo ra, permission mặc định sẽ được xác định bởi các quyền trừ bớt bởi các quyền hiển thị bằng **umask**

```
[tnminh@pasteur tnminh]$ umask
002
[tnminh@pasteur tnminh]$ echo tao mot file > tmp
[tnminh@pasteur tnminh]$ ls -l
total 5472
-rw-rw-r--   1 tnminh   tnminh        13 Oct  3 21:55 tmp
 [tnminh@pasteur /etc]$ umask 022
[tnminh@pasteur tnminh]$ echo tao mot file khac >tmp1
[tnminh@pasteur tnminh]$ ls -l
-rw-rw-r--   1 tnminh   tnminh        13 Oct  3 21:55 tmp
-rw-r--r--   1 tnminh   tnminh        18 Oct  3 21:59 tmp1
```

Trong ví dụ trên, quyền mặc định lúc đầu xác định bởi umask=002. Khi đó, tập tin tmp tạo ra sẽ có quyền là 664 và đó chính là bù đến 6 của umask. Quyền thực hiện chương trình cần được gán cố ý bởi người sử dụng hay các chương trình biên dịch. Sau đó ta đổi giá trị của umask thành 022 và tập tin tạo ra có quyền 644. Giá trị mặc định của các quyền thường được gán mỗi khi người sử dụng login vào hệ thống thông qua các tập tin khởi tạo biến môi trường như **.profile, .bashrc.** Đứng trên quan điểm bảo mật hệ thống, giá trị 024 là tốt nhất, nó cho người cùng nhóm có quyền đọc và không cho quyền nào với những người khác.

- Lệnh chown, chgrp và chmod :

Đây là nhóm lệnh được sử dụng rất phổ biến, cho phép thay quyền truy cập của tập tin hay thư mục. Chỉ có chủ sở hữu và superuser mới có quyền thực hiện các lệnh này.

Cách dùng lệnh : **chmod quyền_truy_cập_mới tên_file.**

darkstar:~$ ls -l myfile

-rw-r—r— 1 fido users 114 Dec 7 14:31 myfile

darkstar:~$ chmod 345 myfile

darkstar:~$ ls -l myfile

—wxr—r-x 1 fido users 114 Dec 7 14:31 myfile

darkstar:~$ chmod 701 myfile

darkstar:~$ ls -l myfile

-rwx——x 1 root users 114 Dec 7 14:31 myfile

Ví dụ thay đổi và hiện thị cho thấy sự thgay đổi quyền truy cập tập tin **myfile** . Chú ý là ta có quyền cấp phát quyền thực hiện (execute) mà không cần biết là tập tin có phải là một chương trình hay không.

Phương pháp thay đổi tuyệt đối này có một số ưu điểm vì nó là cách định quyền tuyệt đối, kết quả cuối cùng không phụ thuộc vào quyền truy cập trước đó của tập tin. Đồng thời, dễ nói "thay quyền tập tin thành bảy-năm-năm" thì dễ hơn là "thay quyền tập tin thành đọc-viết-thực hiện, đọc-thực hiện, đọc-thực hiện"

Bạn cũng có thể thay đổi quyền truy nhập một cách tương đối và dễ nhớ. Để chỉ ra nhóm quyền nào cần thay đổi, bạn có thể sử dụng **u (user), g (group), o (other), hay a (all)**. Tiếp theo đó là dấu + để thêm quyền và – để bớt quyền. Cuối cùng là bản thân các qyuyền viết tắt bởi r,w,x. Ví dụ như để bổ sung quyền thực hiện cho nhóm và còn lại, ta nhập vào dòng lệnh

darkstar:~$ chmod go+x myfile

Đây là cách thay đổi tương đối vì kết quả cuối cùng phụ thuộc vào quyền đã có trước đó mà lệnh này không liên quan đến. Trên quan điểm bảo mãt hệ thống, cách thay đổi tuyệt đối dẫn đến ít sai sót hơn. Thay đổi quyền truy cập của một thư mục cũng được thực hiện giống như đối với một tập tin. Chú ý là nếu bạn không có quyền thực hiện (execute) đối với một thư mục, bạn không thể thay đổi thư mục **cd** vào thư mục đó. Mọi người sử dụng có quyền viết vào thư mục đều có quyền xóa tập tin trong thư mục đó, không phụ thuộc vào quyền của người đó đối với tập tin. Vì vậy, đa số các thư mục có quyền **drwxr-xr-x**. Như vậy chỉ có người sở hữu của thư mục mới có quyền tạo và xóa tập tin trong thư mục. Ngoài ra, thư mục còn có một quyền đặc biệt, đó là cho phép mọi người đều có quyền tạo tập tin trong thư mục, mọi người đều có quyền thay đổi nội dung tập tin trong thư mục, nhưng chỉ có người tạo ra mới có quyền xóa tập tin. Đó là sticky bit cho thư mục. Thư mục /tmp thường có sticky bit bật lên

drwxrwxrwt 7 root root 16384 Oct 21 15:33 tmp

Ta thấy chữ **t** cuối cùng trong nhóm các quyền, thể hiện cho sticky bit của **/tmp**

***III.3 Liên kết (link) tập tin*:** Trong Unix có 2 hình thức liên kết hoàn toàn khác nhau, đó là **hard link** và **soft link** hay **symbolic link**. Hard link cho phép tạo một tên mới cho tập tin. Các tên này có vai trò hoàn toàn như nhau và tập tin chỉ bị hoàn toàn xóa bỏ khi hard link cuối cùng của nó bị xóa. Lệnh **ls –l** cho phép hiển thị số hard link đến tập tin. Symbolic link có chức năng giống như **shortcut** của MS Windows. Khi ta đọc/ghi soft link, ta đọc/ghi tập tin; khi ta xóa symbolic link, ta chỉ xóa symbolic link và tập tin được giữ nguyên. Link được tạo bởi lệnh **ln** . Tự chọn **ln –s** cho phép tạo symbolic link. Ví dụ

```
[tnminh@pascal tnminh]$ls -l
-rw-------  1 tnminh  pkt      517 Oct 27 12:00 mbox
drwxr-xr-x  2 tnminh  pkt     4096 Aug 31 17:50 security
[tnminh@pascal tnminh]$ln –s mbox mybox
[tnminh@pascal tnminh]$ln –s security securproj
[tnminh@pascal tnminh]$ln –l
-rw-------  1 tnminh  pkt      517 Oct 27 12:00 mbox
lrwxrwxrwx  1 tnminh  pkt  4 Oct 27 17:57     mymail -> mbox
lrwxrwxrwx  1 tnminh  pkt  8 Oct 27 17:57     secrproj -> security
drwxr-xr-x  2 tnminh  pkt     4096 Aug 31 17:50 security
```

[tnminh@pascal tnminh]$

Bạn đọc có thể thấy khá rõ kết quả của symbolic link qua thí dụ trên.

Symbolic link rất có nhiều ứng dụng. Ví dụ như một tập tin XXX của một chương trình YYY nằm trong thư mục /var/ZZZ. Nếu phân mảnh của /var/ZZZ bị quá đầy, ta có thể "sơ tán" XXX qua một thư mục khác thuộc phân mảnh khác và tạo một link thế vào đó mà chương trình YYY vẫn không hề "hay biết" vì nó vẫn truy cập đến /var/ZZZ/XXX như thường lệ.

Các thao tác trên tập tin. Các lệnh cơ bản cho phép làm việc với các tập tin là

ls : xem nội dung một thư mục

cp : copy tập tin/thư mục

mv : di chuyển tập tin/thư mục

rm : xoá tập tin. rm –rf cho phép xóa thư mục không rỗng và tất cả các thư mục con

mkdir : tạo thư mục mới

more, **less, cat** : xem nội dung tập tin

diff : so sánh nội dung hai tập tin

touch : ghi lại thời gian tập tin hoặc tạo tập tin mới nếu chưa có

vi, pico, emacs ... cho phép soạn thảo văn bản, lập trình.

Tiện ích **mc** giống như Norton Cammander trên DOS cho phép thao tác dễ dàng các tập tin, kể cả thao tác với tập tin của máy khác thông qua mạng.

Lệnh tar và gzip. Đây là 2 lệnh cho phép lưu trữ (backup) cũng như sao chép dữ liệu chủ yếu của Unix. Lệnh tar cho phép đóng gói một hệ thống tập tin thành một tập tin với phần đuôi .tar. Cấu trúc của hệ thống tập tin này (hệ thống các thư mục con) được lưu trữ và phụ hồi trong quá trình mở gói (untar). Lệnh gzip cho phép nén (compact) một tập tin. Thông thường, để lưu trữ, người ta tar các dữ liệu, rồi sau đó zip tập tin kết quả của tar. Quá trình phục hồi làm theo quy trình ngược lại. Lệnh tar và tùy chọn phổ biến là :

Để tạo ra một lưu trữ tar

```
tar -cv dir_name > dir_name.tar
```

Khi đó, toàn bộ nội dung và cấu trúc thư mục con của dir_name sẽ được lưu trong tập tin dir_name.tar. Ngược lại, để phục hồi lưu trữ, ta dùng

```
tar -xvf dir_name.tar
```

Lệnh df, du. Lệnh **df** (disk free) cho phép hiển thị tình trạng sử dụng của các ổ đĩa như dung lượng, đã sử dụng và dung lượng còn rảnh. Lệnh **du thư_mục** cho phép hiển thị độ lớn của thư mục đó.

Lệnh fsck. Linux đòi hỏi cần được dừng theo đúng quy trình, tức là phải shutdown máy trước khi tắt điện bằng công tắc. Mỗi khi máy Linux bị tắt đột ngột, hệ thống tập tin bị hư hại và cần được sửa chữa qua dịch vụ **fsck** (file system check). Thông thường, fsck sửa chữa thành công một cách tự động hệ thống tập tin và Linux khởi động lại dễ dàng. Tuy nhiên, nếu hư hỏng quá nặng, Linux sẽ chuyển qua chế độ single mode để sửa chữa. Khi đó, chúng ta chỉ có thể làm việc với máy trực tiếp trên bàn phím của nó. Ta sẽ phải sử dụng lệnh

fsck thiết_bị_đĩa_cứng_bị_hư

để sửa chữa. Ví dụ như fsck /dev/hda1 sẽ sửa pphân đoạn đầu của ổ đĩa master của controller số 0.

IV. <u>Quá trình khởi động và kết thúc của UNIX</u> :

Như thông lệ, khi một máy tính được khởi động, sau khi kiểm tra các thiết bị phần cứng gắn trên máy tính qua các chương trình kiểm tra ghi trong ROM, hệ điều hành được tải lên bộ nhớ. Công tác đầu tiên của hệ điều hành là kiểm tra các thiết bị ngoại vi và tải các chương trình điều khiển (driver) cần thiết lên bộ nhớ. Sau các công tác này, bắt đầu

giai đoạn định hình hệ thống và mỗi hệ điều hành, thậm trí mỗi phiên bản của một hệ điều hành thực hiện một khác. Chúng tôi xin giới thiệu cách thức khởi động và cấu hình hệ thống của Linux RedHat 6.x . Các Unix khác như SUN OS 6.x, 7.x cũng có hệ thống các tập tin khởi động và cơ chế hoạt động gần giống như Linux RedHat 6.x.

Tập tin đầu tiên mà hệ điều hành xem xét đến là **/etc/inittab**

```
[tnminh@proxy tnminh]$ ls –l /etc/ inittab
-rw-r--r--   1 root     root        1756 May 30 15:51 inittab
[tnminh@proxy tnminh]$ more /etc/inittab
#
# inittab      This file describes how the INIT process should set up
#              the system in a certain run-level.
# Default runlevel. The runlevels used by RHS are:
#   0 - halt (Do NOT set initdefault to this)
#   1 - Single user mode
#   2 - Multiuser, without NFS (The same as 3, if you do not have networking)
#   3 - Full multiuser mode
#   4 - unused
#   5 - X11
#   6 - reboot (Do NOT set initdefault to this)
#
id:3:initdefault:
```

Mức (level) làm việc mặc định được quy định trong tập tin này. Ví dụ trên cho thấy mức mặc định là mức 3 ở dòng cuối cùng. Unix nói chung có 7 mức hoạt động khác nhau từ 0 đến 6. Mức 0 là để shutdown hệ thống. Mức 1 là đơn người sử dụng (single user) và thường được dùng để sửa chữa lỗi hệ thống tập tin, mức 2, 3 là hai mức cho đa người sử dụng, mức 6 dùng để reboot hệ thống, mức 4,5 do người sử dụng tự thiết kế cho mình. Tương ứng với các mức trên, trong thư mục /etc/rc.d có các thư mục rc0.d – rc6.d, chứa các tập tin khởi động trong từng mức (rc là viết tắt của run command). RedHat 6.x có thư mục **/etc/rc.d/init.d** chứa tất cả các tập tin khởi động. Thường các tập tin này là các shell script (tập hợp lệnh shell) hoặc perl script (như Debian Linux chẳng hạn). Trong các thư mục rc?.d chỉ có các liên kết hình thức (symbolic link) đến các tập tin khởi động trong **/etc/rc.d/init.d**. SUN OS 7.0 lại đặt thực sự các script khởi động vào các thư mục **rc?.d** thay vì symbolic link.

```
[tnminh@proxy /etc/rc.d]$ ls -l
total 22
drwxr-xr-x   2 root     root         1024 May 10 09:44 init.d
-rwxr-xr-x   1 root     root         2722 Apr 15  1999 rc
-rwxr-xr-x   1 root     root          693 Aug 17  1998 rc.local
-rwxr-xr-x   1 root     root         9822 Apr 14  1999 rc.sysinit
drwxr-xr-x   2 root     root         1024 May  3 01:44 rc0.d
drwxr-xr-x   2 root     root         1024 May  3 01:44 rc1.d
drwxr-xr-x   2 root     root         1024 May  3 01:44 rc2.d
drwxr-xr-x   2 root     root         1024 May 10 09:47 rc3.d
drwxr-xr-x   2 root     root         1024 May  3 01:44 rc4.d
drwxr-xr-x   2 root     root         1024 May  3 01:44 rc5.d
drwxr-xr-x   2 root     root         1024 May  3 01:44 rc6.d
```

Trong các thư mục **rc?.d**, các script bắt đầu bằng **S** (start) được sử dụng khi khởi động, còn các script bắt đầu từ **K** (kill) dùng để dừng các tiến trình trước khi qua một mức hoạt động khác.

Toàn bộ các tập tin này quyết định cấu hình làm việc của một máy Unix sau khi hoàn thành quá trình khởi động. Việc khởi động hệ thống các dịch vụ cũng thực hiện thông qua cơ chế như đã miêu tả trên.

Lệnh **init số_mức** cho phép chuyển giữa các mức của hệ thống. Ví dụ

```
[root@proxy /etc/rc.d]# init 1
```

cho phép chuyển hệ thống từ mức hiện hành qua mức 1 để sửa chữa. Sau đó **init 3** cho phép quay về mức 3 đa người dùng.

Khi chúng ta đánh lệnh **shutdown**, toàn bộ hệ thống chuyển về mức 0 và chúng ta dừng hệ thống. *Chú ý luôn shutdown hay halt hệ thống trước khi tắt công tắc điện.*

V. Quản lý người sử dụng :

Trong quá trình cài đặt Linux chúng ta khởi tạo người sử dụng **root** cho hệ thống. Đây là superuser, tức là người sử dụng đặc biệt và không có giới hạn nào về quyền hạn đối với root. Sử dụng quyền root chúng ta rất thấy thoải mái vì chúng ta có thể làm được thao tác mà không phải lo lắng gì đết xét quyền truy cập này hay khác. Tuy nhiên, khi hệ thống bị sự cố do một lỗi lầm nào đó, chúng ta mới thấy sự nguy hiểm khi làm việc như root. Bạn thử hình dung toàn bộ các Email của một mail server của toàn công ty bị xóa do đánh một lệnh sai thì tác hại sẽ lơn đến mức nào (đừng đến gặp trực tiếp giám đốc khi báo tin này mà nên thông báo qua điện thoại để tránh một cái … bạt tai, ☹). Vì vậy, hãy chỉ dùng quyền root khi bạn không có cách nào khác.

Cần phân biệt bạn đang login như root hay người sử dụng thường thông qua dấu nhắc của shell.

```
login: tnminh
```

```
Password:
Last login: Sat Oct 28 14:30:15 from 172.16.10.199
 [tnminh@pascal tnminh]$ su –l root
Password:
[root@pascal /root]#
```

Dòng 4 với dấu $ cho thấy ta đang kết nối như một người sử dụng thường ***(tnminh).*** Dòng cuối cùng với dấu # cho thấy bạn đang thực hiện các lệnh như **root.** Lệnh **su** cho phép bạn thay đổi **login** dưới một user khác mà không phải **logout** rồi **login** lại.

Bạn cần tạo các tài khoản (**account**) cho người sử dụng thường sớm nhất có thể được (đầu tiên là cho bản thân bạn). Với những server quan trọng và có nhiều dịch vụ khác nhau, thậm trí bạn có thể tạo ra các superuser thích hợp cho từng dịch vụ để tránh dùng root cho các công tác này. Ví dụ như superuser cho công tác **backup** chỉ cần chức năng đọc (read-only) mà không cần chức năng ghi.

Tập tin /etc/passwd. Tập tin **/etc/passwd** đóng một vai trò sống còn đối với một hệ thống Unix. Mọi người đều có thể đọc được tập tin này nhưng chỉ có **root** mới có quyền thay đổi nó. Tập tin /etc/passwd được lưu dưới dạng text hiển như đại đa số các tập tin cấu hình của Unix.

```
[oracle@appserv oracle]$ more /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:
...
tnminh:x:501:501:TNMinh:/home/tnminh:/bin/bash
```

Mỗi user được lưu trong một dòng gồm 7 cột.

Cột 1 : tên người sử dụng

Cột 2 : mã liên quan đến passwd cho Unix chuẩn và ‘x’ đối với Linux. Linux lưu mã này trong một tập tin khác **/etc/shadow** mà chỉ có root mới có quyền đọc.

Cột 3:4 : user ID:group ID

Cột 5: Tên đầy đủ của người sử dụng. Một số phần mềm phá password sử dụng dữ liệu của cột này để thử đoán password.

Cột 6: thư mục cá nhân

Cột 7: chương trình sẽ chạy đầu tiên sau khi login (thường là shell) cho user

Tập tin mở đầu bởi superuser **root.** Chú ý là tất cả những user có user ID = 0 đều là root!!! Tiếp theo là các user hệ thống. Đây là các user không có thật và không thể login vào hệ thống. Cuối cùng là các user bình thường.

Tập tin /etc/shadow. Unix truyền thống lưu các thông tin liên quan tới mật khẩu để đăng nhập (login) ở trong /etc/passwd. Tuy nhhiên, do đây là tập tin phải đọc được bởi tất cả mọi người do một số yêu cầu cho hoạt động bình thường của hệ thống (như chuyển User ID thành tên khi hiển thị trong lệnh ls chẳng hạn) và nhìn chung các user đặt mật khẩu

"yếu", do đó hầu hết các Unix phiên bản mới đều lưu mật khẩu trong một tập tin khác /etc/shadow và chỉ co root được quyền đọc tập tin này.

Chú ý: Theo cách xây dựng mã hóa mật khẩu, chỉ có 2 cách phá mật khẩu là vét cạn (brute force) và đoán. Phương pháp vét cạn, theo tính toán chặt chẽ, là không thể thực hiện nổi vì đòi hỏi thời gian tính toán quá lớn, còn đoán thì chỉ tìm ra những mật khẩu ngắn, hoặc "yếu", ví dụ như những từ tìm thấy trong từ điển như god, darling ...

Tạo user (account) mới **:** Để tạo một một account, bạn có thể sử dụng lệnh **adduser** (hoặc useradd tùy vào phiên bản). Tất nhiên là bạn phải làm thao tác này dưới quyền root (dấu nhắc #)

```
[root@appserv oracle]# /usr/sbin/adduser foo
[root@appserv oracle]# passwd foo
Changing password for user foo
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully
[root@appserv oracle]#
```

Sau khi bạn tạo xong user bởi dòng đầu tiên của ví dụ trên, user **foo** vẫn chưa kết nối được vì thiếu password. Bạn phải khởi tạo password cho **foo** bởi lệnh **passwd foo** như thấy ở trên.

Vì vấn đề an ninh của máy Unix này và kéo theo sự an toàn của toàn hệ thống mạng của bạn, rất quan trọng chọn đúng password. Một password gọi là đúng nếu :

- Có độ dài tối thiểu 8 ký tự.
- Phối hợp giữa chữ thường, chữ hoa, số và các ký tự đặc biệt
- Không liên quan đến tên tuổi, ngày sinh … của bạn và người thân
- Không có trong từ điển

Trong ví dụ trên, bạn khởi tạo người dùng và không quan tâm gì đến nhóm (group) của người dùng. Rất tiện lợi nếu bạn tập hợp nhiều người dùng vào chung một nhóm có cùng một chức năng và cùng chia sẻ nhau dữ liệu. Khi bạn tạo người sử dụng như trên, Linux sẽ tạo cho mỗi người một nhóm. Đọc tập tin **/etc/passwd** ta thấy

```
[root@appserv oracle]# more /etc/passwd|grep foo
foo:x:1012:1013::/home/foo:/bin/bash
[root@appserv oracle]#
```

foo là user số 1012 và thuộc nhóm 1013.

Xem tập tin /etc/group ta thấy

[root@appserv oracle]# more /etc/group

root:x:0:root

…….……..

users:x:100:

……..……

foo:x:1013:

và ta có thể kết nạp **foo** vào nhóm **users** bằng cách thay số 1013 bằng 100, là group ID của users.

Xóa user (account) : Lệnh **userdel** dùng để xóa một user. Bạn cũng có thể xóa một user bằng cách xóa đi dòng dữ liệu tương ứng trong tập tin **/etc/passwd**.

VI. Quyền truy cập của tiến trình. Setuid và setgid.

Người sử dụng khi thao tác trên hệ thống và truy cập các tập tin đều thực hiện thông qua các tiến trình. Nhìn chung các tiến trình sẽ có quyền như người sử dụng nó. User foo khi đánh lệnh **more /etc/passwd** sẽ gọi lên bộ nhớ tiến trình more và tiến trình sẽ hiển thị t nội dung của /etc/passwd. Tiến trình more này sẽ có quyền truy cập như **foo** và do /etc/passwd cho phép tất cả đều đọc được và lệnh này thực hiện thành công. Tuy nhiên **more /etc/shadow** sẽ không thành công vì với quyền của foo, tiến trình more không thể hiển thị nội dung của /etc/shadow. Như ta biết, người sử dụng có thể thay đổi mật khẩu của bản thân mình thông qua lệnh **passwd.** Nếu vẫn theo cơ chế trên, lệnh passwd sẽ có quyền như người gọi nó và do đó không có quyền ghi vào tập tin /etc/shadow**.** Để giải quyết những yêu cầu là có những thao tác người sử dụng cần có quyền root, Unix đặt ra cơ chế setuid/setgid cho các chương trình. Theo cơ chế này, chương trình khi được người sử dụng (gọi là real user) gọi lên bộ nhớ sẽ có quyền như người sở hữu của tập tin chương trình (gọi là effective user). Ví dụ nếu ta coi chương trình **passwd** ta sẽ thấy

```
[tnminh@backup tnminh]$ ls -l /usr/bin/passwd
-r-s--x--x    1 root     root        13536 Jul 12  2000 /usr/bin/passwd
[tnminh@backup tnminh]$
```

Chương trình này có sở hữu là root và có một ký tự "**s**" đứng vào vị trí của "**x**". Ký tự này nói lên rằng **passwd** được **setuid** và cho dù ai gọi **passwd** lên bộ nhớ, **passwd** sẽ có quyền **root**. Hoàn toàn tương tự đối với setgid.

Chú ý: Trên quan điểm bảo mật hệ thống, các chương trình có setuid về quyền root là những điểm quan trọng cần xem xét vì đa số các xâm nhập hệ thống liên quan tới lỗi của các chương trình loại này.

VII. <u>Kết nối mạng thông qua TCP/IP</u>:

Chúng ta sẽ xem xét quá trình nối một máy Linux vào mạng Ethernet để trao đổi thông tin bằng giao thức TCP/IP trên Ethernet.

VII.1. HDH Linux và card mạng:

Để nối một máy Linux vào một mạng Ethernet, bạn cần phải có đầu tiên là một card mạng mà Linux đã có chương trình driver. Sau đây là một số mạng mà Linux có trợ giúp

(danh sách sau không đầy đủ và các phiên bản mới của Linux hỗ trợ rất nhiếu các card mạng khác nhau) :

3Com 3C509

3Com 3C503/16

Novell NE1000

Novell NE2000

Western Digital WD8003

Western Digital WD8013

Hewlett-Packard HP27245

Hewlett-Packard HP27247

Hewlett-Packard HP27250

Giả sử các bạn muốn gắn máy của mình vào một mạng LAN Ethernet và bạn đã có một card mạng. Vấn đề đầu tiên là sự nhận biết của Linux đối với card này. Nếu card của bạn là một card khá phổ biến như 3c509 của 3COM hay NE2000 của Novell, HDH Linux sẽ nhận biết sự hiện diện của card trong quá trình boot. Để biết xem kết quả nhận biết card mạng, ta có thể xem xét các thông báo của kernel Linux trong quá trình boot của hệ thống qua lệng **dmesg**

Freeing unused kernel memory: 60k freed

Adding Swap: 72572k swap-space (priority -1)

eth0: 3c509 at 0x300 tag 1, BNC port, address 00 a0 24 4f 3d dc, IRQ 10.

3c509.c:1.16 (2.2) 2/3/98 becker@cesdis.gsfc.nasa.gov.

eth0: Setting Rx mode to 1 addresses.

Hai dòng in đậm báo rằng card mạng 3c509 đã được kernel nhận biết. Trong trường hợp kernel không nhận biết card ☹, chúng ta phải làm lại kernel Linux và đặt module điều khiển (driver) của card vào trong kernel hay cấu hình ở chế độ load module.

Để cấu hình tiếp nối mạng qua TCP/IP chúng ta phải xác định rõ các thông tin liên quan đến địa chỉ IP của máy. Các thông tin cần biết là :

Địa chỉ IP của máy

Netmask

Địa chỉ của mạng

Broadcast

Địa chỉ IP của gateway

Chúng ta sẽ lần lượt điểm qua các khái niệm cơ bản trên và sẽ học sâu hơn trong phần TCP/IP của khóa học.

Địa chỉ IP của máy là một dãy 4 số viết dước dạng A.B.C.D, trong đó mỗi số nhận giá tri từ 0-255. Nếu máy của bạn kết nối một mạng nhỏ tại nhà do bạn thiết lập thì địa chỉ kiểu 192.168.1.D là một địa chỉ nên đặt, với D là các số khác nhau cho từng máy. Nếu máy của bạn sẽ hòa nhập với một mạng LAN đã có trước đó và bạn muốn kết nối với các máy khác thì hỏi người quản trị mạng về địa chỉ IP bạn có thể gán cho máy của mình cùng với tất cả các thông số tiếp theo.

Netmask. Tương tự như trên, nếu bạn tự quản, netmask sẽ là 255.255.255.0

Địa chỉ mạng. Nếu bạn tự quản, địa chỉ của mạng sẽ là 192.168.1.0

Broadcast. Nếu bạn tự quản, broadcast là 192.168.1.255

Địa chỉ gateway. Đây là địa chỉ của máy cho phép bạn kết nối với mạng LAN khác, tức là các máy tính với 3 số đầu của địa chỉ không giống bạn là 192.168.1. Bạn bỏ trống nếu bạn chỉ liên lạc với các máy cùng mạng 192.168.1.XXX. Chú ý là địa chỉ mạng của máy gateway bắt buộc phải trùng với địa chỉ mạng của bạn.

Sau khi đã xác định các thông số, ví dụ như

IP address = 192.168.1.15

Netmask = 255.255.255.0

suy ra network address = 192.168.1.0 và broadcast = 192.168.1.255

Gateway = 192.168.1.1

VII.2. Cấu hình card mạng:

➢ **Lệnh ifconfig.** Sau khi làm cho kernel nhận biết sự hiện diện của card mạng, công tác tiếp theo là cấu hình TCP/IP cho card. Trong quá trình cài đặt Linux Redhat 6.X, bình thường chúng ta đã được chương trình cài đặt hỏi và cấu hình hộ . Trong trường hợp khi chúng ta bổ sung card mạng sau khi Linux đã được cài đặt, chúng ta có thể sử dụng tiện ích **netconf** cho mục đích này hoặc chúng ta sử dụng lệnh **ifconfig** để tự cài đặt.

Lệnh **ifconfig** được sử dụng trong quá trình boot hệ thống để cấu hình các trang thiết bị mạng. Sau đó, trong quá trình vận hành, **ifconfig** được sử dụng cho debug, hoặc để cho người quản trị hệ thống thay đổi cấu hình khi cần thiết .

Lệnh **ifconfig** không có tùy chọn dùng để hiển thị cấu hình hiện tại của máy.

```
[root@pasteur tnminh]# /sbin/ifconfig
eth0      Link encap:Ethernet  HWaddr 00:A0:24:4F:3D:DC
          inet addr:192.168.2.20  Bcast:192.168.2.255  Mask:255.255.255.0
          UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Metric:1
          RX packets:531 errors:4 dropped:0 overruns:0 frame:4
          TX packets:1854 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          Interrupt:10 Base address:0x300

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:3924  Metric:1
          RX packets:1179 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1179 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
```

Để gán địa chỉ IP 193.105.106.10 cho card mạng Ethernet đầu tiên ta dùng lệnh

```
ifconfig eth0 193.105.106.10 netmask 255.255.255.0 broadcast 192.105.106.255
```

Linux cho phép bạn sử dụng bí danh (alias) cho card mạng, tức là cho phép bạn có nhiều địa chỉ IP cho cùng một card vật lý. Kết quả nhận được gần giống như bạn có gắn nhiều card vật lý lên máy. Do đó, bạn có thể dùng một card để nối với nhiều mạng logic khác nhau. Cú pháp của lệnh này là :

```
ifconfig eth0:0 208.148.45.58 netmask 255.255.255.248 broadcast 208.148.45.255 up
```

Các tập tin cấu hình của kết nối mạng là **/etc/sysconfig/network-scripts/ifcfg-eth*X*** với X là 0,1 ... hay 0:0, 0:1 .... Bạn có thể thay đổi cấu hình kết nối mạng bằng cách sửa đổi lại tập tin này bằng một chương trình soạn thảo text như **mc** chẳng hạn, sau đó khởi động lại kết nối mạng bằng

```
/etc/rc.d/init.d/network restart
```

Nhớ kiểm tra lại kết quả qua lệnh **ifconfig**.

➢ **Lệnh route.**

Lệnh Route cho phép làm các thao tác đến bảng dẫn đường (forwarding table) của kernel. Nó được sử đầu tiên để xác định đường dẫn cố định (static) đến những máy hoặc những mạng qua các card mạng ethernet đã được cấu hình trước đó bởi ifconfig.

Lệnh **route** không có tùy chọn (option) cho phép hiển thị bảng dẫn đường hiện tại của kernel (Lệnh **netstat –r** cũng có tác dụng tương tự)

```
[root@pasteur tnminh]# /sbin/route
Kernel IP routing table
Destination     Gateway   Genmask   Flags Metric Ref Use Iface
192.168.2.20    *   255.255.255.255 UH   0     0     0   eth0
192.168.2.0     *   255.255.255.0   U    0     0     0   eth0
127.0.0.0       *   255.0.0.0       U    0     0     0   lo
default       192.168.2.10   0.0.0.0   UG    0    0   0 eth0
```

Để chỉ ra rằng card mạng eth0 được nối với một mạng 208.148.45.56 ta dùng lệnh route như sau :

```
route add -net 208.148.45.56 eth0
```

Còn nếu chúng ta muốn sử dụng bí danh của card mạng để nối vào một mạng logic khác, ta có thể sử dụng lệnh

```
route add -net 193.105.106.0 eth0:0
```

Công tác cuối cùng là phải chỉ ra các địa chỉ của gateway mặc định.

```
route add default gw 193.105.106.1 metric 1
```

Biết sử dụng thành thạo cú pháp của 2 lệnh **ifconfig** và **route** rất quan trọng, nó cho phép các cán bộ quản trị thay đổi cấu hình kết nối mạng của một server một cách nhanh chóng và không phải khởi động lại máy. Vì vậy, server luôn sẵn sàng. Bạn cũng có thể sử dụng tiện ích **netconfig** để cấu hình liên kết mạng nếu chưa thành thạo nhiều cú pháp của các lệnh trên.

➢ **Lệnh ping.** Ứng dụng của lệnh này là để thử xem 2 máy có kết nối được với nhau chưa. Cú pháp cơ bản của lệnh rất đơn giản là *ping địa_chỉ_IP_máy_đích.* Ví dụ như

```
[tnminh@proxy tnminh]$ ping sun
PING sun.vnuhcm.edu.vn (172.16.1.4): 56 data bytes
64 bytes from 172.16.1.4: icmp_seq=0 ttl=255 time=0.1 ms
64 bytes from 172.16.1.4: icmp_seq=1 ttl=255 time=0.2 ms
```

64 bytes from 172.16.1.4: icmp_seq=2 ttl=255 time=0.1 ms

64 bytes from 172.16.1.4: icmp_seq=3 ttl=255 time=0.1 ms

--- sun.vnuhcm.edu.vn ping statistics ---

4 packets transmitted, 4 packets received, 0% packet loss

round-trip min/avg/max = 0.1/0.1/0.2 ms

[tnminh@proxy tnminh]$

Nếu 2 máy có thể liên lạc được với nhau, đồng thời chúng ta sẽ có trả lời cùng với thời gian trả lời để cho biết sự thông thoáng về mạng giữa 2 máy. Có thể nói, **ping** phải chạy trước tiên trước tất cả các hoạt động mạng khác.

Chú ý: Nên sử dụng ping –n để tránh trục trặc do dịch vụ DNS làm ảnh hưởng tới việc kết quả thử kết nối mạng.

➢ **Lệnh Traceroute.** Đây cũng là lệnh cho phép chẩn đoán hoạt động của mạng. Cú pháp của lệnh giống như lệnh **ping** nhưng kết quả không chỉ dừng ở sự trả lời mà còn chỉ ra các thiết bị trung gian nằm giữa 2 máy.

tnminh@nefertiti ~ > traceroute 203.162.44.33

traceroute to 203.162.44.33 (203.162.44.33): 1-30 hops, 38 byte packets

1 makeda.pasteur.fr (157.99.64.3), 1.66 ms, 1.66 ms, 1.66 ms

2 418.ATM4-0.GW21.Defense.OLEANE.NET (195.25.28.149), 5.0 ms, 4.17 ms, 4.17 m

3 FastEth0-0.GW16.Defense.OLEANE.NET (195.25.25.208), 4.17 ms, 4.17 ms, 4.17s

4 100.ATM6-1.GW2.Telehouse.OLEANE.NET (194.2.3.245), 5.0 ms, 5.0 ms, 5.0 ms

.....

14 210.132.93.210 (210.132.93.210), 849 ms (ttl=241!), 807 ms (ttl=241!), 970

s (ttl=241!)

15 202.167.121.195 (202.167.121.195), 905 ms !H 203.162.3.42 (203.162.3.42), 1

88 ms (ttl=242!)

Chú ý là khi chúng ta thử kết nối với một máy ở xa trong Internet, do nhiều mạng áp dụng các bức tường lửa (firewall) nên nhiều khi lệnh ping và traceroute không chạy nhưng trên thực chất là mạng vẫn thông.

VIII. X-Window. X-window (chú ý window không có "s" như Windows của Miscrosoft) là giao diện đồ họa của Unix. *X Window System* được phát triển tại Laboratory for Computer Science, *Massachusetts Institute of Technology* vào 1984.

Tuy nhiên, do Unix làhệ điều hành mạng nên phương thức hoạt động của X-widow cũng khác hẳn Windows của Microsoft. Cơ chế hoạt động của Xwindow được miêu tả trong sơ đồ sau:

Xserver (XF86-SVGA) Xclient (netscape, xterm)

User------- Máy trạm WS <------ *mạngTCP/IP*------→ Máy chủ S

(màn hình, bàn phím, chuột)

Tham gia vào mô hình X window có chương trình X server và X client. Nhìn chung, X server là chương trình chạy trên máy trạm làm việc WS của người sử dụng, còn X client chạy trên máy chủ S nằm xa người sử dụng. Chương trình Xserver trên máy trạm chịu trách nhiệm quản lý tài nguyên của máy trạm (màn hình, bàn phím, chuột) và thực hiện giao tiếp giữa người sử dụng và chương trình X client chạy trên máy chủ (nói chung là ở xa nhười sử dụng). Kết nối giữa X server và Xclient có thể thực hiện hoàn toàn trên TCP/IP qua mạng LAN cũng như WAN. Một Xserver có thể cho phép "hiển thị" nhiều Xclient ở nhiều máy khác nhau và đó là ưu điểm cơ bản của Xwindow. Xserver "nghe" tại cổng 6000 và Xclient mở một kết nối từ một cổng nào đó (lớn hơn 1023) về cổng 6000 của Xserver.

Trước khi kết nối, Xserver phải cho phép Xclient được quyền kết nối thông qua lệnh **xhost +địa_chỉ_máy_Xclient** trên màn hình của **Xserver.**

Để Xclient biết phải hiển thị đi đâu, ta cần thay đổi biến môi trường DISPLAY trên máy có Xclient qua lệnh **export DISPLAY=địa_chỉ_máy_Xserver:0.0.** Sau đó gọi chương trình Xclient, ví dụ xterm & hay netscape & (chú ý dấu & ở cuối cho phép chương trình chạy background).

Những công tác trên thực chất phải làm để cấu hình một Xserver là :

- Xác định nhà sản xuất, phiên bản của video controller. Qua đó xác định được chương trình Xserver. Trong nhiều trường hợp, XF86-SVGA là tương thích.
- Xác định màn hình để qua đ1o xác định các thông số về tốc độ quét dọc và ngang của màn hình
- Xác định độ phân giải của màn hình, đa số là 800x600 hay 1024x768
- Người sử dụng lựa chọn chương trình quản lý cửa sổ (Window Manager). Sự lựa chọn này phụ thuộc vào sở thích là chính. KDE và GNOME là 2 lựa chọn chính của Linux.

Rất may mắn là việc cài đặt giao diện Xwindow trên Linux hiện nay đã được tự động hóa rất nhiều. Trong trường hợp có trục trặc, lệnh **X –probeonly >/tmp/test 2>&1** cho phép chúng ta ghi lại toàn bộ các thông báo của Xserver vào tập tin /tmp/test và dùng cho xem xét tìm nguyên nhân trục trặc của Xserver. Ta thử xem trong ví dụ sau:

[root@backup X11]# more /tmp/t

XFree86 Version 3.3.6a / X Window System

.....

Configured drivers:

SVGA: server for SVGA graphics adaptors (Patchlevel 1): *(tên Xserver)*

s3_savage, NV1, STG2000, RIVA 128, RIVA TNT, RIVA TNT2,

... danh sách các video controller mà Xserver hỗ trợ .....

ct65550, ct65554, ct65555, ct68554, ct69000, ct64200, ct64300,

mediagx, V1000, V2100, V2200, p9100, spc8110, i740, i740_pci,

Voodoo Banshee, Voodoo3, i810, i810-dc100, i810e, smi, generic

XF86Config: /usr/X11R6/lib/X11/XF86Config ***(tập tin cấu hình)***

(**) stands for supplied, (--) stands for probed/default values

(**) XKB: keycodes: "xfree86"

.....

(**) Mouse: zaxismapping: (-)4 (+)5

(**) SVGA: Graphics device ID: "Cirrus Logic GD5480"

(Xserver nhận dạng được video controller – Đặc biệt quan trọng)

(**) SVGA: Monitor ID: "My Monitor"

.....

(**) SVGA: Using 16 bpp, Depth 16, Color weight: 565 ***(độ phân giải màn hình)***

(--) SVGA: Maximum allowed dot-clock: 100.000 MHz

(**) SVGA: Mode "800x600": mode clock = 40.000, clock used = 39.991

.....

Tiện ích **Xconfigurator** cũng có thể giúp ích cho bạn. Tất cả cấu hình của Xserver được ghi lại trong tập tin text **/etc/X11/XFConfig.** Bạn có thể tự thay đổi các thông số trong này nếu hiểu rõ ý nghĩa của chúng.

IX. Theo dõi hoạt động của hệ thống.

Tiện ích syslog. Syslog là tiện ích của Unix cho phép ghi nhận lại một cách tập trung và chuẩn (giữa các Unix) hoạt động của hệ thống các dịch vụ và thông báo của kernel. Thông qua syslog, ta có thể:

- Xem thông báo lỗi khi khởi động một chương trình dịch vụ, qua đó có thể sửa đổi lại cấu hình cho thích hợp
- Xem xét lại những gì đã xảy ra, dịch vụ nào đã khởi động lại, những ai đã thực hiện kết nối tại thời điểm nào ...
- Viết chương trình dịch vụ và gửi đến syslog các thông báo nhằm ghi lại hoạt động của chương trình của mình.

Để thực hiện các chức năng kể trên, syslog có một tiến trình server **syslogd** thường được khởi động cùng với hệ thống. Chương trình syslogd này đọc tập tin cấu hình /etc/syslog.conf để xác định phải ghi lại những gì và ở đâu. Ta thử coi /etc/syslog.conf (mọi user đều có thể đọc tập tin này)

```
[tnminh@backup X11]$ cat /etc/syslog.conf
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*                                 /dev/console
# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none        /var/log/messages
# The authpriv file has restricted access.
authpriv.*                              /var/log/secure
# Log all the mail messages in one place.
mail.*                                  /var/log/maillog
# Log cron stuff
cron.*                                  /var/log/cron
# Everybody gets emergency messages, plus log them on another
# machine.
*.emerg                                 *
# Save mail and news errors of level err and higher in a
# special file.
uucp,news.crit                          /var/log/spooler
# Save boot messages also to boot.log
local7.*                                /var/log/boot.log
# Receive log from center router
local0.*                                /var/log/router
[tnminh@backup X11]$
```

Hai khái niệm cần biết để hiểu tập tin này là facility và priority. Facility chính là danh sách các dịch vụ có thể log. Priority biểu thị cấp độ nghiêm trọng của thông báo. Cú pháp trong

tập tin /etc/syslog là **facility.priority.** Cột bên phải xác định địa chỉ ghi log. /etc/log/maillog là tập tin ghi lại thông báo của mail.*; dấu "*" ám chỉ thông báo sẽ được hiện ra ở tất cả các màn hình của các user; **@máy_khác** cho phép gửi thông báo tới syslogd của máy khác; | **chương_trình_xử_lý** cho phép gửi thông báo qua pipe "|" tới chương_trình_xử_lý...

Syslog facilities:

Name	**Facility**
Kern	Kernel
User	Regular user processes
Mail	Mail system
Lpr	Line printer system
auth	Authorization system, or programs that ask for user names and passwords (login, su, getty, ftpd, etc.)
daemon	Other system daemons
news	News subsystem
uucp	UUCP subsystem
local0... local7	Reserved for site-specific use
mark	A timestamp facility that sends out a message every 20 minutes

syslog Priorities

Priority	**Meaning**
emerg	Emergency condition, such as an imminent system crash, usually broadcast to all users
Alert	Condition that should be corrected immediately, such as a corrupted system database
Crit	Critical condition, such as a hardware error
Err	Ordinary error
warning	Warning
notice	Condition that is not an error, but possibly should be handled in a special way
Info	Informational message
debug	Messages that are used when debugging programs
None	Do not send messages from the indicated facility to the selected file. For example, specifying *.*debug;mail.none* sends all messages except mail messages to the

Name	**Facility**
Kern	Kernel
User	Regular user processes

selected file.

Chú ý nếu ta log một priority thì ta sẽ log toàn bộ priorities có độ nghiêm trọng cao hơn. Ví dụ nếu trong /etc/syslog có mail.info thì tất cả mail.notice hay mail.emerg đều được log.

Với một máy chủ bận rộn, tập tin log phình to rất nhanh và chúng ta đứng trước một bài toán là đồng thời phải giữ log lâu nhất có thể được để đề phòng sự cố và xóa log để có không gian đĩa cho máy hoạt động. **Logrotate** là một tiện ích giúp cho nhà quản trị xoay vòng (rotate), nén (compact) và gửi mail thông tin log. Logrotate đọc tập tin cấu hình /etc/logrotate.conf để biết chu kỳ quay vòng và các thông tin khác. Ví dụ sau

```
# sample logrotate configuration file
    errors sysadmin@my.org
    compress

    /var/log/messages {
       rotate 5
       weekly
       postrotate
                              /sbin/killall -HUP syslogd
       endscript
    }
```

cho thấy tập tin /var/log/message được lưu và quay vòng 5 tuần. Lệnh **/sbin/killall –HUP syslogd** cho phép khởi tạo lại tập tin /var/log/message vì tập tin cũ đã bị đổi tin và nén.

X. Cài đặt RedHat Linux. RedHat là một Linux distributor phổ biến nhất hiện nay. RedHat lựa chọn phiên bản kernel của Linux và các chương trình dịch vụ khác đóng thành các gói (tập tin có phần mở rộng .rpm) và lưu vào một hoặc hai đĩa CDROM. Phiên bản cuối cùng của Redhat Linux hiện nay là 7.1 với kernel 2.4.2-2. Các đĩa CDROM của RedHat đều có thể dùng để boot máy và điều này làm đơn giản rất nhiều quá trình cài đặt RedHat Linux. Có thể miêu tả sơ lược các bước cần phải qua khi cài đặt RedHat Linux là

1. sửa cấu hình máy để boot từ ổ CDROM
2. Đặt đĩa số 1 của RedHat Linux vào ổ CDROM và khởi động lại máy
3. Lựa chọn một phươngpháp cài đặt, ví dụ text
4. lựa chọn kiểu cài đặt , server hay trạm làm việc hay custom
5. Chia lại ổ đĩa cứng
6. lực chọn các gói sẽ cài đặt
7. để cho chương trình cài đặt Linux tự làm việc
8. thực hiện một số cấu hình nếu có yêu cầu hiển thị trên màn hình.

Sau khi Linux được cài xong, ta có thể thêm bớt các gói (package) vào/ra hệ thống thông qua tiện ích rpm (Redhat Package Manager). Các gói của RedHat thường nằm trong thư mục RPMS của CDROM. Để cài một gói X, ta dùng lệnh

rpm -i [install-options] <package_file>+

Các tập tin của gói X sẽ được rpm đặt vào các vị trí quy định đảm bảo cho sự hoạt động của dịch vụ X. Trong một số trường hợp chúng ta muốn cài "đè" lên gói đã cài trước và có trục trặc. Khi đó option --force cho phép thay gói cũ bằng gói mới.

Lệnh

rpm -e <package_name>+

cho phép xóa một gói đã cài đặt. Ngoài ra rpm còn cho phép xem xét tính toàn vẹn của một chương trình, nâng cấp một dịch vụ, liệt kê các tập tin trong một gói hoặc chỉ ra gói chức một tập tin ... Đây là một công cụ rất mạnh cho phép quản trị một máy Linux. Bạn đọc có thể đọc manpage của rpm để biết thêm.