

TÀI LIỆU ĐÀO TẠO
QUẢN TRỊ HỆ THỐNG LINUX 1



Tài liệu này được biên soạn theo tài liệu giảng dạy của Viện Linux (LPI)

HÀ NỘI 2006

GIỚI THIỆU GIẤY PHÉP CÔNG CỘNG GNU

BẢN DỊCH TIẾNG VIỆT CỦA GIẤY PHÉP CÔNG CỘNG GNU

Đây là bản dịch tiếng Việt không chính thức của Giấy phép Công cộng GNU. Bản dịch này không phải do Tổ chức Phần mềm Tự do ấn hành, và nó không quy định về mặt pháp lý các điều khoản cho các phần mềm sử dụng giấy phép GNU GPL -- chỉ có bản tiếng Anh gốc của GNU GPL mới có tính pháp lý. Tuy nhiên, chúng tôi hy vọng rằng bản dịch này sẽ giúp cho những người nói tiếng Việt hiểu rõ hơn về GNU GPL.

GIẤY PHÉP CÔNG CỘNG GNU (GPL)

Giấy phép công cộng GNU

Phiên bản 2, tháng 6/1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.

59 Temple Place - Suite 330, Boston, MA 02111-1307, USA

Mọi người đều được phép sao chép và lưu hành bản sao nguyên bản nhưng không được phép thay đổi nội dung của giấy phép này.

Lời nói đầu

Giấy phép sử dụng của hầu hết các phần mềm đều được đưa ra nhằm hạn chế bạn tự do chia sẻ và thay đổi nó. Ngược lại, Giấy phép Công cộng của GNU có mục đích đảm bảo cho bạn có thể tự do chia sẻ và thay đổi phần mềm tự do - tức là đảm bảo rằng phần mềm đó là tự do đối với mọi người sử dụng. Giấy phép Công cộng này áp dụng cho hầu hết các phần mềm của Tổ chức Phần mềm Tự do và cho tất cả các chương trình khác mà tác giả cho phép sử dụng. (Đối với một số phần mềm khác của Tổ chức Phần Mềm Tự do, áp dụng Giấy phép Công cộng Hạn chế của GNU thay cho giấy phép công cộng). Bạn cũng có thể áp dụng nó cho các chương trình của mình.

Khi nói đến phần mềm tự do, chúng ta nói đến sự tự do sử dụng chứ không quan tâm về giá cả. Giấy phép Công cộng của chúng tôi được thiết kế để đảm bảo rằng bạn hoàn toàn tự do cung cấp các bản sao của phần mềm tự do (cũng như kinh doanh dịch vụ này nếu bạn muốn), rằng bạn có thể nhận được mã nguồn nếu bạn có yêu

cầu, rằng bạn có thể thay đổi phần mềm hoặc sử dụng các thành phần của phần mềm đó cho những chương trình tự do mới; và rằng bạn biết chắc là bạn có thể làm được những điều này.

Để bảo vệ bản quyền của bạn, chúng tôi cần đưa ra những hạn chế để ngăn chặn những ai chối bỏ quyền của bạn, hoặc yêu cầu bạn chối bỏ quyền của mình. Những hạn chế này cũng có nghĩa là những trách nhiệm nhất định của bạn khi cung cấp các bản sao phần mềm hoặc khi chỉnh sửa phần mềm đó.

Ví dụ, nếu bạn cung cấp các bản sao của một chương trình, dù miễn phí hay không, bạn phải cho người nhận tất cả các quyền mà bạn có. Bạn cũng phải đảm bảo rằng họ cũng nhận được hoặc tiếp cận được mã nguồn. Và bạn phải thông báo những điều khoản này cho họ để họ biết rõ về quyền của mình.

Chúng tôi bảo vệ quyền của bạn với hai bước: (1) bảo vệ bản quyền phần mềm, và (2) cung cấp giấy phép này để bạn có thể sao chép, lưu hành và/hoặc chỉnh sửa phần mềm một cách hợp pháp.

Ngoài ra, để bảo vệ các tác giả cũng như để bảo vệ chính mình, chúng tôi muốn chắc chắn rằng tất cả mọi người đều hiểu rõ rằng không hề có bảo hành đối với phần mềm tự do này. Nếu phần mềm được chỉnh sửa thay đổi bởi một người khác và sau đó lưu hành, thì chúng tôi muốn những người sử dụng biết rằng phiên bản họ đang có không phải là bản gốc, do đó tất cả những trục trặc do những người khác gây ra hoàn toàn không ảnh hưởng tới uy tín của tác giả ban đầu.

Cuối cùng, bất kỳ một chương trình tự do nào cũng đều thường xuyên có nguy cơ bị đe dọa về giấy phép bản quyền. Chúng tôi muốn tránh nguy cơ khi những người cung cấp lại một chương trình tự do có thể có được giấy phép bản quyền cho bản thân họ, từ đó trở thành độc quyền đối với chương trình đó. Để ngăn ngừa trường hợp này, chúng tôi đã nêu rõ rằng mỗi giấy phép bản quyền hoặc phải được cấp cho tất cả mọi người sử dụng một cách tự do hoặc hoàn toàn không cấp phép.

Dưới đây là những điều khoản và điều kiện rõ ràng đối với việc sao chép, lưu hành và chỉnh sửa.

Những điều khoản và điều kiện đối với việc sao chép, lưu hành và chỉnh sửa

0. Giấy phép này áp dụng cho bất kỳ một chương trình hay sản phẩm nào mà người giữ bản quyền công bố rằng nó có thể được cung cấp trong khuôn khổ những điều

khoản của Giấy phép Công cộng này. Từ “Chương trình” dưới đây có nghĩa là tất cả các chương trình hay sản phẩm như vậy, và “sản phẩm dựa trên Chương trình” có nghĩa là Chương trình hoặc bất kỳ một sản phẩm nào bắt nguồn từ chương trình đó tuân theo luật bản quyền, nghĩa là một sản phẩm dựa trên Chương trình hoặc một phần của nó, đúng nguyên bản hoặc có một số chỉnh sửa và/hoặc được dịch ra một ngôn ngữ khác. (Dưới đây, việc dịch cũng được hiểu trong khái niệm “chỉnh sửa”). Mỗi người được cấp phép được gọi là “bạn”.

Trong Giấy phép này không đề cập tới các hoạt động khác ngoài việc sao chép, lưu hành và chỉnh sửa; chúng nằm ngoài phạm vi của giấy phép này. Hành động chạy chương trình không bị hạn chế, và những kết quả từ việc chạy chương trình chỉ được đề cập tới nếu nội dung của nó tạo thành một sản phẩm dựa trên chương trình (độc lập với việc chạy chương trình). Điều này đúng hay không là phụ thuộc vào Chương trình.

1. Bạn có thể sao chép và lưu hành những phiên bản nguyên bản của mã nguồn Chương trình đúng như khi bạn nhận được, qua bất kỳ phương tiện phân phối nào, với điều kiện trên mỗi bản sao bạn đều kèm theo một ghi chú bản quyền rõ ràng và từ chối bảo hành; giữ nguyên tất cả các ghi chú về Giấy phép và về việc không có bất kỳ một sự bảo hành nào; và cùng với Chương trình bạn cung cấp cho người sử dụng một bản sao của Giấy phép này.

Bạn có thể tính phí cho việc chuyển giao bản sao, và tùy theo quyết định của mình bạn có thể cung cấp bảo hành để đổi lại với chi phí mà bạn đã tính.

2. Bạn có thể chỉnh sửa bản sao của bạn hoặc các bản sao của Chương trình hoặc của bất kỳ phần nào của nó, từ đó hình thành một sản phẩm dựa trên Chương trình, và sao chép cũng như lưu hành sản phẩm đó hoặc những chỉnh sửa đó theo điều khoản trong Mục 1 ở trên, với điều kiện bạn đáp ứng được những điều kiện dưới đây:

a) Bạn phải có ghi chú rõ ràng trong những tập tin đã chỉnh sửa là bạn đã chỉnh sửa nó, và ngày tháng của bất kỳ một thay đổi nào.

b) Bạn phải cấp phép miễn phí cho tất cả các bên thứ ba đối với các sản phẩm bạn cung cấp hoặc phát hành, bao gồm Chương trình nguyên bản, từng phần

của nó hay các sản phẩm dựa trên Chương trình hay dựa trên từng phần của Chương trình, theo những điều khoản của Giấy phép này.

c) Nếu chương trình đã chỉnh sửa thường đọc lệnh tương tác trong khi chạy, bạn phải thực hiện sao cho khi bắt đầu chạy để sử dụng tương tác theo cách thông thường nhất phải có một thông báo bao gồm bản quyền và thông báo về việc không có bảo hành (hoặc thông báo bạn là người cung cấp bảo hành), và rằng người sử dụng có thể cung cấp lại Chương trình theo những điều kiện này, và thông báo để người sử dụng có thể xem bản sao của Giấy phép này. (Ngoại lệ: nếu bản thân Chương trình là tương tác nhưng không có một thông báo nào như trên, thì sản phẩm của bạn dựa trên Chương trình đó cũng không bắt buộc phải có thông báo như vậy).

Những yêu cầu trên áp dụng cho toàn bộ các sản phẩm chỉnh sửa. Nếu có những phần của sản phẩm rõ ràng không bắt nguồn từ Chương trình, và có thể được xem là độc lập và riêng biệt, thì Giấy phép này và các điều khoản của nó sẽ không áp dụng cho những phần đó khi bạn cung cấp chúng như những sản phẩm riêng biệt. Nhưng khi bạn cung cấp những phần đó như những phần nhỏ trong cả một sản phẩm dựa trên Chương trình, thì việc cung cấp này phải tuân theo những điều khoản của Giấy phép này, cho phép những người được cấp phép có quyền đối với toàn bộ sản phẩm, cũng như đối với từng phần trong đó, bất kể ai đã viết nó.

Như vậy, điều khoản này không nhằm mục đích xác nhận quyền hoặc tranh giành quyền của bạn đối với những sản phẩm hoàn toàn do bạn viết; mà mục đích của nó là nhằm thi hành quyền kiểm soát đối với việc cung cấp những sản phẩm bắt nguồn hoặc tổng hợp dựa trên Chương trình.

Ngoài ra, việc kết hợp thuần túy Chương trình (hoặc một sản phẩm dựa trên Chương trình) với một sản phẩm không dựa trên Chương trình với mục đích lưu trữ hoặc quảng bá không đưa sản phẩm đó vào trong phạm vi áp dụng của Giấy phép này.

3. Bạn có thể sao chép và cung cấp Chương trình (hoặc một sản phẩm dựa trên Chương trình, nêu trong Mục 2) dưới hình thức mã đã biên dịch hoặc dạng có thể thực thi được trong khuôn khổ các điều khoản nêu trong Mục 1 và 2 ở trên, nếu như bạn:

- a) Kèm theo đó một bản mã nguồn dạng đầy đủ có thể biên dịch được theo các điều khoản trong Mục 1 và 2 nêu trên trong một môi trường trao đổi phần mềm thông thường; hoặc,
- b) Kèm theo đó một đề nghị có hạn trong ít nhất 3 năm, theo đó cung cấp cho bất kỳ một bên thứ ba nào một bản sao đầy đủ của mã nguồn tương ứng, và phải được cung cấp với giá chi phí không cao hơn giá chi phí vật lý của việc cung cấp theo các điều khoản trong Mục 1 và 2 nêu trên trong một môi trường trao đổi phần mềm thông thường; hoặc
- c) Kèm theo đó thông tin bạn đã nhận được để đề nghị cung cấp mã nguồn tương ứng. (Phương án này chỉ được phép đối với việc cung cấp phi thương mại và chỉ với điều kiện nếu bạn nhận được Chương trình dưới hình thức mã đã biên dịch hoặc dạng có thể thực thi được cùng với lời đề nghị như vậy, theo phần b trong điều khoản nêu trên).

Mã nguồn của một sản phẩm là một dạng ưu tiên của sản phẩm dành cho việc chỉnh sửa nó. Với một sản phẩm có thể thi hành, mã nguồn hoàn chỉnh có nghĩa là tất cả các mã nguồn cho các môđun trong sản phẩm đó, cộng với tất cả các tệp tin định nghĩa giao diện đi kèm với nó, cộng với các hướng dẫn dùng để kiểm soát việc biên dịch và cài đặt các tệp thi hành. Tuy nhiên, một ngoại lệ đặc biệt là mã nguồn không cần chứa bất kỳ một thứ gì mà bình thường được cung cấp (từ nguồn khác hoặc hình thức nhị phân) cùng với những thành phần chính (chương trình biên dịch, nhân, và những phần tương tự) của hệ điều hành mà các chương trình chạy trong đó, trừ khi bản thân thành phần đó lại đi kèm với một tệp thi hành.

Nếu việc cung cấp lưu hành mã đã biên dịch hoặc tệp tin thi hành được thực hiện qua việc cho phép tiếp cận và sao chép từ một địa điểm được chỉ định, thì việc cho phép tiếp cận tương đương tới việc sao chép mã nguồn từ cùng địa điểm cũng được tính như việc cung cấp mã nguồn, mặc dù thậm chí các bên thứ ba không bị buộc phải sao chép mã nguồn cùng với mã đã biên dịch.

4. Bạn không được phép sao chép, chỉnh sửa, cấp phép hoặc cung cấp Chương trình trừ phi phải tuân thủ một cách chính xác các điều khoản trong Giấy phép. Bất kỳ ý định sao chép, chỉnh sửa, cấp phép hoặc cung cấp Chương trình theo cách khác đều làm mất hiệu lực và tự động huỷ bỏ quyền của bạn trong khuôn khổ Giấy phép này.

Tuy nhiên, các bên đã nhận được bản sao hoặc quyền từ bạn với Giấy phép này sẽ không bị huỷ bỏ giấy phép nếu các bên đó vẫn tuân thủ đầy đủ các điều khoản của giấy phép.

5. Bạn không bắt buộc phải chấp nhận Giấy phép này khi bạn chưa ký vào đó. Tuy nhiên, không có gì khác đảm bảo cho bạn được phép chỉnh sửa hoặc cung cấp Chương trình hoặc các sản phẩm bắt nguồn từ Chương trình. Những hành động này bị luật pháp nghiêm cấm nếu bạn không chấp nhận Giấy phép này. Do vậy, bằng việc chỉnh sửa hoặc cung cấp Chương trình (hoặc bất kỳ một sản phẩm nào dựa trên Chương trình), bạn đã thể hiện sự chấp thuận đối với Giấy phép này, cùng với tất cả các điều khoản và điều kiện đối với việc sao chép, cung cấp hoặc chỉnh sửa Chương trình hoặc các sản phẩm dựa trên nó.

6. Mỗi khi bạn cung cấp lại Chương trình (hoặc bất kỳ một sản phẩm nào dựa trên Chương trình), người nhận sẽ tự động nhận được giấy phép từ người cấp phép đầu tiên cho phép sao chép, cung cấp và chỉnh sửa Chương trình theo các điều khoản và điều kiện này. Bạn không thể áp đặt bất cứ hạn chế nào khác đối với việc thực hiện quyền của người nhận đã được cấp phép từ thời điểm đó. Bạn cũng không phải chịu trách nhiệm bắt buộc các bên thứ ba tuân thủ theo Giấy phép này.

7. Nếu như, theo quyết định của toà án hoặc với những bằng chứng về việc vi phạm bản quyền hoặc vì bất kỳ lý do nào khác (không giới hạn trong các vấn đề về bản quyền), mà bạn phải tuân theo các điều kiện (nêu ra trong lệnh của toà án, biên bản thoả thuận hoặc ở nơi khác) trái với các điều kiện của Giấy phép này, thì chúng cũng không thể miễn cho bạn khỏi những điều kiện của Giấy phép này. Nếu bạn không thể đồng thời thực hiện các nghĩa vụ của mình trong khuôn khổ Giấy phép này và các nghĩa vụ thích đáng khác, thì hậu quả là bạn hoàn toàn không được cung cấp Chương trình. Ví dụ, nếu trong giấy phép bản quyền không cho phép những người nhận được bản sao trực tiếp hoặc gián tiếp qua bạn có thể cung cấp lại Chương trình thì trong trường hợp này cách duy nhất bạn có thể thoả mãn cả hai điều kiện là hoàn toàn không cung cấp Chương trình.

Nếu bất kỳ một phần nào trong điều khoản này không có hiệu lực hoặc không thể thi hành trong một hoàn cảnh cụ thể, thì sẽ cân đối áp dụng các điều khoản, và toàn bộ điều khoản sẽ được áp dụng trong những hoàn cảnh khác.

Mục đích của điều khoản này không nhằm buộc bạn phải vi phạm bất kỳ một bản quyền nào hoặc các quyền sở hữu khác hoặc tranh luận về giá trị hiệu lực của bất kỳ quyền hạn nào như vậy; mục đích duy nhất của điều khoản này là nhằm bảo vệ sự toàn vẹn của hệ thống cung cấp phần mềm tự do đang được thực hiện với giấy phép công cộng. Nhiều người đã đóng góp rất nhiều vào sự đa dạng của các phần mềm tự do được cung cấp thông qua hệ thống này với sự tin tưởng rằng hệ thống được sử dụng một cách thống nhất; tác giả/người cung cấp có quyền quyết định rằng họ có mong muốn cung cấp phần mềm thông qua hệ thống nào khác hay không, và người được cấp phép không thể có ảnh hưởng tới sự lựa chọn này.

Điều khoản này nhằm làm rõ những hệ quả của các phần còn lại của Giấy phép này.

8. Nếu việc cung cấp và/hoặc sử dụng Chương trình bị cấm ở một số nước nhất định bởi quy định về bản quyền, người giữ bản quyền gốc đã đưa Chương trình vào dưới Giấy phép này có thể bổ sung một điều khoản hạn chế việc cung cấp ở những nước đó, nghĩa là việc cung cấp chỉ được phép ở các nước không bị liệt kê trong danh sách hạn chế. Trong trường hợp này, Giấy phép đưa vào những hạn chế được ghi trong nội dung của nó.

9. Tổ chức Phần mềm Tự do có thể theo thời gian công bố những phiên bản chỉnh sửa và/hoặc phiên bản mới của Giấy phép Công cộng. Những phiên bản đó sẽ đồng nhất với tinh thần của phiên bản hiện này, nhưng có thể khác ở một số chi tiết nhằm giải quyết những vấn đề hay những lo ngại mới.

Mỗi phiên bản sẽ có một mã số phiên bản riêng. Nếu Chương trình và "bất kỳ một phiên bản nào sau đó" có áp dụng một phiên bản Giấy phép cụ thể, bạn có quyền lựa chọn tuân theo những điều khoản và điều kiện của phiên bản giấy phép đó hoặc của bất kỳ một phiên bản nào sau đó do Tổ chức Phần mềm Tự do công bố. Nếu Chương trình không nêu cụ thể mã số phiên bản giấy phép, bạn có thể lựa chọn bất kỳ một phiên bản nào đã từng được công bố bởi Tổ chức Phần mềm Tự do.

10. Nếu bạn muốn kết hợp các phần của Chương trình vào các chương trình tự do khác mà điều kiện cung cấp khác với chương trình này, hãy viết cho tác giả để được phép. Đối với các phần mềm được cấp bản quyền bởi Tổ chức Phần mềm Tự do, hãy đề xuất với tổ chức này; đôi khi chúng tôi cũng có những ngoại lệ. Quyết định của chúng tôi sẽ dựa trên hai mục tiêu là bảo hộ tình trạng tự do của tất cả các sản

phẩm bắt nguồn từ phần mềm tự do của chúng tôi, và thúc đẩy việc chia sẻ và tái sử dụng phần mềm nói chung.

KHÔNG BẢO HÀNH

DO CHƯƠNG TRÌNH ĐƯỢC CẤP PHÉP MIỄN PHÍ NÊN KHÔNG CÓ MỘT CHẾ ĐỘ BẢO HÀNH NÀO TRONG MỨC ĐỘ CHO PHÉP CỦA LUẬT PHÁP. TRỪ KHI ĐƯỢC CÔNG BỐ KHÁC ĐI BẰNG VĂN BẢN, NHỮNG NGƯỜI GIỮ BẢN QUYỀN VÀ/HOẶC CÁC BÊN CUNG CẤP CHƯƠNG TRÌNH NGUYÊN BẢN SẼ KHÔNG BẢO HÀNH DƯỚI BẤT KỲ HÌNH THỨC NÀO, BAO GỒM NHƯNG KHÔNG GIỚI HẠN TRONG CÁC HÌNH THỨC BẢO HÀNH ĐỐI VỚI TÍNH THƯƠNG MẠI CŨNG NHƯ TÍNH THÍCH HỢP CHO MỘT MỤC ĐÍCH CỤ THỂ. BẠN LÀ NGƯỜI CHỊU TOÀN BỘ RỦI RO VỀ CHẤT LƯỢNG CŨNG NHƯ VIỆC VẬN HÀNH CHƯƠNG TRÌNH. TRONG TRƯỜNG HỢP CHƯƠNG TRÌNH CÓ KHIẾM KHUYẾT, BẠN PHẢI CHỊU TOÀN BỘ CHI PHÍ CHO NHỮNG DỊCH VỤ SỬA CHỮA CẦN THIẾT.

TRONG TẤT CẢ CÁC TRƯỜNG HỢP TRỪ KHI CÓ YÊU CẦU CỦA LUẬT PHÁP HOẶC CÓ THỎA THUẬN BẰNG VĂN BẢN, NHỮNG NGƯỜI CÓ BẢN QUYỀN HOẶC BẤT KỲ MỘT BÊN NÀO CHỈNH SỬA VÀ/HOẶC CUNG CẤP LẠI CHƯƠNG TRÌNH TRONG CÁC ĐIỀU KIỆN NHƯ ĐÃ NÊU TRÊN ĐỀU KHÔNG CÓ TRÁCH NHIỆM VỚI BẠN VỀ CÁC LỖI HỎNG HÓC, BAO GỒM CÁC LỖI CHUNG HAY ĐẶC BIỆT, NGẪU NHIÊN HAY TẤT YẾU NẢY SINH DO VIỆC SỬ DỤNG HOẶC KHÔNG SỬ DỤNG ĐƯỢC CHƯƠNG TRÌNH (BAO GỒM NHƯNG KHÔNG GIỚI HẠN TRONG VIỆC MẤT DỮ LIỆU, DỮ LIỆU THIẾU CHÍNH XÁC HOẶC CHƯƠNG TRÌNH KHÔNG VẬN HÀNH ĐƯỢC VỚI CÁC CHƯƠNG TRÌNH KHÁC), THẬM CHÍ CẢ KHI NGƯỜI CÓ BẢN QUYỀN VÀ CÁC BÊN KHÁC ĐÃ ĐƯỢC THÔNG BÁO VỀ KHẢ NĂNG XẢY RA NHỮNG THIỆT HẠI ĐÓ.

KẾT THÚC CÁC ĐIỀU KIỆN VÀ ĐIỀU KHOẢN.

Áp dụng những điều khoản trên như thế nào đối với chương trình của bạn

Nếu bạn xây dựng một chương trình mới, và bạn muốn cung cấp một cách tối đa cho công chúng sử dụng, thì biện pháp tốt nhất để đạt được điều này là phát triển chương

trình đó thành phần mềm tự do để ai cũng có thể cung cấp lại và thay đổi theo những điều khoản như trên.

Để làm được việc này, hãy đính kèm những thông báo như sau cùng với chương trình của mình. An toàn nhất là đính kèm chúng trong phần đầu của tập tin mã nguồn để thông báo một cách hiệu quả nhất về việc không có bảo hành; và mỗi tệp tin đều phải có ít nhất một dòng về “bản quyền” và trỏ đến toàn bộ thông báo.

Một dòng đề tên chương trình và nội dung của nó.

Bản quyền (C) năm, tên tác giả.

Chương trình này là phần mềm tự do, bạn có thể cung cấp lại và/hoặc chỉnh sửa nó theo những điều khoản của Giấy phép Công cộng của GNU do Tổ chức Phần mềm Tự do công bố; phiên bản 2 của Giấy phép, hoặc bất kỳ một phiên bản sau đó (tùy sự lựa chọn của bạn).

Chương trình này được cung cấp với hy vọng nó sẽ hữu ích, tuy nhiên KHÔNG CÓ BẤT KỲ MỘT BẢO HÀNH NÀO; thậm chí kể cả bảo hành về KHẢ NĂNG THƯƠNG MẠI hoặc TÍNH THÍCH HỢP CHO MỘT MỤC ĐÍCH CỤ THỂ. Xin xem Giấy phép Công cộng của GNU để biết thêm chi tiết.

Bạn phải nhận được một bản sao của Giấy phép Công cộng của GNU kèm theo chương trình này; nếu bạn chưa nhận được, xin gửi thư về Tổ chức Phần mềm Tự do, 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

Xin hãy bổ sung thông tin về địa chỉ liên lạc của bạn (thư điện tử và bưu điện).

Nếu chương trình chạy tương tác, hãy đưa một thông báo ngắn khi bắt đầu chạy chương trình như sau:

Gnomovision phiên bản 69, Copyright (C) năm, tên tác giả.

Gnomovision HOÀN TOÀN KHÔNG CÓ BẢO HÀNH; để xem chi tiết hãy gõ `show w'. Đây là một phần mềm miễn phí, bạn có thể cung cấp lại với những điều kiện nhất định, gõ `show c' để xem chi tiết.

Giả thiết lệnh `show w' và `show c' cho xem những phần tương ứng trong Giấy phép Công cộng. Tất nhiên những lệnh mà bạn dùng có thể khác với `show w' và `show c'; những lệnh này có thể là nhấn chuột hoặc lệnh trong thanh công cụ - tùy theo chương trình của bạn.

Bạn cũng cần phải lấy chữ ký của người phụ trách (nếu bạn là người lập trình) hoặc của trường học (nếu có) xác nhận từ chối bản quyền đối với chương trình. Sau đây là ví dụ:

Yoyodyne, Inc., tại đây từ chối tất cả các quyền lợi bản quyền đối với chương trình 'Gnomovision' viết bởi James Hacker.

Chữ ký của Ty Coon, 1 April 1989

Ty Coon, Phó Tổng Giám đốc.

Giấy phép Công cộng này không cho phép đưa chương trình của bạn vào trong các chương trình độc quyền. Nếu chương trình của bạn là một thư viện thủ tục phụ, bạn có thể thấy nó hữu ích hơn nếu cho thư viện liên kết với các ứng dụng độc quyền. Nếu đây là việc bạn muốn làm, hãy sử dụng Giấy phép Công cộng Hạn chế của GNU thay cho Giấy phép này.

GIỚI THIỆU

Giới thiệu tài liệu

Tài liệu Quản trị hệ thống Linux là cuốn giáo trình bổ ích, được xây dựng với mục đích chuyển tải các kiến thức hết sức cơ bản nhưng cần thiết đối với các học viên, đặc biệt là đối với những người làm công tác giảng dạy.

Được biên soạn dựa trên bộ giáo trình của Học viện Linux LPI (Linux Professional Institute). Đây là bộ giáo trình được biên soạn một cách công phu, tỉ mỉ và khoa học, dùng cho việc đào tạo và ôn luyện các chứng chỉ LPI của Học viện Linux.

Do đang trong quá trình xây dựng hệ thống giáo trình và bài giảng một cách khoa học và chuyên nghiệp. Vì vậy, trong quá trình dịch và biên soạn tài liệu không tránh khỏi những thiếu sót. Rất mong được sự đóng góp ý kiến của người đọc để tài liệu ngày càng được hoàn chỉnh hơn.

Hi vọng trong thời gian tới, cùng với sự cộng tác chặt chẽ giữa RedHat và Công ty ISE, chúng tôi sẽ xây dựng được bộ giáo trình hoàn chỉnh, khoa học và phong phú hơn.

Nhóm tác giả xin chân thành cảm ơn và chúc cho người đọc có được một khoá học bổ ích.

Nhóm tác giả Công ty ISE

Giới thiệu chương trình đào tạo ISE Linux

Chương trình đào tạo ISE Linux bao gồm 3 khoá học:

- Linux Cơ bản (Basic Course)
- Linux Trung cấp (Intermediate Course)
- Linux Nâng cao (Advanced Course)

Với 03 khoá ISE Linux này, lượng kiến thức đem lại cho học viên là đủ để có thể tham gia vào các kỳ thi đạt chứng chỉ quốc tế như Chứng chỉ LPI, Chứng chỉ RedHat, ...

MỤC LỤC

GIỚI THIỆU	12
Giới thiệu tài liệu	12
Giới thiệu chương trình đào tạo ISE Linux	13
CÀI ĐẶT	18
Cấu trúc của đĩa cài.....	18
Cài đặt Cục bộ.....	19
Cài đặt qua Mạng.....	20
Phục hồi Hệ thống.....	20
Chiến lược Phân vùng.....	21
Khởi động kép với nhiều hệ điều hành	22
Thực hành	22
CÁU HÌNH PHẦN CỨNG	24
Bộ nhớ.....	24
Quản lý Tài nguyên.....	24
USB.....	26
SCSI.....	27
Network Card.....	27
Modem	28
Máy in	29
Thực hành	29
QUẢN LÝ THIẾT BỊ.....	31
Đĩa và Phân vùng.....	31
Công cụ Phân vùng đĩa	33

Bootloader.....	34
Những thiết bị đã quản lý.....	35
Quotas	37
Thực hành	39
HỆ THỐNG FILE TRONG LINUX.....	40
Cấu trúc của hệ thống file	40
Sự nhất quán trong định dạng và kiến trúc của hệ thống file	43
Kiểm tra dung lượng đĩa	45
Quyền truy xuất File, Thư mục.....	46
Thực hành	49
DÒNG LỆNH.....	50
Tương tác với SHELL	50
Biến môi trường của Shell	51
Xuất, nhập, đổi hướng.....	53
Dấu ngoặc và Các ký tự Đa nghĩa (Metacharacter).....	57
Lịch sử dòng lệnh.....	59
Một số lệnh khác	60
Thực hành	61
QUẢN LÝ FILE	65
Di chuyển quanh hệ thống file.....	65
Tìm kiếm file và thư mục.....	65
Làm việc với thư mục	68
Sử dụng cp và mv	69
Hard links và symbol links	70
Touching và dd-ing.....	72

Thực hành	73
QUẢN LÝ TIẾN TRÌNH.....	75
Xem các tiến trình đang chạy	75
Thay đổi tiến trình.....	77
Tiến trình và Shell.....	80
Thực hành	83
XỬ LÝ VĂN BẢN	85
cat the Swiss Army Knife	85
Các công cụ đơn giản.....	86
Xử lý văn bản.....	88
Thực hành	93
CÀI ĐẶT PHẦN MỀM.....	95
Giới thiệu	95
Thư viện tĩnh và thư viện chia sẻ	96
Cài đặt nguồn	100
Quản lý gói Redhat (Redhat Package Manager RPM).....	101
Công cụ Alien.....	106
Thực hành	107
THAO TÁC VỚI VĂN BẢN NÂNG CAO	109
Các biểu thức chính qui	109
Họ grep.....	110
Làm việc với grep	110
egrep và fgrep	111
Bộ soạn thảo Stream – sed.....	112

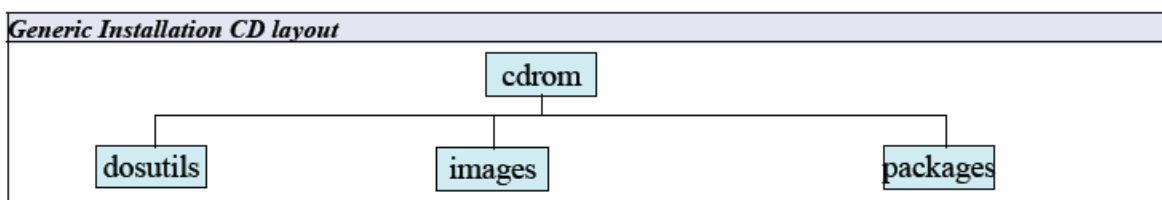
Thực hành	115
SỬ DỤNG TRÌNH SOẠN THẢO VI	117
Các chế độ Vi.....	117
Các mục văn bản.....	118
Chèn văn bản.....	118
Xoá văn bản	119
Copy / Paste	120
Tìm kiếm.....	121
Làm lại (Undo).....	121
Ghi văn bản.....	122
Thực hành	123

CÀI ĐẶT

CÀI ĐẶT

Cấu trúc của đĩa cài

Hiện tại, có rất nhiều phiên bản phân phối Linux khác nhau. Với mỗi bản, cách đặt tên của các thư mục trên đĩa cài cũng khác nhau. Thông thường chúng có dạng như sau:



packages: Thư mục chứa các gói phần mềm đã được biên dịch trước. Tùy vào từng bản phân phối mà thư mục này có tên khác nhau. Dưới đây là một số ví dụ:

debian: **dist**

mandrake: **Mandrake**

redhat: **RedHat**

suse: **suse**

fedora: **Fedora**

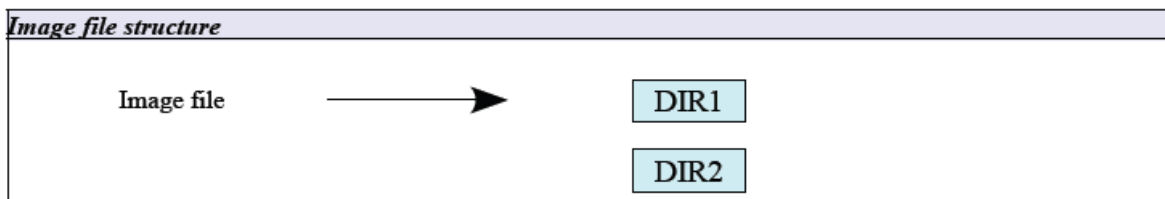
images: Dùng để chứa ảnh của Linux. Có nhiều kiểu file ảnh khác nhau. Mỗi file có một công dụng riêng:

- Khởi động tiến trình cài đặt
- Cung cấp module cho nhân
- Khôi phục lại hệ thống

CÀI ĐẶT

Một số ảnh có thể được ghi lại vào đĩa mềm hoặc CD, USB nhằm mục đích khởi động quá trình cài đặt từ nhiều nguồn khác nhau.

Bản thân nhiều file ảnh cũng chứa bên trong nó những file và thư mục con. Có thể truy cập đến những file và thư mục này thông qua việc ánh xạ file ảnh vào một thiết bị loop.



```
mount -o loop /path/to/Image /mnt
```

dosutils: Thư mục chứa một số công cụ giúp cho việc chuẩn bị cài đặt được thuận lợi hơn trong môi trường DOS.

Cài đặt Cục bộ

Cài đặt cục bộ là cách thức dễ dàng và phổ thông nhất trong tất cả các phương thức cài đặt. Hầu hết các bản phân phối Linux đều có dạng boot CD cho phép khởi động quá trình cài đặt một cách tự động. Với những máy tính không có ổ CD, có thể thay thế nó bởi đĩa mềm hoặc USB để khởi động quá trình này (khi đó, thư mục **packages** thường được đặt trong ổ cứng).

Để tạo ra đĩa mềm hoặc đĩa USB có khả năng khởi động, có thể dùng lệnh **dd** trong Linux hoặc **rawrite.exe** trong DOS/Win.

```
dd if=/path/to/Image of=/dev/fd0 (hoặc /dev/sdX)  
rawrite.exe
```

CÀI ĐẶT

Ví dụ: Đối với các bản phân phối của RedHat các file ảnh này có tên là **boot.img**. Ngoài ra, có thể còn có một số file ảnh đặc biệt khác được cung cấp như: **bootnet.img** hay **pcmcia.img**.

Cài đặt qua Mạng

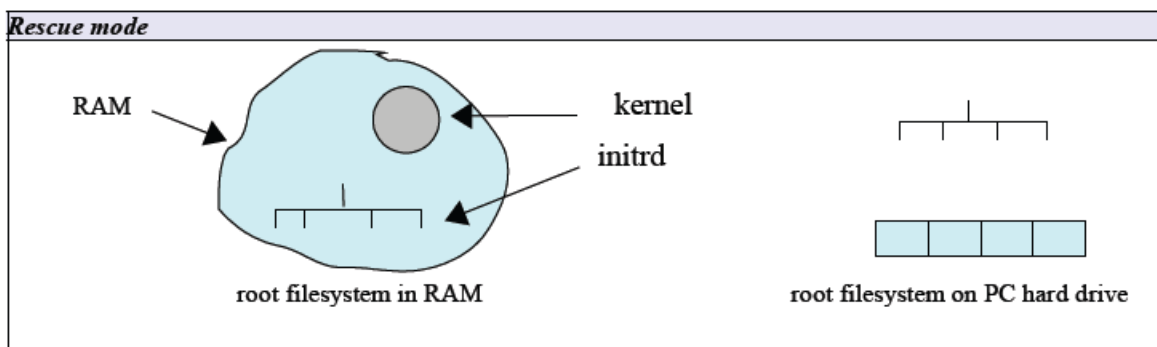
Thông thường các gói cài đặt được để tại một server ở xa, người dùng chỉ cần khởi động quá trình cài đặt, thiết lập các tham số mạng chính xác sau đó, tiến trình cài đặt sẽ tự động download các gói cần thiết về máy tính để cài (thông qua các giao thức như FTP, HTTP, NFS).

Để khởi động quá trình cài đặt có thể sử dụng bất kỳ phương thức nào như đã miêu tả trong phần Cài đặt Cục bộ. Ngoài ra, quá trình này cũng có thể được khởi động thông qua một Card Mạng có khả năng boot kết hợp với DHCP và TFTP Server được thiết lập cho mục đích này.

Phục hồi Hệ thống

Trong trường hợp hệ thống bị trục trặc, không thể khởi động chính xác, có thể phục hồi được một số lỗi thông qua cơ chế khởi động Phục hồi Hệ thống.

Khi khởi động cơ chế này, một phiên bản thu gọn của Linux và một hệ thống file ảo được nạp vào và chạy ngay trên RAM hệ thống. Hệ thống file thật sẽ được tìm kiếm và ánh xạ vào một thư mục của hệ thống file ảo này. Người dùng có thể dùng lệnh **chroot** để chuyển qua hệ thống file thật và xử lý sự cố. Thông thường nếu tìm thấy, nó sẽ được ánh xạ vào thư mục **/mnt/sysimage** của hệ thống ảo.



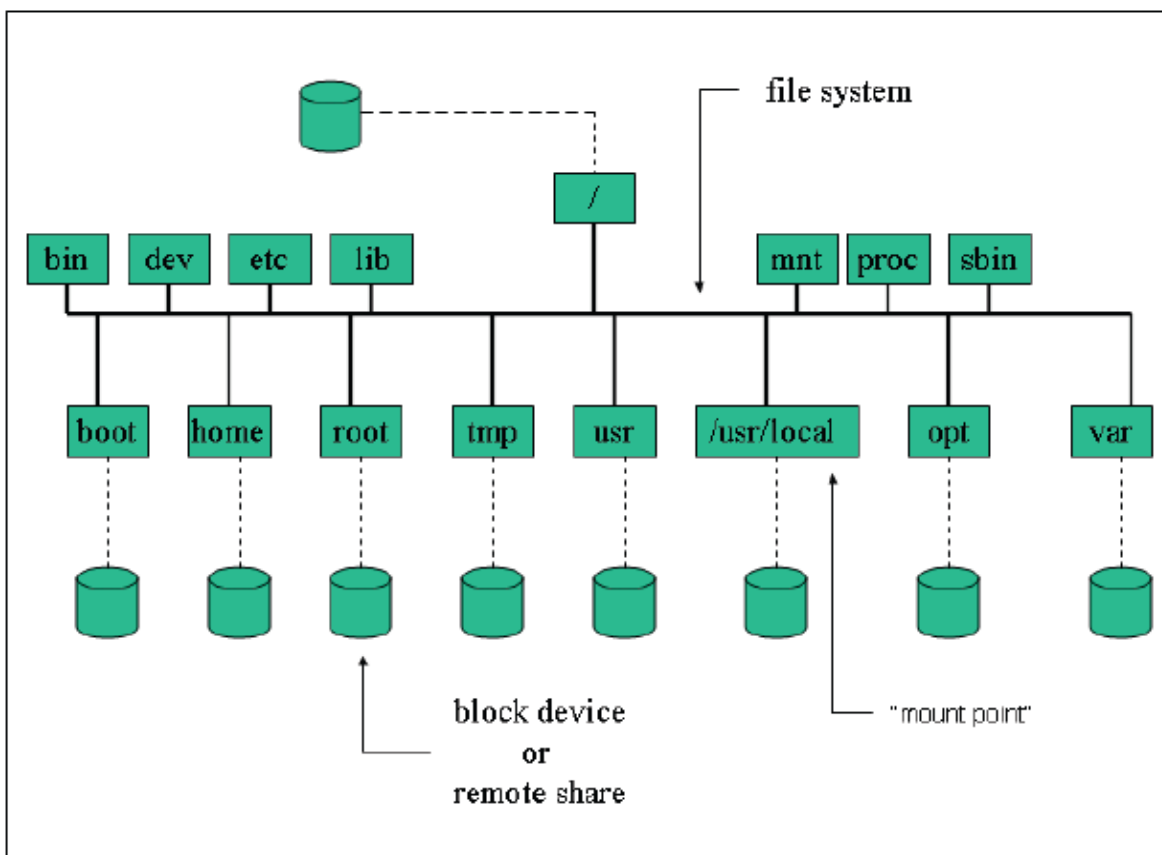
CÀI ĐẶT



```
chroot /mnt/sysimage
```

Chiến lược Phân vùng

Phân vùng ổ cứng là quá trình không thể thiếu trong khi cài đặt Linux. Tùy theo từng phiên bản phân phối mà quá trình này có thể thực hiện tự động hoặc thủ công. Để thực hiện thủ công, cần có sự hiểu biết sâu sắc về hệ thống file trong Linux cũng như mục đích sử dụng hệ thống.



Hình trên mô tả một lược đồ phân vùng dạng đơn giản cùng hệ thống file của một hệ thống mẫu. Thực chất hệ thống file trong Linux là một cây bao gồm thư mục gốc "/"

CÀI ĐẶT

và các thư mục con nhiều cấp. Các tài nguyên hệ thống được sử dụng để lưu trữ dữ liệu được **gắn kết/ánh xạ (mounted)** vào các điểm chỉ định trên hệ thống file, các điểm này được gọi là các **điểm gắn kết/ánh xạ (mount point)**. Thư mục gốc “/” cũng là một điểm gắn kết và phân vùng lưu trữ dữ liệu cho “/” sẽ được xác định trong quá trình khởi động.

Khởi động kép với nhiều hệ điều hành

Cũng giống như Windows, Linux hỗ trợ nhiều hệ điều hành trên một máy tính. Chương trình khởi động của Linux sẽ tự động nhận biết các hệ điều hành này. Nếu nhận biết thành công, một lựa chọn sẽ được tự động thêm vào menu khởi động.

Linux cũng hỗ trợ khởi động hệ điều hành Windows. Để thiết lập được hệ thống như vậy, cần phải có chiến lược phân vùng đúng đắn. Sau đó, cách đơn giản nhất là cài đặt Linux sau khi cài đặt Windows. Nếu không, phải có kinh nghiệm về ổ cứng, bảng phân vùng... và những phần liên quan để có thể khắc phục sự cố.

Thực hành

1. Cài đặt (qua mạng hoặc không) một hệ thống Linux với yêu cầu như sau:

+ Chọn “Custom System”

+ Phân vùng ổ cứng với Disk Druid thành các phân vùng như sau:

/boot 100M

SWAP 512M

/ 2000M

/usr 6000M

/home 1000M

/tmp 2000M

/var Phần dung lượng còn lại của ổ cứng

+ Cài đặt GRUB lên MBR và đặt mật khẩu cho GRUB.

CÀI ĐẶT

+ Cài đặt các gói theo yêu cầu của giảng viên.

2. Phục hồi hệ thống

+ Giả sử bạn bị quên mật khẩu root, khởi động lại máy tính và phục hồi lại nó bằng chế độ single.

+ Giả sử bạn cũng quên cả mật khẩu của GRUB nên không khởi động vào chế độ single được. Khởi động máy tính bằng đĩa có khả năng cứu hộ (Rescue Mode). Sửa lại file cấu hình của GRUB (/boot/grub/grub.conf) để xóa mật khẩu.

CẤU HÌNH PHẦN CỨNG

Bộ nhớ

RAM hệ thống được dò tìm bởi BIOS trong quá trình khởi động và kernel sử dụng kết quả của quá trình này. Vì vậy, trong những trường hợp hệ thống sử dụng RAM số lượng ít hơn thực tế cài đặt, cần phải kiểm tra lại phần cứng xem đã cắm đúng qui cách chưa hoặc BIOS có hoạt động đúng không.

Trong trường hợp muốn chỉ định chính xác lượng RAM mà Linux phải dùng, có thể thêm các tham số cho chương trình khởi động được cài đặt trên hệ thống:

LILO

Sửa file /etc/lilo.conf, thêm dòng

```
append="mem=<Dung lượng RAM>M"
```

Sau đó chạy /sbin/lilo.

GRUB

Sửa file /boot/grub/grub.conf như sau:

```
kernel vmlinuz mem=<Dung lượng RAM>M
```

Quản lý Tài nguyên

Để truy cập vào các thiết bị, hệ thống (CPU) phải cấp phát các tài nguyên truy cập cho chúng. Sau đây là các kiểu tài nguyên này:

IRQs (Interrupt Request Lines)

Là các đường truyền liên lạc trực tiếp từ thiết bị đến CPU giúp các thiết bị yêu cầu CPU xử lý thông tin của chúng gửi đến. Mỗi khi có yêu cầu/ngắt, CPU phải tạm dừng công việc đang thi hành để xử lý ngắt. Có 16 IRQs đánh số từ 0 đến 15.

CẤU HÌNH PHẦN CỨNG

Địa chỉ I/O (Input/Output – Nhập/Xuất)

Là những địa chỉ trên bộ nhớ hệ thống được ánh xạ vào bộ nhớ của thiết bị. Mọi thao tác trên vùng nhớ này tương đương với thao tác lên bộ nhớ của thiết bị.

DMA (Direct Memory Access channels)

Là các kênh truyền dữ liệu cho phép thiết bị thao tác trực tiếp lên bộ nhớ hệ thống mà không phải thông qua CPU.

Liệt kê các tài nguyên đã cấp phát

Nhân lưu giữ các thông tin này trong thư mục /proc. Các file được sử dụng là:

/proc/dma

/proc/interrupts

/proc/ioports

/proc/pci

Những thông tin này cũng có thể được liệt kê ra bởi các công cụ như *lspci* hay *dmesg*:

lspci

Liệt kê danh sách thông tin của những chipset gắn trên thiết bị tại các khe PCI. Với tham số **-v**, lệnh sẽ liệt kê các thiết lập về I/O và IRQ.

Với tham số **-b** (BUS centric) lệnh sẽ liệt kê thông tin do BIOS quản lý (có thể khác với do nhân quản lý).

dmesg

Hiển thị tất cả các thông điệp mức nhân tính từ lúc khởi động máy. Những thông tin này cũng có thể lấy được từ file **/var/log/dmesg**.

CẤU HÌNH PHẦN CỨNG

Một số Tài nguyên thường dùng

Device	I/O port	IRQ
<i>/dev/ttyS0</i>	<i>0x03f8</i>	<i>4</i>
<i>/dev/ttyS1</i>	<i>0x02f8</i>	<i>3</i>
<i>/dev/lp0</i>	<i>0x378</i>	<i>7</i>
<i>/dev/lp1</i>	<i>0x278</i>	<i>5</i>
<i>soundcard</i>	<i>0x220</i>	

USB

USB (Universal Serial Bus) là chuẩn kết nối giữa các thiết bị với nhau và với PC. Chúng được chia thành các lớp thiết bị như sau:

Display Devices

Communication Devices

Audio Devices

Mass Storage Devices

Human Interface Devices (HID)

Mỗi thiết bị gắn vào cổng USB đều được điều khiển bởi một bộ điều khiển USB Controller. Bắt đầu từ phiên bản nhân 2.2.7, Linux mới hỗ trợ USB Controller. Có 3 kiểu USB Controllers như sau:

Host Controller	Kernel Module
<i>OHCI (Compaq)</i>	<i>usb-ohci.o</i>
<i>UHCI (Intel)</i>	<i>usb-uhci.o</i>
<i>EHCI (USB v 2.0)</i>	<i>ehci-hdc.o</i>

CẤU HÌNH PHẦN CỨNG

SCSI

Hiện nay, chuẩn SCSI có hai kiểu giao tiếp là:

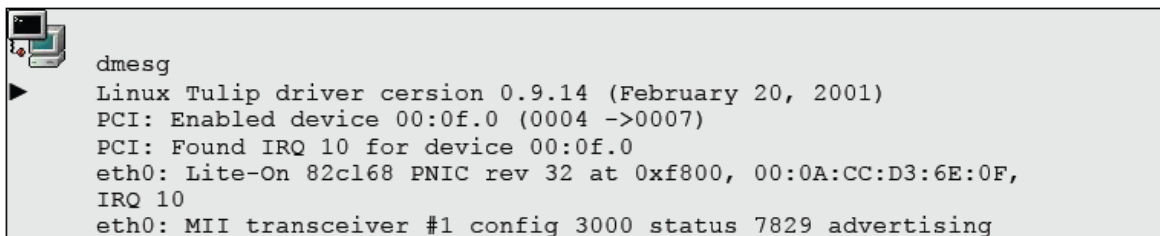
- Chuẩn giao tiếp 8-bit với một kênh truyền hỗ trợ 8 thiết bị SCSI. Tuy nhiên do đã bao gồm cả controller nên card SCSI theo chuẩn này chỉ có thể kết nối được với tối đa 7 thiết bị SCSI khác.
- Chuẩn giao tiếp 16-bit (WIDE) cũng tương tự như chuẩn 8-bit với số thiết bị kết nối tối đa là 15.

Mỗi thiết bị SCSI được gán một số hiệu SCSI ID duy nhất. Các số hiệu này chạy từ 0 đến 7 hoặc 15 và có thể được thiết lập bởi các jump trên chúng.

Hệ thống sẽ tự động khởi động từ thiết bị có SCSI ID = 0. Tuy nhiên, thứ tự này có thể thay đổi được trong menu SCSI khi hệ thống khởi động

Network Card

Để dùng được card mạng, nhân của hệ thống phải hỗ trợ chúng. Thông tin về card mạng đang dùng trong hệ thống có thể được tìm thấy thông qua các lệnh hoặc file sau: **dmesg, lspci, scanpci, /proc/interrupts, /sbin/lsmmod** hay **/etc/modules.conf**



```
dmesg
Linux Tulip driver cersion 0.9.14 (February 20, 2001)
PCI: Enabled device 00:0f.0 (0004 ->0007)
PCI: Found IRQ 10 for device 00:0f.0
eth0: Lite-On 82c168 PNIC rev 32 at 0xf800, 00:0A:CC:D3:6E:0F,
IRQ 10
eth0: MII transceiver #1 config 3000 status 7829 advertising
```

CẤU HÌNH PHẦN CỨNG

```
cat /proc/interrupts
0: 8729602 XT-PIC timer
1: 4 XT-PIC keyboard
2: 0 XT-PIC cascade
7: 0 XT-PIC parport0
8: 1 XT-PIC rtc
10: 622417 XT-PIC eth0
11: 0 XT-PIC usb-uhci
14: 143040 XT-PIC ide0
15: 180 XT-PIC ide1

/sbin/lsmmod
Module Size Used by
tulip 37360 1 (autoclean)
```

Ví dụ trên cho thấy một card mạng đang dùng có chipset là Tulip, địa chỉ I/O là 0xf800 và IRQ = 10. Nếu các thông tin này hoạt động tốt thì có thể sử dụng nó tại các hệ thống khác tương đương về phần cứng nhưng có lỗi khi nạp module hay I/O hoặc IRQ bị xung đột. Để thay đổi các tham số này có thể sử dụng các lệnh **modprobe** hoặc **insmod** (thay đổi trực tiếp) hay ghi vào trong file **/etc/modules.conf** (có hiệu lực từ lần khởi động sau).

Modem

Do các Modem cắm trong yêu cầu CPU xử lý dữ liệu cho chúng nên thông thường Linux không hỗ trợ các thiết bị loại này (mặc dù có nhiều cách đi đường vòng để giải quyết vấn đề trên).

Vì vậy, tài liệu này chỉ đề cập đến các modem cắm ngoài (sử dụng cổng serial). Trong Linux, các cổng serial được định nghĩa khác so với trong DOS/Windows:

DOS	Linux
<i>COM1</i>	<i>/dev/ttyS0</i>
<i>COM2</i>	<i>/dev/ttyS1</i>
<i>COM3</i>	<i>/dev/ttyS2</i>

CẤU HÌNH PHẦN CỨNG

Mặc dù hầu hết các phiên bản phân phối Linux đều có công cụ đồ họa để dò tìm và thiết lập tham số cho modem, nhưng trong các server chỉ sử dụng giao diện text, **setserial** là công cụ hữu ích cho việc này. Với tham số **-g**, **setserial** có thể tìm được cổng serial nào đang có thiết bị kết nối. Ngoài ra, **setserial** cũng có khả năng thiết lập lại tham số hoạt động cho những cổng này.

```
setserial -g /dev/ttyS*

/dev/ttyS0, UART: 16550A, Port: 0x03f8, IRQ: 4
/dev/ttyS1, UART: 16550A, Port: 0x02f8, IRQ: 3
```

```
ln -s /dev/ttyS1 /dev/modem
```

Để thiết lập kết nối và quay số trong môi trường text, có thể dùng bộ công cụ **wvdial**, **wvdialconf**, **wvdial.conf**.

Máy in

Hướng dẫn chi tiết hơn về máy in sẽ được đề cập đến trong những phần sau của tài liệu. Thông thường những máy in có khả năng PnP sẽ được dò tìm ngay khi hệ thống khởi động (kể cả máy in USB cũng có thể được dò thấy) và có thể nhìn thấy bởi lệnh **dmesg**.

Quá trình In trong Linux được thực hiện trong hai bước. Đầu tiên, dữ liệu in được lọc qua một bộ lọc theo định dạng của trình quản lý máy in. Sau đó, dữ liệu mới được xử lý để đưa ra máy in.

Thực hành

1. Sử dụng lệnh **dmesg** để xem thông tin từ file **/var/log/dmesg**. Tìm trong đó các thông tin về USB, tty hoặc eth0 và trả lời:

CẤU HÌNH PHẦN CỨNG

- Tên của USB controllers được sử dụng?
- Số hiệu IRQ của hai cổng serial đầu tiên là bao nhiêu?

2. Kiểm tra nội dung của các file:

/proc/ioports

/proc/interrupts

/proc/pci

/proc/dma

3. PCI bus:

- Kiểm tra output của các lệnh `lspci -v` and `scanpci -v`. Kiểu của card mạng trên máy bạn là gì?
- Kiểm tra xem có bao nhiêu mục 'bus' trong file /proc/pci. Những thông tin này có giống như kết quả của 2 lệnh trên không?

4. USB:

- Dùng lệnh `lsmod` và `lsusb` để kiểm tra xem kiểu host controller nào đang được sử dụng trong hệ thống? UHCI, OHCI hay EHCI.

QUẢN LÝ THIẾT BỊ

QUẢN LÝ THIẾT BỊ

Đĩa và Phân vùng

Đĩa vật lý

Đĩa vật lý được nhân Linux gán vào các mục trong thư mục /dev. Mọi kết nối từ nhân đến các thiết bị đều thông qua bộ số major/minor. Các số major được định nghĩa trong file /proc/devices. Ví dụ: Đĩa cứng IDE đầu tiên có số major = 3

Block devices:

1	<i>ramdisk</i>
2	<i>fd</i>
3	<i>ide0</i>

Để nhận dạng các ổ cứng trong /dev, Linux dùng hai ký tự bắt đầu là hd cho các thiết bị IDE, sd cho các thiết bị SCSI hoặc ổ đĩa USB (nhưng lại dùng st cho ổ băng SCSI). Sau đó là các ký tự thêm vào để định danh các thiết bị cùng họ:

<i>hda</i>	Primary Master
<i>hdb</i>	Primary Slave
<i>hdc</i>	Secondary Master
<i>hdd</i>	Secondary Slave
<i>sda</i>	First SCSI/USB disk
<i>sdb</i>	Second SCSI/USB disk

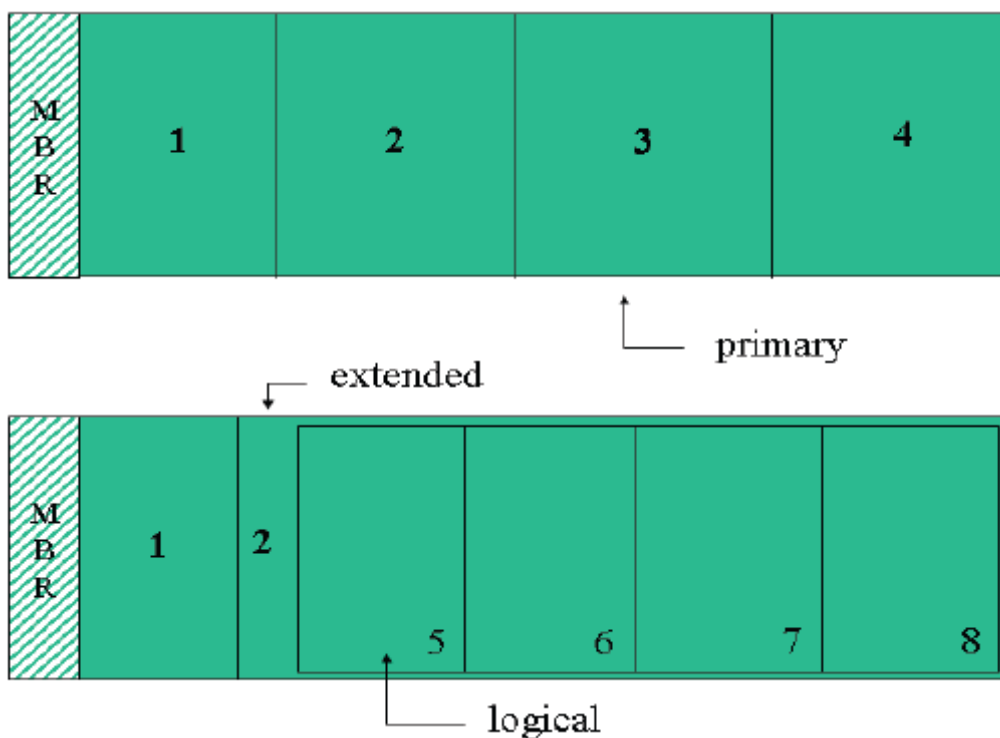
Phân vùng đĩa

Để có thể sử dụng được, các đĩa cứng cần phải được phân vùng. Linux thêm vào đằng sau định danh đĩa cứng số hiệu của các phân vùng để quản lý.

QUẢN LÝ THIẾT BỊ

<i>hda1</i>	Partition đầu tiên trên ổ IDE đầu tiên
<i>hda2</i>	Partition thứ hai trên ổ IDE đầu tiên
<i>sdc3</i>	Partition thứ ba trên ổ SCSI thứ ba

Mỗi ổ IDE chỉ cho phép có 4 phân vùng chính và một trong số chúng có thể được đánh dấu là phân vùng mở rộng. Phân vùng này có thể được đánh chia thành nhiều phân vùng con bên trong. Linux hỗ trợ tối đa 64 phân vùng trên ổ IDE và 16 phân vùng trên ổ SCSI.



QUẢN LÝ THIẾT BỊ

Typical output of fdisk -l

Device	Boot	Start	End	Blocks	Id	System
/dev/hda1	*	1	748	6297448+	b	Win95 FAT32
/dev/hda2		785	788	32130	83	Linux
/dev/hda3		789	2432	13205430	5	Extended
/dev/hda5		789	1235	3590496	83	Linux
/dev/hda6		1236	1618	3076416	83	Linux
/dev/hda7		1619	1720	819283+	83	Linux
/dev/hda8		1721	1784	514048+	83	Linux
/dev/hda9		1785	1835	409626	83	Linux
/dev/hda10		1836	1874	313236	83	Linux
/dev/hda11		1875	1883	72261	82	Linux swap

Trong ví dụ trên (dùng **fdisk -l**), hệ thống có ba phân vùng chính được định danh từ hda1 đến hda3. Phân vùng thứ 3 được đánh dấu là mở rộng và chứa trong nó 7 phân vùng con. Do đó hda3 không được dùng. Các phân vùng con được định danh từ hda5 trở đi.

Công cụ Phân vùng đĩa

Trước khi cài đặt Linux

PartitionMagic

fips

fdisk

...

Trong khi cài đặt Linux

Trong quá trình cài đặt Linux, có thể sử dụng chính công cụ Tự động phân vùng của một số bản phân phối hoặc dùng công cụ phân vùng thủ công đi kèm:

diskdrake **Mandrake**

DiskDruid **RedHat**

QUẢN LÝ THIẾT BỊ

Trên hệ thống đang hoạt động

fdisk luôn là công cụ được lựa chọn để phân vùng các đĩa cứng. Tập lệnh của fdisk tương đối đơn giản, chỉ cần gõ lệnh m để xem đầy đủ các lệnh của nó.

Sau khi fdisk, nếu có thay đổi bảng phân vùng, cần phải khởi động lại máy tính. Để sử dụng được các phân vùng này, phải định dạng chúng với các định dạng hệ thống file mà Linux hiểu được thông qua các lệnh: **mkfs** hoặc **mke2fs**.

Bootloader

Bootloader là chương trình mặc định được cài đặt trên MBR nhằm giúp máy tính lựa chọn được phân vùng khởi động, nạp bộ môi hệ điều hành và chuyển quyền kiểm soát cho hệ điều hành.

Các bản Linux được phân phối với hai Bootloader riêng. Tuy nhiên, chúng cũng nhận vai cho mỗi hệ điều hành nên có thể cài đặt vào BR của phân vùng khởi động chứ không nhất thiết phải cài đặt trên MBR.

LILO (the **L**inux **bootLO**ader)

Được thiết kế với 3 thành phần chính

LILO

Mã nhị phân của trình bootloader, được cài đặt trên MBR hoặc BR. Nó sẽ nạp mã khởi động giai đoạn 2 tại /boot/boot.b.

/etc/lilo.conf

File cấu hình của LILO với một số tham số như sau:

QUẢN LÝ THIẾT BỊ

<i>boot*</i>	Nơi LILO được cài đặt (/dev/hda là MBR)
<i>install</i>	Nơi mã khởi động giai đoạn 2 được cài đặt (mặc định là boot.b)
<i>prompt</i>	Cho người dùng lựa chọn hệ điều hành khi khởi động máy.
<i>default</i>	Tên của file ảnh được nạp khi khởi động mặc định
<i>timeout</i>	Thời gian kết thúc lựa chọn
<i>image*</i>	Đường dẫn chỉ đến nhân để khởi động
<i>label*</i>	Tên của file ảnh
<i>root*</i>	Tên của đĩa chứa thư mục gốc của hệ thống file.
...	

/sbin/lilo

Công cụ dùng để đọc tham số từ /etc/lilo.conf và thiết lập cho LILO.

GRUB (the Grand Unified Bootloader)

Được phát triển sau LILO với một vài ưu điểm so với LILO. Thông tin chi tiết về GRUB có thể được xem qua lệnh **info**.

<i>Example</i>	<i>grub.conf</i>
	<pre>default=0 timeout=10 splashimage=(hd0,0)/grub/splash.xpm.gz title Linux (2.4.18-14) root (hd0,0) kernel /vmlinuz-2.4.18-14 ro root=/dev/hda5 initrd /initrd-2.4.18-14.img</pre>

Những thiết bị đã quản lý

File /etc/fstab lưu thông tin về các điểm kết nối xác định trước cho các thiết bị khởi.

QUẢN LÝ THIẾT BỊ

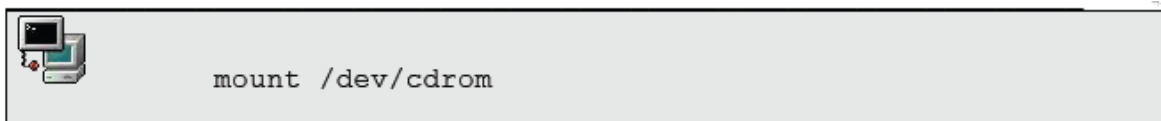
The /etc/fstab format

device	mount-point	fstype	options	dump-number	fsck-number
---------------	--------------------	---------------	----------------	--------------------	--------------------

Sample /etc/fstab

LABEL=/	/	ext2	defaults	1	1
LABEL=/boot	/boot	ext2	defaults	1	2
LABEL=/home	/home	ext3	defaults	1	2
/dev/fd0	/mnt/floppy	auto	noauto,owner	0	0
LABEL=/usr	/usr	ext2	defaults	1	2
LABEL=/var	/var	ext3	defaults	1	2
none	/proc	proc	defaults	0	0
none	/dev/shm	tmpfs	defaults	0	0
none	/dev/pts	devpts	gid=5,mode=620	0	0
/dev/hdc9	swap,pri=-1	swap	defaults	0	0
/dev/cdrom	/mnt/cdrom	iso9660	noauto,owner,kudzu,ro	0	0

Ngoài ra, /etc/fstab cũng được dùng để trợ giúp cho các kết gán tài nguyên thời gian thực. Ví dụ:



Chương trình **mount** sẽ đọc /etc/fstab và quyết định tài nguyên (hoặc điểm kết nối) nào sẽ được sử dụng và các tham số của việc kết nối cũng có thể được xác định tại bước này. Sau đây là một số tham số tùy chọn (option) của mount:


<i>rw, ro</i>	đọc-ghi hoặc chỉ đọc
<i>users</i>	có thể đọc và umount bởi mọi người dùng
<i>user</i>	chỉ có thể umount bởi người dùng đã mount nó
<i>owner</i>	có thể thay đổi quyền và thuộc về người dùng đã

QUẢN LÝ THIẾT BỊ

	mount nó
<i>usrquota</i>	bật thiết lập hạn ngạch đĩa mức người dùng
<i>grpquota</i>	bật thiết lập hạn ngạch đĩa mức nhóm người dùng

Nếu sử dụng với tham số **-a**, mount sẽ tự động ánh xạ tất cả các khai báo trong */etc/fstab* mà chưa được mount và không có tùy chọn **noauto**.

Một số thiết bị được truy cập thông qua các nhãn. Nhãn được gán cho thiết bị bởi lệnh **tune2fs**:




```
tune2fs -L /usr/local /dev/hdb12
```

Quotas

Quota là công cụ cho phép quản trị hệ thống thiết lập hạn ngạch lưu trữ trên đĩa. Công cụ này không yêu cầu khởi động lại hệ thống. Sau đây là một số bước làm chung:

1. Thêm tùy chọn *usrquota* vào file */etc/fstab* tại dòng chứa phân vùng cần phân hạn ngạch.
2. Remount lại phân vùng này:



```
mount -o remount <device>
```

3. Thiết lập tình trạng quota:



```
quotacheck -ca
```

QUẢN LÝ THIẾT BỊ

Sau lệnh này, nếu thiết lập đúng, file aquota.user sẽ được sinh ra tại thư mục gốc của phân vùng.

4. Sửa lại hạn ngạch cho từng người dùng:



```
edquota -u <user>
```

Tham số soft/hard limit phải được thiết lập cho cả số blocks lẫn inodes cho mỗi user. Hệ thống sẽ cho phép sử dụng vượt quá con số soft limit cho đến khi hết hạn. Khi đó, hard limit sẽ được sử dụng để quyết định chính xác hạn ngạch của người dùng. Sử dụng tham số -T để quyết định thời gian này.

5. Bật chế độ hạn ngạch lên:



```
quotaon -a
```

Người dùng có thể kiểm tra hạn ngạch của mình bằng lệnh **quota**, quản trị có thể sinh ra báo cáo về hạn ngạch bằng lệnh **repquota** hoặc **quotastats**.

Thực hành

1. Sử dụng fdisk, xóa phân vùng /home, sau đó tạo lại 1 phân vùng mới. Khởi động lại máy tính. Vấn đề gì sẽ xảy ra? Giải quyết như thế nào?
2. Dùng lệnh mkfs tạo ra định dạng hệ thống file kiểu ext3 trên phân vùng này
3. Tạo thư mục data trong thư mục gốc. Thiết lập lại /etc/fstab sao cho thư mục này là mount point của phân vùng mới định dạng.
4. Dùng lệnh mount có tham số để kiểm tra lại xem đã thiết lập /etc/fstab đúng chưa.
5. Thiết lập hạn ngạch đĩa cho phân vùng trên theo từng bước đã hướng dẫn.

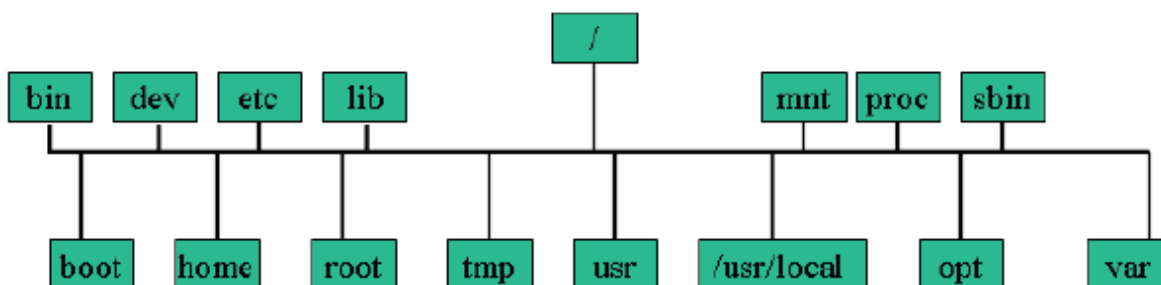
HỆ THỐNG FILE TRONG LINUX

HỆ THỐNG FILE TRONG LINUX

Cấu trúc của hệ thống file

Mỗi hệ thống file có cấu trúc giống như một cái cây dựng ngược. Gốc của cây được đặt trên cùng và bên dưới là lá của nó.

Như đã đề cập ở trên, mỗi phân vùng khi được tạo ra đều có thể có một mount point. Công việc này thường được thi hành trong quá trình cài đặt. Để hiểu kỹ hơn về vấn đề này, hãy quan sát kiến trúc phân cấp của một hệ thống file trong Linux dưới đây:



The base directories



Directories that can be mount points for separate devices

Trong hình trên, gốc của kiến trúc phân cấp này là thư mục gốc “/”. Nó gần tương tự như “C:\” trong DOS ngoại trừ việc “C:\” chính là phân vùng đầu tiên của đĩa cứng đầu tiên, trong khi thư mục gốc “/” của Linux có thể là ánh xạ của bất kỳ phân vùng nào.

HỆ THỐNG FILE TRONG LINUX

The base directories

Các thư mục cơ sở là những thư mục con cấp 1 nằm ngay dưới thư mục gốc “/”. Chúng được tạo ra bởi một gói thường có tên là filesystem.



Tiến trình khởi động sẽ ánh xạ thư mục gốc đầu tiên nhằm giúp đỡ tất cả các thao tác tiếp theo như kiểm tra phân vùng, nạp module cho nhân...vv vì khi ánh xạ thư mục gốc xong thì các chương trình như: **fsck**, **insmod** hay **mount** mới có thể được sử dụng.

Để đảm bảo cho quá trình khởi động diễn ra chính xác, các thư mục **/dev**, **/bin**, **/sbin**, **/etc** và **/lib** bắt buộc phải là thư mục con của “/” và không thể là ánh xạ của bất kỳ phân vùng nào khác.

Sau đây là một số thư mục cơ sở và giải thích ngắn gọn ý nghĩa của chúng:

/bin và **/sbin**

Chứa những file cần thiết cho quá trình khởi động và những lệnh thiết yếu để duy trì hệ thống.

/dev

Chứa các định danh ánh xạ của thiết bị hoặc những file đặc biệt.

/etc

Chứa các file cấu hình của hệ thống và nhiều chương trình tiện ích.

/lib

Chứa các thư viện dùng chung cho các lệnh nằm trong **/bin** và **/sbin**. Và thư mục này cũng chứa các module của nhân.

HỆ THỐNG FILE TRONG LINUX

/mnt hoặc **/media**

Mount point mặc định cho những hệ thống file kết nối bên ngoài.

/proc

Lưu các thông tin của nhân, chỉ có thể ghi được nội dung trong thư mục **/proc/sys**.

/boot

Chứa nhân Linux để khởi động và các file system maps cũng như các file khởi động giai đoạn hai.

/home (tùy chọn)

Thư mục dành cho người dùng khác root. Thông tin khởi tạo thư mục mặc định của người dùng được đặt trong **/etc/skel/**

/root (tùy chọn)

Thư mục mặc định của người dùng root.

/tmp

Thư mục chứa các file tạm thời.

/usr

Thư mục chứa những file cố định hoặc quan trọng để phục vụ tất cả người dùng.

/usr/local hoặc **/opt** (tùy chọn)

Thư mục chứa các phần mềm cài thêm.

/var/www, **/var/ftp** hoặc **/srv** (Suse)

Thư mục chứa thông tin của các dịch vụ WEB hay FTP.

/var

Thư mục chứa các thông tin hay thay đổi như: spool và log

HỆ THỐNG FILE TRONG LINUX

Sự nhất quán trong định dạng và kiến trúc của hệ thống file

Để có thể lưu trữ và quản lý dữ liệu, mỗi phân vùng trên đĩa cứng đều phải được tạo ra một hệ thống file. Ngay trước khi khởi tạo, bao giờ người thiết lập cũng phải chỉ định kiểu định dạng của hệ thống file mới cần tạo.

Hiện nay, nhân Linux hỗ trợ rất nhiều kiểu định dạng của hệ thống file. Trong đó, kiểu hệ thống file **ext2** được coi là mặc định trong các hệ thống của Linux “Linux Native” (Trong nhiều hệ thống **ext3** được coi là mặc định nhưng thực tế **ext3** chính là **ext2** kèm thêm chức năng journal).

Một kiểu khác của hệ thống file cũng hay được dùng là SWAP. Kiểu định dạng hệ thống file này chỉ được dùng cho phân vùng swap.

The Second Extended File System

Ext2 là kiểu định dạng hệ thống file được thiết kế dựa trên việc quản lý các khối dữ liệu có kích thước 1KB (1024 byte), đây là kích thước mặc định và có thể thay đổi được. Có 3 loại khối như trên được định nghĩa trong ext2:

Superblocks

Lặp lại sau mỗi 8193 khối. Khối này chứa thông tin như: block-size, free inodes, last mounted time ...

Inodes

Chứa các con trỏ trỏ đến khối dữ liệu. 12 khối dữ liệu đầu tiên được truy cập trực tiếp từ con trỏ này. Nếu dữ liệu > 12KB thì các inodes gián tiếp sẽ được sử dụng.

Mỗi inode bao gồm 256 byte và chứa các thông tin về user, group, permissions và time stamp của dữ liệu mà nó quản lý.

Khối dữ liệu

Có thể là file hoặc thư mục với nội dung thật được chứa trong các khối này.

HỆ THỐNG FILE TRONG LINUX

Tiện ích định dạng

Do nhân Linux chỉ có thể đọc được các hệ thống file đã được định dạng từ trước nên để lưu trữ và quản lý dữ liệu trên các phân vùng mới, cần phải định dạng một hệ thống file trên đó thông qua các công cụ định dạng.

Để định dạng một phân vùng có kiểu hệ thống file là **ext2** bằng lệnh **mkfs.ext2** hay **mke2fs**. Tương tự như vậy với kiểu hệ thống file **xfs** (của Silicon Graphics) với lệnh **mkfs.xfs**.

Lệnh **mkfs** thực chất là một chương trình kiểm tra yêu cầu định dạng và lựa chọn đúng lệnh để thi hành. Cú pháp của mkfs là:

mkfs -t <fstype>



```
mkfs -t jfs /dev/hda12
```



```
mke2fs /dev/hda11 [or mkfs -t ext2 /dev/hda11]
```

Sự an toàn của hệ thống file

Nếu hệ thống file bị hỏng hoặc sai lệch, tiện ích **fsck** được sử dụng để chỉnh sửa lại các hư hỏng này tuy nhiên các hệ thống file này cần phải unmount trước đó để đảm bảo tính chính xác.

Cũng như **mkfs**, **fsck** thực chất chỉ kiểm tra các tham số của người dùng và lựa chọn đúng chương trình để thi hành, ví dụ: **fsck.ext2**, **fsck.ext3**



```
fsck -t reiserfs /dev/sdb10  
fsck.reiserfs /dev/sdb10
```

HỆ THỐNG FILE TRONG LINUX

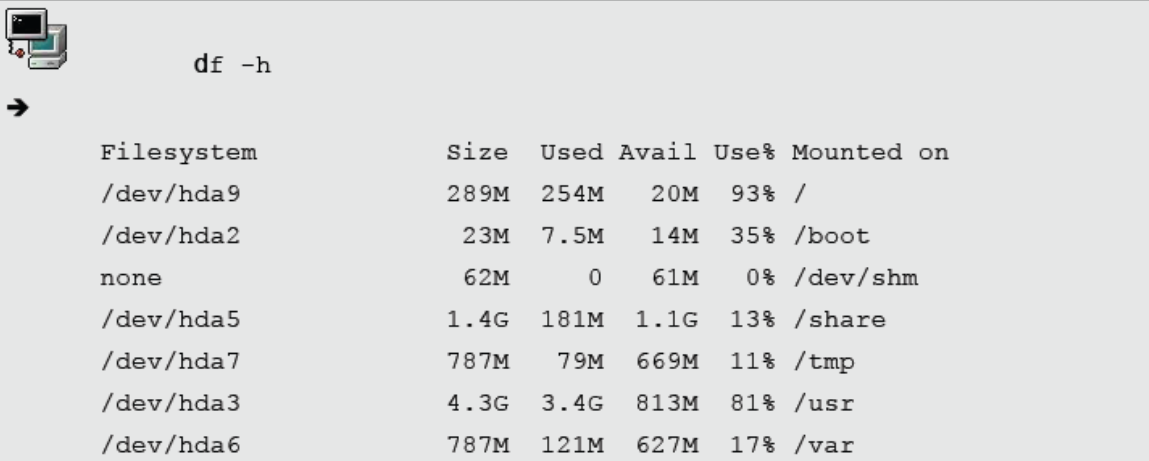
Kiểm tra dung lượng đĩa

Sử dụng mount và df

Cả hai lệnh trên đều cùng hoạt động ở cùng mức thiết bị. Hai lệnh **mount** và **umount** dùng để quản trị các hệ thống file đã gắn kết trong file **/etc/mtab**.

Nếu sử dụng **mount** không tham số, tất cả các hệ thống file được gắn kết trong hệ thống sẽ được liệt kê ra màn hình. Kết quả giống như trong file **/etc/mtab**. Ngoài ra, nhân cũng lưu giữ thông tin về hệ thống file đã được kết nối trong **/proc/mount**.

Để xem thêm thông tin về điểm kết nối hiện tại có thể sử dụng lệnh **df**. Lệnh này cho phép hiển thị thêm dung lượng đĩa đã sử dụng và dung lượng còn trống. Đơn vị kích thước để hiển thị là 1K.



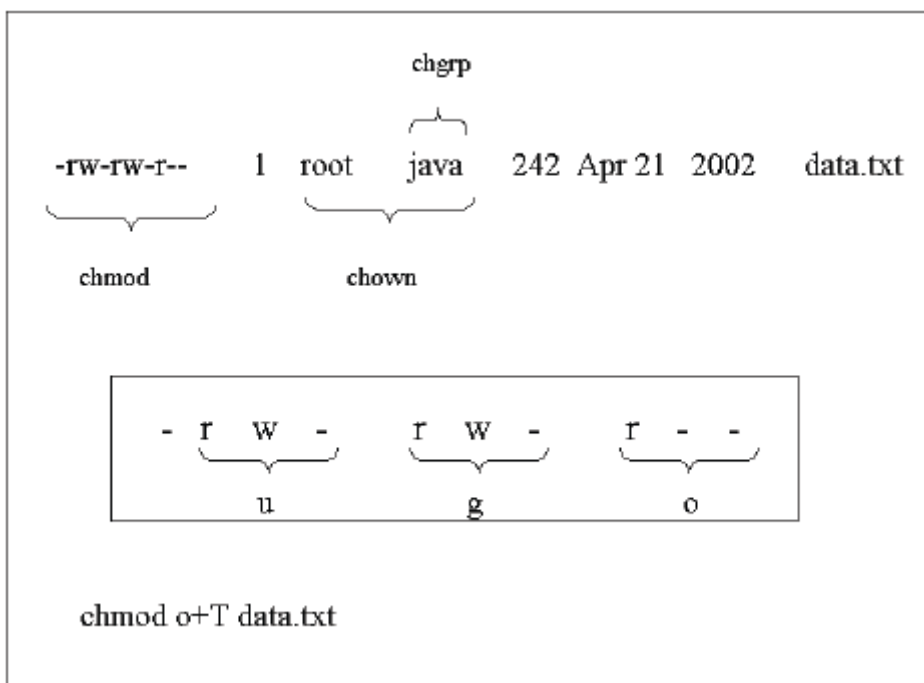
```
df -h
→
Filesystem      Size  Used Avail Use% Mounted on
/dev/hda9       289M  254M   20M  93% /
/dev/hda2       23M   7.5M  14M  35% /boot
none            62M    0    61M   0% /dev/shm
/dev/hda5       1.4G  181M  1.1G  13% /share
/dev/hda7       787M   79M  669M  11% /tmp
/dev/hda3       4.3G  3.4G  813M  81% /usr
/dev/hda6       787M  121M  627M  17% /var
```

Sử dụng du

Tiện ích này được sử dụng để hiển thị không gian đĩa được sử dụng nhưng ở mức thư mục. Vì vậy, du cũng không thể hiển thị khoảng trống còn thừa của đĩa.

HỆ THỐNG FILE TRONG LINUX

Quyền truy xuất File, Thư mục



Thay đổi quyền truy xuất và chủ sở hữu

Quyền truy xuất file, thư mục và chủ sở hữu được định nghĩa để quy định cách thức truy cập dữ liệu trong hệ thống.

Để thay đổi quyền truy cập, sử dụng lệnh **chmod**. Có ba nhóm đối tượng chính được tác động bởi quyền truy cập là:

<i>u</i>	Người dùng sở hữu
<i>g</i>	Nhóm người dùng sở hữu
<i>o</i>	Không thuộc hai đối tượng trên

Ví dụ:

```
-rw-rw-r-- 1 jade sales 24880 Oct 25 17:28 libcgic.a
```

HỆ THỐNG FILE TRONG LINUX



```
chmod g=r,o-r libcgic.a  
chmod g+w libcgic.a
```



```
chown root libcgic.a  
chgrp apache libcgic.a
```

Tùy chọn hay dùng với chmod, chown và chgrp là -R cho phép thay đổi trong cả các thư mục, file bên trong thư mục chỉ định.

Ngoài cách sử dụng ký tự đại diện cho các quyền: read=r, write=w, execute=x, chmod cho phép sử dụng một bộ số hệ bát phân để thay đổi quyền theo bảng sau:

<i>read</i>	4
<i>write</i>	2
<i>execute</i>	1

user	group	other
<i>rwX</i>	<i>r-x</i>	<i>rw-</i>
$4+2+1=7$	$4+1=5$	$4+2=6$

Quyền truy xuất chuẩn

Các hệ thống UNIX tạo ra file và thư mục với quyền truy xuất chuẩn như sau::

<i>Files</i>	<i>666</i>	<i>-rw-rw-rw-</i>
<i>Directories</i>	<i>777</i>	<i>-rwxrwxrwx</i>

umask



HỆ THỐNG FILE TRONG LINUX

Là khái niệm được thiết lập để chỉ định quyền truy xuất mặc định cho các file và thư mục mới tạo **đối với mỗi người dùng**. umask là một mặt nạ gồm một bộ các số hệ bát phân. Khi đó, quyền truy xuất mặc định của các file và thư mục đối với mỗi người dùng được tính theo công thức sau:

$$\text{Final Permissions} = \text{Standard Permissions (logical AND)} (\text{NOT})\text{Umask}$$

Quyền truy cập SUID

Là quyền truy cập được thiết lập bởi root cho phép người dùng bình thường có thể thi hành một lệnh như là root. Quyền này được thiết lập với tên là s (nằm ở vị trí x của nhóm u) và được gán số hệ bát phân là 4000.

Quyền truy cập SGID

Là quyền truy cập cho phép người dùng thuộc nhóm sở hữu có thể thi hành lệnh mà không cần dùng **newgrp** để chuyển nhóm. Quyền này được thiết lập với tên là s (nằm ở vị trí x của nhóm g) và được gán số hệ bát phân là 2000.

Bit đánh dấu (The sticky bit)

Quyền này được thiết lập với tên là t (nằm ở vị trí x của nhóm o) và được gán số hệ bát phân là 1000. Quyền này được thiết lập để:

- Cho phép các thư mục cấm người dùng xóa file trừ phi họ là chủ sở hữu.
- Cho phép file được thi hành hoặc nạp vào bộ nhớ nhanh hơn.

Thực hành

Filesystem

1. Xóa phân vùng được ánh xạ vào /data của bài trước, tạo ra 2 phân vùng mới có kiểu định dạng của hệ thống file là ext2 và reiserfs.

2. Tạo 2 thư mục con trong /mnt và ánh xạ hai phân vùng mới vào.

```
mkdir /mnt/ext2
```

```
mkdir /mnt/reiserfs
```

3. Sử dụng các lệnh mount, df, fsck để kiểm tra đối với 2 phân vùng mới tạo.

4. Chuyển đổi từ ext2 sang ext3 bằng lệnh tune2fs

File permissions

1. Login bằng 1 người dùng không phải root và tạo 1 file mới bằng lệnh touch. Kiểm tra xem quyền truy xuất của file này là gì?

2. Thay đổi umask thành 027. Quyền truy xuất mặc định sẽ là gì?

3. Nơi nào sẽ thiết lập giá trị mặc định của umask? /etc/profile, /etc/bashrc...

4. Thêm 2 người dùng mới user1, user2 với password tương ứng. Tạo nhóm mới sales. Và thêm 2 người dùng mới tạo vào nhóm này.

5. Tạo thư mục /news sở hữu bởi nhóm sales và có quyền 770 cho thư mục này. Sau đó đặt GID cho thư mục này.

6. Kiểm tra các tính chất của GID với user1 và user2.

7. Thêm Sticky-Bit cho thư mục /news. Kiểm tra tính chất của bit này.

DÒNG LỆNH

DÒNG LỆNH

Khái quát

Sử dụng dòng lệnh là cách cơ bản để tương tác với hệ thống máy tính. Bộ biên dịch shell (hệ vỏ) thông dịch các lệnh được nhập vào từ bàn phím. Dấu nhắc shell (\$ hoặc # đối với người quản trị hệ thống) cho biết hệ thống đã sẵn sàng hoạt động.

Shell còn là một môi trường lập trình cho phép thực hiện các lệnh khởi động. Chương trình shell được gọi là script (kịch bản).

Most Common shells	
The Bourne shell	/bin/sh
The Bourne again shell	/bin/bash
The Korn shell	/bin/ksh
The C shell	/bin/csh
Tom's C shell	/bin/tcsh

Do bash shell là một trong những shell thông dụng nhất trong cộng đồng linux, vì thế LPI (Linux Professional Institute) tập trung chủ yếu vào bash shell.

Tương tác với SHELL

Các câu lệnh thực hiện trên shell có dạng sau:

command [options] {arguments}

- Hiển thị kí tự ra màn hình

Bash shell sử dụng lệnh **echo** để hiển thị kí tự ra màn hình

```
echo "this is a short line"
```

- Đường dẫn tuyệt đối/tương đối

DÒNG LỆNH

Shell thông dịch từ đầu tiên của bất kỳ dòng lệnh nào như là một câu lệnh. Nếu dòng lệnh có một **đường dẫn tuyệt đối hoặc tương đối** đến câu lệnh thì câu lệnh sẽ được thực thi. Nếu từ đầu tiên không có kí tự “/” thì shell sẽ tìm kiếm ở các thư mục đã được khai báo trong nội dung biến môi trường PATH và thực hiện chương trình có tên trùng với câu lệnh.

Ví dụ nếu tham biến PATH chỉ chứa các thư mục **/bin** và **/usr/bin** thì câu lệnh **xeyes** sẽ không được tìm thấy khi mà nó nằm trong **/usr/X11R6/bin/xeyes** và vì thế đường dẫn tuyệt đối là cần thiết để cho câu lệnh này được thực thi.

```
/usr/X11R6/bin/xeyes
```

Người dùng có thể sử dụng đường dẫn tương đối thay cho đường dẫn tuyệt đối trong khi thực hiện một câu lệnh. Ví dụ nếu người dùng đang truy cập vào thư mục chứa chương trình **xeyes** thì họ có thể sử dụng câu lệnh sau:

```
./xeyes
```

Biến môi trường của Shell

Các biến của Shell giống như các biến được sử dụng trong các ngôn ngữ máy tính khác. Các tên biến được giới hạn trong các kí tự chữ số. Ví dụ CREDIT=300 có nghĩa là gán giá trị 300 cho biến có tên là CREDIT

1. Khởi tạo một biến	Variable-Name=giá trị (không có dấu cách)
2. Tham chiếu một biến	\$Variable-Name

```
CREDIT=300  
echo $CREDIT
```

Export, Set và Env:

DÒNG LỆNH

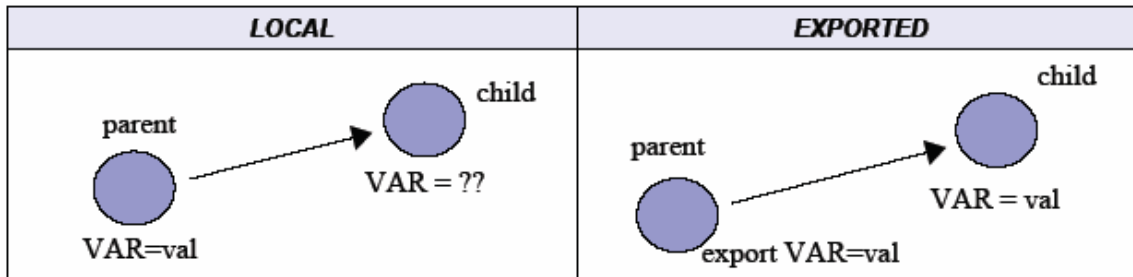
Có hai loại biến: biến cục bộ và biến xuất (có thể gọi là biến toàn cục địa phương, thực tế khái niệm này không thể đồng nghĩa với khái niệm biến toàn cục trong các ngôn ngữ lập trình. Tuy vậy để cho ngắn gọn, tất cả các thể hiện biến toàn cục xuất hiện trong tài liệu này đều có ý nghĩa như biến xuất).

Biến cục bộ chỉ được truy cập bởi shell hiện thời. Trong khi đó biến xuất sẽ được truy cập bởi cả shell và bất kỳ tiến trình con của shell này.

Các lệnh **set** và **env** dùng để hiển thị các biến đã được định nghĩa

Các lệnh set và env	
set	Hiển thị tất cả các biến
env	Hiển thị tất cả các biến xuất

Một biến được gọi là biến toàn cục khi bất kỳ tiến trình con nào cũng có thể tham chiếu đến nó.



Ví dụ: Tạo biến CREDIT là biến toàn cục. Hiển thị nó với lệnh **set** hoặc **env**.

```
export CREDIT
env | grep CREDIT
```

Khởi tạo một shell mới (tiến trình con) và kiểm tra xem biến CREDIT có được truy cập đến không? Chúng ta có thể khởi tạo bất kỳ shell mới và xem biến CREDIT có được truy cập đến không?

DÒNG LỆNH

Các biến được định nghĩa trước	Ý nghĩa
DISPLAY	Được sử dụng bởi X để xác định vị trí thực hiện một ứng dụng khách (client)
HISTFILE	Đường dẫn đến file của các người dùng <code>.bash_history</code>
HOME	Đường dẫn đến thư mục dành riêng (home) của người dùng
LOGNAME	Tên được sử dụng bởi người dùng để truy nhập
PATH	Chứa những thư mục sẽ được tìm kiếm bởi shell khi người dùng thực hiện chương trình mà không chỉ ra đường dẫn
PWD	Thư mục làm việc hiện thời
SHELL	Shell được sử dụng (bash thông dụng nhất)
TERM	Mô phỏng thiết bị cuối hiện thời

Các biến đặc biệt

Một số biến liên quan đến việc quản lý tiến trình

\$! Hiển thị mã tiến trình (PID) của tiến trình con cuối cùng

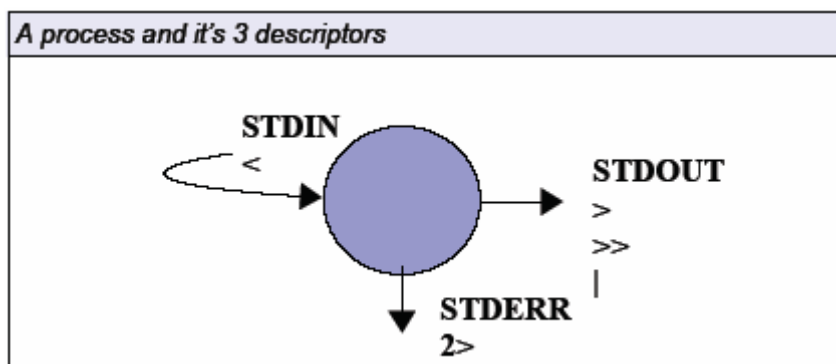
\$\$ Hiển thị mã tiến trình (PID) của shell đang thực thi

\$? Có giá trị 0 nếu lệnh cuối cùng được thực hiện thành công và 1 nếu ngược lại

Xuất, nhập, đổi hướng

Các tiến trình UNIX thông thường mở 3 dạng mô tả file chuẩn cho phép nó thực hiện việc xuất, nhập và báo lỗi. Các dạng mô tả chuẩn này có thể được định nghĩa lại bởi bất kỳ tiến trình nào. Trong hầu hết các trường hợp, mô tả **stdin** là bàn phím, và hai dạng mô tả xuất + báo lỗi (**stdout** và **stderr**) là màn hình.

DÒNG LỆNH



stdin	0
stdout	1
stderr	2

- Đổi hướng stdout

program > *file*

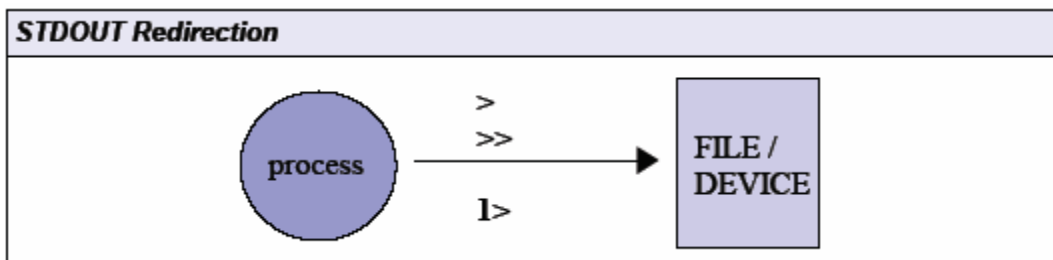
Dữ liệu theo hướng từ trái sang phải

```
fdisk -l > partions.txt
```

Câu lệnh này sẽ thực hiện tiện ích **fdisk** và kết quả đầu ra sẽ được ghi vào file *partions.txt*. Kết quả không được hiển thị ra màn hình. Chú ý rằng shell sẽ thực hiện câu lệnh này bắt đầu từ bên phải. Như vậy, file *partions.txt* sẽ được tạo ra nếu như nó chưa tồn tại và sẽ bị ghi đè vào khi toán tử '>' được dùng.

Toán tử '>>' sẽ bổ sung thêm kết quả vào nội dung file.

DÒNG LỆNH



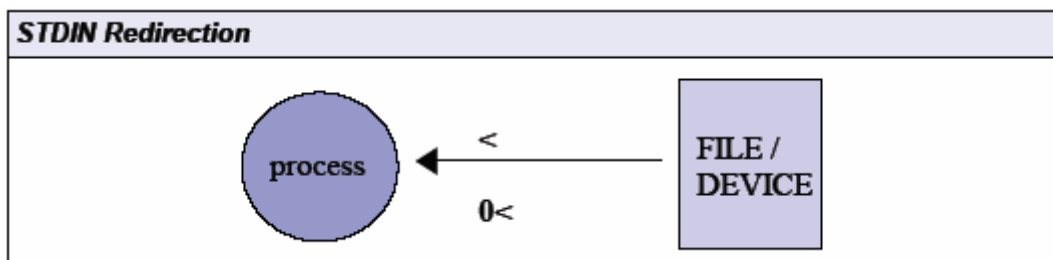
- **Đổi hướng stdin**

program < file

Trong trường hợp này dữ liệu theo hướng từ phải sang trái. Toán tử '<' chỉ được sử dụng cho **stdin** và không thể dùng cho **stdout**.

Nếu file instruction chứa trên mỗi dòng các kí tự *p*, *m*, và *q* thì trong ví dụ sau đây **fdisk** sẽ in bảng phân vùng (partition) của */dev/hda*, in tiện ích trợ giúp, và cuối cùng là thoát khỏi câu lệnh.

```
fdisk /dev/hda < instructions
```



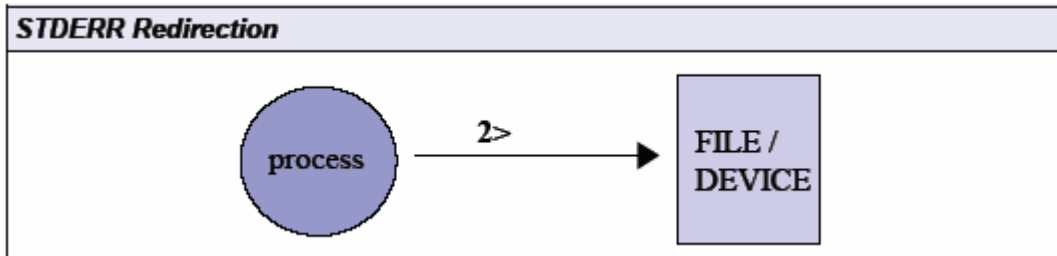
- **Đổi hướng stderr**

program 2> errorfile

stdin, stdout, và stderr được đại diện bằng 0, 1, và 2 tương ứng. Câu lệnh trên cho phép chúng ta chọn luồng stderr.

DÒNG LỆNH

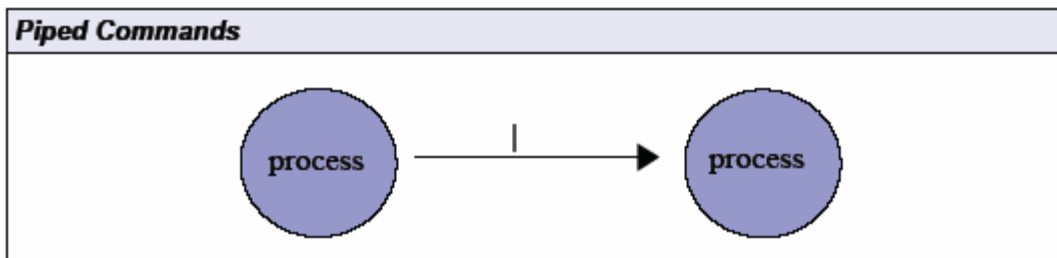
```
find / 2> /dev/null
```



- **Các lệnh đường ống**

Program1| Program2

Các đường ống (pipe) được đại diện bằng kí hiệu "|". Dòng dữ liệu chuyển từ trái sang phải. Hình sau đây minh họa stdout của một tiến trình được chuyển hướng đến stdin của một tiến trình khác như thế nào.



```
cat /var/log/messages | less
```

Các chuyển hướng của dữ liệu xuất được phân tách từ phải sang trái, do đó các lệnh sau là không tương đương

DÒNG LỆNH

Do-command 2>&1 >logfile

Do-command >logfile 2>&1

Dấu ngoặc và Các ký tự Đa nghĩa (Metacharacter)

Metacharacter là các ký tự có nghĩa đặc biệt trong shell. Chúng được dùng chủ yếu cho *file globbing*, tức là đối sánh một vài file hoặc tên thư mục bằng một số lượng tối thiểu các ký tự.

Các ký tự nhập (<), xuất (>), và đường ống (|) cũng là các ký tự đặc biệt và ký tự \$ được dùng cho các biến. Các ký tự đặc biệt này sẽ không được liệt kê hết ở đây.

Các ký tự đại diện (wildcard)

- Ký tự * có thể đại diện cho 0 hoặc một số ký tự tùy ý

*ls /usr/bin/b** hiển thị tất cả các chương trình bắt đầu bằng ký tự 'b'

- Ký tự ? đại diện cho một ký tự tùy ý

*ls usr/bin/?b** hiển thị tất cả các chương trình có ký tự 'b' ở vị trí thứ 2

Các miền (range)

- [] được dùng để định nghĩa một miền các giá trị

ls a[0-9] hiển thị tất cả các file bắt đầu bằng ký tự 'a' và có một chữ số ở vị trí thứ 2.

*ls [!Aa]** hiển thị tất cả các file không bắt đầu bằng ký tự 'a' hoặc 'A'

DÒNG LỆNH

- {xâu1,xâu2} mặc dù chúng không được dùng để đại diện một họ tên file nhưng chúng có thể sử dụng để đối sánh với tên những file đã có.

```
ls index.{htm,html}
```

Các dấu ngoặc (quote) và mã escape

Ý nghĩa đặc biệt của các metacharacter có thể bị huỷ bỏ bằng các ký tự escape-chúng cũng là các metacharacter.

Dấu vạch chéo ngược (\) là một ký tự đặc biệt và huỷ bỏ ý nghĩa của tất cả các metacharacter yêu cầu shell thông dịch chúng.

Dấu ngoặc đơn (' ') huỷ bỏ nghĩa của tất cả các metacharacter ngoại trừ dấu vạch chéo ngược.

Dấu ngoặc kép (" ") có tác dụng yếu nhất nhưng cũng có thể huỷ bỏ phần lớn ý nghĩa đặc biệt của các ký tự nằm trong dấu ngoặc kép ngoại trừ đường ống (|), dấu vạch chéo ngược, và một biến (\$var).

Dấu nháy

Dấu nháy này giống dấu huyền của Tiếng Việt và thường được đặt cạnh số 1 của bàn phím đầy đủ.

Cặp dấu nháy (`) sẽ thực hiện câu lệnh nằm bên trong. Ví dụ sau đây sẽ định nghĩa biến TIME sử dụng lệnh **date**

```
TIME="Today's date is `date +%a:%d:%b`"  
echo $TIME  
Today's date is Sun:15:Jul
```

DÒNG LỆNH

Một cách khác để thực hiện câu lệnh giống như sử dụng các dấu nhảy đó là \$(). Ví dụ dưới đây sẽ thực hiện câu lệnh ở bên trong và gán giá trị trả về vào biến TIME.

```
TIME=$(date)
```

Lịch sử dòng lệnh

Để xem danh sách các câu lệnh đã được sử dụng từ trước chúng ta có thể dùng **bash** gắn liền với lệnh **history**

```
history
1.      ls
2.      grep 500 /etc/passwd
```

Chúng ta có thể gọi lại các lệnh đã sử dụng bằng cách dùng mũi tên lên và xuống trên bàn phím. Ngoài ra còn có các liên kết phím emacs cho chúng ta thực hiện và sửa đổi các lệnh trước đó.

Emacs Key Bindings for Editing the Command History	
Ctrl+p	Lên trên 1 dòng
Ctrl+n	Xuống dưới 1 dòng
Ctrl+b	Quay lại (sang trái) 1 ký tự
Ctrl+f	Đi tiếp (sang phải) 1 ký tự
Ctrl+a	Về cuối dòng
Ctrl+e	Về đầu dòng

DÒNG LỆNH

Dấu chấm than (!) có thể được dùng để thực hiện các lệnh trước đó

Ví dụ

!x	Thi hành lệnh gần nhất trong lịch sử lệnh mà có ký tự bắt đầu là 'x'
!2	Thi hành lệnh có số thứ tự = 2 trong lịch sử lệnh
!-2	Thi hành lệnh ngay trước lệnh vừa thi hành
!!	Thi hành lệnh vừa chạy
^string1^string2	Thi hành lệnh vừa chạy và thay thế string1 bởi string2

Một số lệnh khác

Bí danh

Chúng ta có thể tạo các bí danh cho các lệnh sử dụng nhiều tham số. Cách thức để tạo một bí danh là như sau

```
alias myprog='command [options]{arguments}'
```

Sự kết thúc câu lệnh

Bằng cách ấn phím **TAB**, shell sẽ kết thúc câu lệnh mà chúng ta đang gõ vào

Bằng cách chỉ gõ **alias** tại một dòng lệnh, chúng ta sẽ có danh sách của các bí danh đã được định nghĩa

<< là sự đổi hướng cho kết thúc file (EOF)

Ví dụ câu lệnh

```
cat << stop
```

DÒNG LỆNH

sẽ chấp nhận các giá trị nhập chuẩn cho đến khi từ 'stop' được đưa vào.

Thực hành

Stdin-stdout-stderr

Gỡ các câu lệnh sau đây và đưa ra các kết quả thực thi (nếu có thể) sử dụng các sơ đồ giống như những sơ đồ đã được dùng trong chương này

```
ls /etc ; df > /tmp/out.1
(ls /etc ; df) > /tmp/out.2
find /etc -type f 2> /dev/null | sort
tr [a-z] [A-Z] < /etc/passwd | sort > /tmp/passwd.tmp
cat /tmp/passwd.tmp | tr [A-Z] [a-z]
```

Dòng lệnh

1. Hiện thị tất cả các file trong /usr/X11R6/bin mà không bắt đầu với ký tự 'x'

```
ls /usr/X11R6/bin/[!x]*
```

2. Câu lệnh **xterm** có các lựa chọn sau:

-bg <màu> thiết lập màu nền

-fg <màu> thiết lập màu chữ

-e <câu lệnh> thực hiện câu lệnh

Thiết lập một bí danh mới sao cho câu lệnh **su** mở một xterm với các màu mới và lời nhắc cho mật khẩu chủ

```
alias su="xterm -bg orange -fg brown -e su -u &"
```

Bạn sẽ lưu trữ bí danh này ở nơi nào trên hệ thống?

DÒNG LỆNH

3. Bạn có thể mã hoá các file sử dụng câu lệnh **uuencode**. File mã hoá sẽ được chuyển hướng đến stdout

Ví dụ: `uuencode /bin/bash super-shell > uufile` sẽ mã hoá **/bin/bash** và tạo ra một file **super-shell** khi thực hiện **uudecode** đối với *uufile*

- Gửi /bin/bash được mã hoá đến người dùng cục bộ (trong trường hợp này chúng ta có thể sử dụng **uuencode** và đường ống | hoặc lưu lại kết quả mã hoá vào file *uufile* và sử dụng đối hướng STDIN <)
- Chia file mã hoá thành 5 file nhỏ

```
uuencode /bin/bash super-shell > uufile
split -b 150000 uufile base-name.
```

Lệnh này sẽ tạo ra các file `base-name.aa`, `base-name.ab`,...

Thực hiện lệnh sau để ghép nối các file mã hoá đã được chia nhỏ thành file dữ liệu ban đầu (`unsplit`)

```
cat base-name.* > uufile.new
```

Cuối cùng giải mã file đó và kiểm tra xem nó có hoạt động không

```
uudecode uufile.new
```

Câu lệnh này sẽ tạo ra một file nhị phân gọi là **super-shell**

4. Tiện ích nào sẽ tìm ra đường dẫn tuyệt đối của một file nhị phân thông qua quá trình kiểm tra biến PATH?

Các biến

1. Thực hiện các câu lệnh sau

DÒNG LỆNH

Khởi tạo giá trị 'virus' cho biến ALERT

```
ALERT=virus
```

Kiểm chứng xem biến ALERT đã được khởi tạo chưa bằng lệnh **set**?

```
set |grep ALERT
```

Hoặc có thể hiển thị ALERT bằng cách dùng lệnh **env**

Tiếp theo, gõ 'bash'. Bạn có thể truy cập vào biến ALERT?

```
bash
```

```
echo $ALERT
```

Xem giá trị của ALERT là trống hay không?

Gõ exit (hoặc ^D) để trở về phiên làm việc của bạn.

Sử dụng lệnh **export** để tạo biến ALERT là biến toàn cục

```
export ALERT
```

Kiểm chứng ALERT đã là biến toàn cục chưa?

```
env | grep ALERT
```

(v) Khởi tạo một **bash** shell mới và chắc chắn rằng ALERT được định nghĩa trong shell mới

```
bash
```

```
echo $ALERT
```

Trong shell mới này, định nghĩa lại biến ALERT

```
export ALERT=green
```

Thoát khỏi shell này. Giá trị của biến ALERT trong shell ban đầu sẽ là bao nhiêu?

DÒNG LỆNH

2. Tại lời nhắc câu lệnh gõ các dòng sau:

```
CREDIT01=300;CREDIT02=400
for VAR in CREDIT01 CREDIT02; do echo $VAR;done
```

Chú ý rằng biến VAR sẽ được tham chiếu bằng \$VAR

(i) Thực hiện lại lệnh này

(ii) Thực hiện lệnh này nhưng thay thế CREDIT01 bằng \$CREDIT01

3. Sử dụng các dấu ngoặc thích hợp để thay đổi biến PS1 sao cho nó gồm đường dẫn tuyệt đối đến thư mục bạn đang làm việc

(Gợi ý: giá trị của PS1 là [\u@\W]\\$, do đó bạn chỉ cần thay thế \W bằng \w)

```
PS1=' [ \u@\h \w ]\$ '
```

Biến PS2 có giá trị như thế nào?

QUẢN LÝ FILE

Di chuyển quanh hệ thống file

Các đường dẫn tuyệt đối và tương đối

Một thư mục hoặc một file có thể truy cập bằng đường dẫn tuyệt đối bắt đầu từ thư mục gốc (/) hoặc đường dẫn tương đối bắt đầu từ thư mục hiện thời.

Đường dẫn tuyệt đối: độc lập với thư mục hiện thời của người dùng và bắt đầu với /

Đường dẫn tương đối: phụ thuộc vào thư mục hiện thời của người dùng và không bắt đầu với /

Đối với một hệ thống file có cấu trúc bất kỳ, có một số tiện ích giúp chúng ta có thể duyệt toàn bộ hệ thống

pwd: đưa ra đường dẫn tuyệt đối về vị trí của bạn trong hệ thống

cd: thay đổi thư mục

Tìm kiếm file và thư mục

Chúng ta sẽ tìm hiểu các tiện ích **find**, **which**, **whereis** và **locate**

- **find**

Cú pháp:

```
find <DIRECTORY> <CRITERIA> [-exec <COMMAND> {} \; ]
```

Tham biến *DIRECTORY* sẽ cho biết vị trí bắt đầu tìm kiếm và *CRITERIA* có thể là tên một file hoặc một thư mục mà chúng ta đang tìm kiếm

Ví dụ



```
find /usr/X11R6/bin -name "x*"  
find / -user 502
```

Các dòng đối sánh sẽ được hiển thị ở đầu ra chuẩn. Kết quả xuất này có thể được thực hiện tiếp theo đó. Ví dụ xoá file hoặc thay đổi quyền hạn. Tiện ích **find** có lựa chọn **-exec** cho phép chúng ta thực hiện điều đó. Ví dụ xoá tất cả các file thuộc về người dùng 502



```
find / -type f -user 502 -exec rm -f {} \;
```

- **xargs**

Tiện ích này thường xem như là một công cụ đi kèm với **find**. Thực tế **xargs** sẽ xử lý mỗi dòng của kết quả xuất chuẩn như một tham biến cho một tiện ích khác. Chúng ta có thể dùng **xargs** để xoá tất cả các file thuộc về một người dùng bằng lệnh sau



```
find / -type f -user 502 | xargs rm -f
```

Các câu lệnh chắc chắn như **rm** không thể xử lý với quá nhiều tham số. Chúng ta có thể xoá toàn bộ các file trong một thư mục với lệnh sau

```
ls | xargs rm -f
```

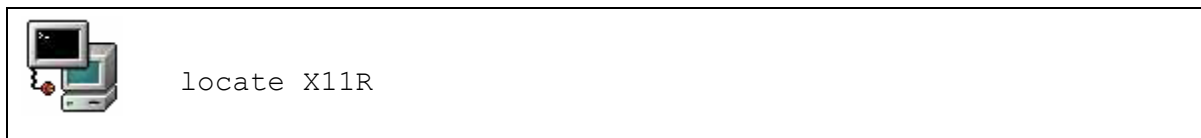
Common criteria switches for find	
-type	specify the type of file
-name	name of the file
-user	user owner
-atime, ctime, mtime	access, creation and modified times (multiples of 24 hrs)
-amin, cmin, mmin	access, creation and modified times (multiples of 1 min)
-newer FILE	files newer than FILE

- **locate**

Cú pháp:

```
locate <STRING>
```

Tiện ích **locate** cho phép hiển thị tất cả các file và thư mục thoả mãn biểu thức (expression)



Với tiện ích này quá trình tìm kiếm sẽ nhanh hơn rất nhiều. Thực tế **locate** sẽ truy vấn cơ sở dữ liệu **/var/lib/slocate**. Cơ sở dữ liệu này sẽ được cập nhật hàng ngày thông qua cron job dựa trên lệnh **updatedb**

Khi thực hiện **updatedb** từ dòng lệnh thì file **/etc/updatedb.cron** sẽ được đọc để xác định hệ thống file đã được chỉnh sửa (tức là NFS) và các thư mục (tức là **/tmp**)

- **which**

Cú pháp:

```
which string
```

Tiện ích này sẽ đưa ra đường dẫn tuyệt đối đối với file gọi là **string** bằng cách chỉ kiểm tra các thư mục được định nghĩa trong biến **PATH** của người dùng. Vì thế **which** chỉ được dùng để tìm kiếm các lệnh.

- **whereis**

QUẢN LÝ FILE

Cú pháp

`whereis string`

Tiện ích này sẽ đưa ra đường dẫn tuyệt đối đối với các file nguồn, nhị phân, và tài liệu phù hợp với **string** bằng cách kiểm tra biến PATH cũng như các vị trí hay được sử dụng.

Các lựa chọn thường được dùng của **ls**

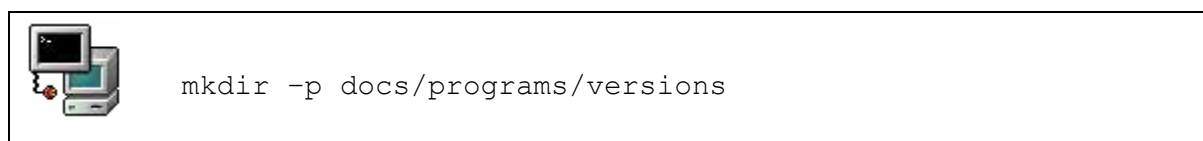
<i>Most common options for ls</i>	
-l	show inode
-h	print human readable sizes
-n	list UIDs and GIDs
-p	append descriptor (/= @) to list
-R	recursively display content of directories
-S	sort by file size
-t	sort by modification time (similar to -c)
-u	show last access time

Làm việc với thư mục

*Tạo thư mục với lệnh **mkdir***

Khi tạo một thư mục chúng ta có thể thiết lập quyền truy nhập với lựa chọn **-m**. Một lựa chọn có ích khác đó là **-p** sẽ tự động tạo tất cả các thư mục con khi cần.

Ví dụ:



Xoá các thư mục

Để xoá một thư mục chúng ta có thể sử dụng lệnh **rmdir** hoặc **rm**. Nếu bạn đang ở thư mục gốc bạn có thể dùng lựa chọn **-f** để xoá tất cả các file.

QUẢN LÝ FILE

Chú ý: `rm -rf /dir1/*` xoá tất cả các file và các thư mục con và để `dir1` là thư mục trống

`rm -rf /dir1/` xoá tất cả các file và các thư mục con bao gồm cả `dir1`

Sử dụng cp và mv

cp

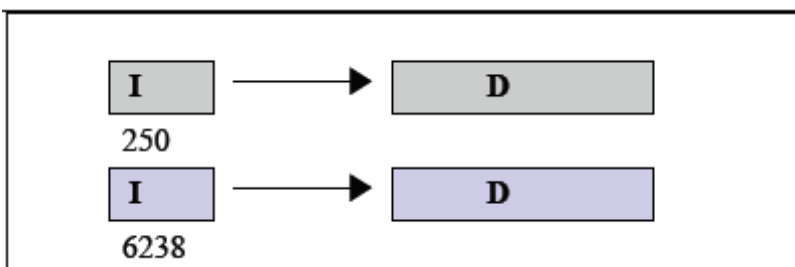
Cú pháp

```
cp [options] file1 file2
```

```
cp [options] file1 directory
```

Chú ý rằng `cp file1 file2` tạo một bản copy mới của `file1` và không làm thay đổi `file1`

Hình minh hoạ: file1 với inode 250 sẽ được copy sang file 2, sao y dữ liệu đến một vùng dữ liệu mới và tạo inode mới 6238 cho file2.



Chúng ta cũng có thể copy một số file đến một thư mục khác bằng cách dùng danh sách liệt kê hoặc ký tự đại diện. Bảng sau đây sẽ hiển thị các lựa chọn thường được sử dụng

QUẢN LÝ FILE

Most common options for cp	
-d	do not follow symbolic link (when used with -R)
-f	force
-i	interactive, prompt before overwrite
-p	preserve file attributes
-R	recursively copy directories

Chú ý: `cp -r /mydir/* /dir2/` sẽ copy tất cả các file và thư mục con ngoại trừ `mydir`

`cp -r /mydir/ /dir2/` sẽ copy tất cả các file và thư mục con bao gồm cả `mydir`

mv

Cú pháp:

```
mv [options] oldname newname
mv [options] source destination
mv [options] source directory
```

Lệnh **mv** có thể di chuyển hoặc đổi tên các file và thư mục. Nếu *oldname* là một file và *newname* là một thư mục thì file *oldname* sẽ được di chuyển đến thư mục này.

Nếu *source* và *destination* cùng nằm trên một hệ thống file thì file không được copy nhưng thông tin về inode sẽ được cập nhật để xác định vị trí mới. Các lựa chọn thông thường là `-f` để ghi đè và `-i` truy vấn tương tác.

Hard links và symbol links

Các liên kết tượng trưng

Một liên kết tắt mềm đến một file hoặc một thư mục tạo một inode mới chỉ đến cùng một vùng dữ liệu.

QUẢN LÝ FILE

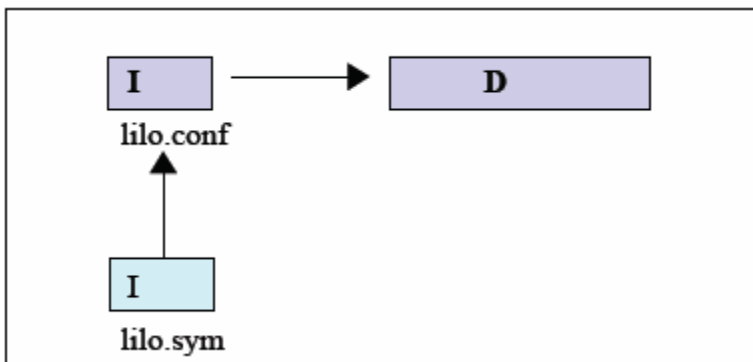


```
ln -s lilo.conf lilo.sys
```

Đây là danh sách những file của câu lệnh trên. Chú ý rằng giá trị tham chiếu là 1 cho cả 2 file.

```
-rw----- 1 root root 223 Nov 9 09:06 lilo.conf  
lrwxrwxrwx 1 root root 9 Nov 9 09:06 lilo.sym -> lilo.conf
```

Hình 2: Một liên kết tắt mềm đến một file



Các liên kết tắt mềm có thể được tạo thông qua các hệ thống file khác nhau

Các liên kết tắt cứng

Một liên kết tắt cứng là một tên được tạo thêm cho cùng một inode và giá trị tham chiếu của file đó sẽ được tăng thêm 1 cho mọi liên kết tắt cứng mới



```
ln lilo.conf lilo.link
```

QUẢN LÝ FILE

Trong bảng dưới đây chú ý rằng giá trị tham chiếu là 2 và cả 2 file có cùng kích thước. Thực tế chúng hoàn toàn giống nhau.

```
-rw----- 2 root  root  223  Nov  9 09:06 lilo.conf  
-rw----- 2 root  root  223  Nov  9 09:06 lilo.link
```

Các liên kết tắt cứng chỉ có thể được tạo trong cùng một hệ thống file.

Touching và dd-ing

touch

Một cách khác để tạo hoặc thay đổi một file là sử dụng touch

Cú pháp:

```
touch {options} file(s)
```

Nếu *file* chưa có thì nó sẽ được tạo mới. Chúng ta có thể thay đổi thời gian truy cập file bằng lựa chọn `-a`, `-m` thay đổi thời gian sửa đổi file, và `-r` dùng để sử dụng các thuộc tính thời gian của file khác.

Ví dụ

```
touch file1.txt file2.txt          tạo các file mới
```

```
touch myfile -r /etc/lilo.conf     myfile sẽ lấy các thuộc tính thời  
gian của lilo.conf
```

Tạo một file **-errors** sử dụng lựa chọn `-:`

```
touch -- -errors
```

dd

QUẢN LÝ FILE

Lệnh này sẽ copy 1 file với kích thước khối I/O có thể thay đổi. Lệnh này cũng được dùng để thực hiện các quá trình chuyển đổi (giống như **tr**). Các lựa chọn chính là **if=** (file nhập), **of=** (file xuất), và **conv=** (chuyển đổi)

Các khoá chuyển đổi có thể là: **lcase**, **ucase**, và **ascii**

Ví dụ

```
dd if=/mnt/cdrom/images/boot.img of=/dev/fd0
```

Thực hành

Điều hướng file

Tạo một thư mục mới **/bin** trong **/tmp**

```
mkdir /tmp/bin
```

Trong **/tmp/bin** tạo một file gọi là **newfile** (sử dụng **touch**, **cat**, hoặc **vi**)

Chuyển đến thư mục gốc (**cd /**). Xem nội dung của **newfile** từ vị trí này.

Câu lệnh ngắn nhất nào giúp bạn quay trở về **/tmp/bin**?

Câu lệnh ngắn nhất nào giúp bạn chuyển đến thư mục home của bạn?

Biến **PWD** là cục bộ hay toàn cục?

Tạo và xoá các thư mục

Cách nào là nhanh nhất để tạo các thư mục **/dir1/dir2**?

Xoá các thư mục này với **rmdir** sau đó với **rm**

Tạo khoảng trống trên hệ thống file

Để tạo thêm khoảng trống trên thiết bị chứa thư mục **/usr/share/doc** chúng ta cần tìm thiết bị dự phòng có đủ khoảng trống để copy nội dung của **/usr/share/doc** vào thiết bị này. Sau đó chúng ta sẽ xoá thư mục **/usr/share/doc** và tạo một điểm liên kết tượng trưng từ **/usr/share/doc** đến vị trí mới.

Tạo một thư mục **/spare** trên đó chúng ta sẽ gắn (mount) các thiết bị dự phòng phù hợp (một trong những phân vùng được tạo từ các bài tập trước sẽ phù hợp cho mục đích này)

```
mkdir /spare
mount <device> /spare
```

Kiểm tra với lệnh **df -h /spare** và **du -hs /usr/share/doc** xem thiết bị này có dung lượng đủ lớn để chứa tất cả dữ liệu đang có.

Tiếp theo, copy các nội dung của **/usr/share/doc** đến **/spare/**

```
cp -a /usr/share/doc /spare
```

Sau khi chắc chắn dữ liệu đã được copy hết thì thay đổi **/etc/fstab** có thể sử dụng ngay sau khởi động.

Xoá **/usr/share/doc** và tạo điểm liên kết tượng trưng từ **/usr/share/doc** đến **/spare/doc**

```
ln -s /spare/doc /usr/share/doc
```

Thực hiện tương tự với **/home**. Xem có vấn đề gì xảy ra?

Tìm kiếm các file trên hệ thống

Copy file **/etc/lilo.conf** đến **/etc/lilo.conf.bak**

1 Dùng lệnh **find** để tìm find mới

2 Dùng **locate** để tìm **/etc/lilo.conf.bak** (Bạn sẽ cập nhật cơ sở dữ liệu **slocate** như thế nào?)

Các sao lưu dự phòng (bước đầu tiên)

Tìm tất cả các file đã được thay đổi trong ngày hôm nay trong thư mục home của bạn.

```
find /home -mtime -1 |tee list1 |wc --lines (-1 có nghĩa là ít hơn 1 ngày)
```

Chúng ta sẽ giới thiệu các tiện ích lưu trữ ở phần sau, tuy nhiên kết quả xuất của các lệnh tìm kiếm sẽ được dẫn trực tiếp vào **cpio**.

QUẢN LÝ TIẾN TRÌNH

QUẢN LÝ TIẾN TRÌNH

Xem các tiến trình đang chạy

Các tiến trình có một mã (ID) tiến trình duy nhất đó là PID. Giá trị của PID có thể dùng để thay đổi sự ưu tiên của tiến trình hoặc dừng hẳn tiến trình đó.

Một tiến trình là bất cứ chương trình nào đang thực hiện. Nếu process_2 được sinh ra bởi process_1 thì nó được gọi là tiến trình con. Còn process_1 thì được gọi là tiến trình cha.

Cây gia hệ của các tiến trình

Lệnh **pstree** sẽ đưa ra một minh họa đầy đủ của hệ phân cấp các tiến trình cha và con.

Hình 1: Một phần các kết quả của pstree

```
bash(1046)--xinit(1085)-+X(1086)
  |-xfwm(1094)-+xfce(1100)--xterm(1111)--bash(1113)-+pstree(1180)
  |
  |   |-soffice.bin(1139)--soffice.bin(1152)-+
  |   |                                     -soffice.bin(1153)
  |   |                                     |-soffice.bin(1154)
  |   |                                     |-soffice.bin(1155)
  |   |                                     |-soffice.bin(1156)
  |   |                                     `--soffice.bin(1157)
  |   |
  |   |--xclock(1138)
  |
  |--xfgnome(1109)
  |--xfpager(1108)
  |--xfsound(1107)
  `--xscreensaver(1098)
```

QUẢN LÝ TIẾN TRÌNH

Trong hình trên tất cả các mã tiến trình (PID) đều được nhìn thấy; giá trị của chúng được tăng dần. Lựa chọn thông dụng nhất của lệnh này là `-p` sẽ hiển thị các PID và `-h` sẽ làm nổi rõ các tiến trình của người dùng.

Tìm kiếm các tiến trình đang thực hiện

Sử dụng lệnh `ps` là một cách trực tiếp để xác định tiến trình nào đang thực hiện. Phần lớn người dùng kết hợp một số các lựa chọn để phù hợp với mục đích tìm kiếm. Dưới đây là 3 lựa chọn như vậy.

- ps -ux** hiển thị tất cả các tiến trình thực hiện bởi người dùng
- ps T** hiển thị các tiến trình đang chạy bởi thiết bị đầu cuối hiện thời của người dùng
- ps aux** hiển thị tất cả các tiến trình trên hệ thống

Để biết chi tiết hơn các lựa chọn chúng ta nên sử dụng lệnh `ps manpage` và chọn ra những lựa chọn phù hợp nhất.

<i>ps accommodates UNIX-style and BSD-style arguments</i>
usage: ps -[Unix98 options] ps [BSD-style options] ps --[GNU-style long options] ps --help for a command summary

<i>Summary of options</i>
-a show all processes for the current user linked to a tty (<i>except the session leader</i>) -e or -A show all processes -f gives the PPID (Parent Process ID) and the STIME (Start Time) -l is similar to -f and displays a long list a show all processes linked to a tty, including other users x show all processes without a controlling tty as well

QUẢN LÝ TIẾN TRÌNH

Cập nhật liên tục thông tin tiến trình

Tiện ích **top** sẽ cập nhật thông tin trên các tiến trình tại một mức điều chỉnh.

Trong khi tiện ích **top** đang thực hiện chúng ta có thể gõ h đối với một danh sách các lệnh. Khoảng trống sẽ được cập nhật thông tin tức thời. Chúng ta cũng có thể dùng top để thay đổi mức độ ưu tiên của một tiến trình.

Thay đổi tiến trình

Dừng các tiến trình

Lệnh **kill** sẽ gửi các tín hiệu đến các tiến trình. Có tổng cộng 63 tín hiệu. Tín hiệu mặc định dừng một tiến trình được gọi là SIGTERM với giá trị 15.

kill

cú pháp

kill SIGNAL process_PID

Mọi tiến trình có thể lựa chọn nhận hay không nhận một tín hiệu ngoại trừ SIGKILL sẽ được thực hiện bằng nhân hệ thống. Các daemon sẽ hiểu SIGUP có nghĩa là "đọc lại file cấu hình"

Most Common Signals



- 1 or SIGHUP hangup or disconnect the process
- 2 or SIGINT same as Ctrl+C interrupt
- 3 or SIGQUIT quit
- 9 or SIGKILL kill the process through a kernel call
- 15 or SIGTERM terminate a process 'nicely'. This is the DEFAULT signal.

Chúng ta có thể sử dụng lệnh **killall** để dừng các tiến trình mà không cần biết PID

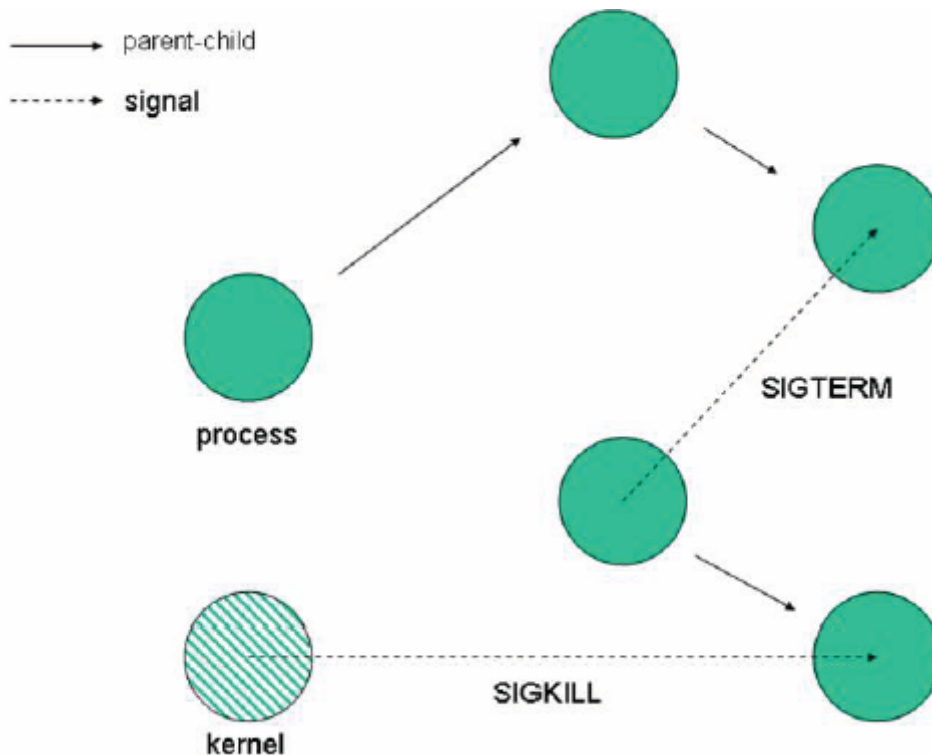
QUẢN LÝ TIẾN TRÌNH

killall

Cú pháp

```
killall SIGNAL process_NAME
```

Hình 1: Tín hiệu giữa các tiến trình

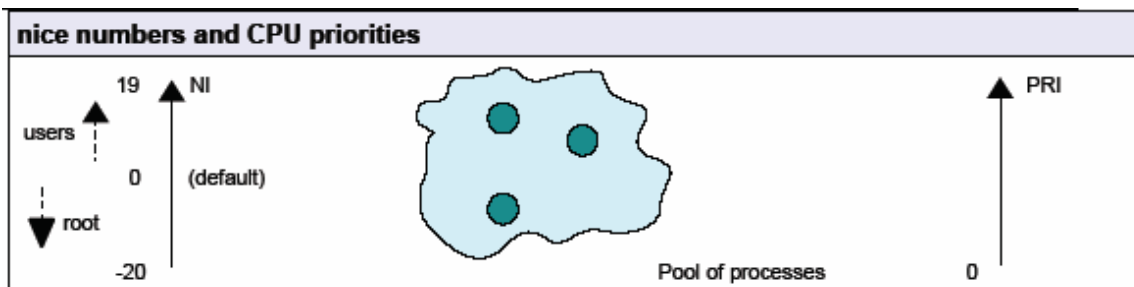


Ưu tiên tiến trình và các giá trị (nice value)

Các giá trị nice value (NI) thay đổi quyền ưu tiên của CPU và được dùng để cân bằng quá trình sử dụng CPU trong môi trường đa người dùng. Mỗi tiến trình bắt đầu với giá trị NI mặc định là 0. Các NI nằm trong phạm vi từ 19 [thấp nhất] đến -20 [cao nhất]

QUẢN LÝ TIẾN TRÌNH

Chỉ có người quản trị hệ thống có thể giảm giá trị NI của một tiến trình. Từ khi tất cả các tiến trình bắt đầu với giá trị NI mặc định là 0, chỉ có người quản trị hệ thống có thể thiết lập giá trị âm cho các giá trị NI.



Sử dụng lệnh **renice** để thay đổi mức độ ưu tiên của một tiến trình. Dùng lệnh **nice** để thiết lập mức độ ưu tiên của một tiến trình.

Cú pháp

```
Nice -<NI> <process>  
renice <+/-NI> -p <PID>
```

Chú ý rằng **renice** thực hiện với các PID và xử lý danh sách các tiến trình tại một thời điểm. Một lựa chọn có ích của **renice** là **-u**, lựa chọn này sẽ ảnh hưởng đến tất cả các tiến trình thực hiện bởi người dùng.

Thiết lập giá trị 1 cho các tiến trình 234 và 765



Thiết lập giá trị -5 cho xclock



```
nide --5 xclock
```

Tiến trình và Shell

Các tiến trình nền sau và nền trước

Sau khi chúng ta bắt đầu một tiến trình từ shell, chúng ta sẽ để tiến trình đó cho shell tự động thông dịch. Chúng ta chú ý rằng sẽ không có lệnh nào đáp ứng nữa. Lý do cho vấn đề này đó là chỉ có thể thực hiện các chương trình trong nền trước **fg** hoặc nền sau **bg** của shell.

Khi một chương trình đang chạy trong chế độ nền trước, dấu nhắc shell có thể khôi phục trong chốc lát bằng cách ngắt chương trình đó. Tín hiệu ngắt là **Ctrl Z**.

Dừng và bắt đầu các công việc (job)

Một tiến trình bắt đầu từ shell còn được gọi là một công việc. Khi một công việc nhận tín hiệu ^Z, nó sẽ được dừng và dấu nhắc shell sẽ xuất hiện. Để khởi tạo lại chương trình trong chế độ nền sau chúng ta chỉ cần gõ: bg

Ví dụ

[mike localhost /bin]\$xclock **xclock chạy trong chế độ nền trước, dấu nhắc shell biên mất**

[1]+ Stoppep xclock **xclock nhận tín hiệu ^Z**

QUẢN LÝ TIẾN TRÌNH

[mike localhost /bin]\$bg **dấu nhắc shell được khôi phục và đưa vào lệnh
bg**

[1]+ xclock & **xclock đang chạy trong chế độ nền sau**

[mike localhost /bin]\$

Chú ý ký hiệu [1]+ ở trên. Giá trị này là job number của tiến trình. Trong đó dấu hiệu '+' chỉ ra tiến trình được thay đổi lần gần nhất. Dấu hiệu '-' chỉ ra tiến trình được thay đổi lần liền kề

Hiển thị các công việc

Tiện ích jobs hiển thị tất cả các tiến trình đang chạy bắt đầu từ shell hiện thời. Giá trị job number, trạng thái công việc (đang chạy hay dừng), và 2 tiến trình được thay đổi gần nhất sẽ được hiển thị

<i>Output for jobs</i>	
[1]- Stopped	xclock
[2] Running	xman &
[3]+ Stopped	xload

Job number

Chúng ta có thể dừng và bắt đầu lựa chọn các công việc một cách thuận tiện bằng cách sử dụng job number. Việc lựa chọn này được thực hiện cùng với lệnh fg

Gọi job 2 ở nền trước và loại bỏ (kill) job 1

fg 2 hoặc kill -9 %1

fg %2 hoặc

fg %?xma

Tránh sử dụng HUP với nohup

QUẢN LÝ TIẾN TRÌNH

nohup là một chương trình có vai trò như một tiến trình cha độc lập với phiên người dùng. Khi một người dùng thoát khỏi hệ thống, thì hệ thống sẽ gửi HUP đến tất cả các tiến trình nằm trong nhóm tiến trình của người dùng. Ví dụ, để tránh tín hiệu HUP, một chương trình gọi là bigbang sẽ cố gắng tính thời gian xuất hiện của các tiến trình



```
nohup bigbang &
```

QUẢN LÝ TIẾN TRÌNH

Thực hành

Bạn nên chạy X trước khi bắt đầu các bài thực hành sau

1. Kiểm tra giá trị nice value (NI) hiện thời của x-terminal đang chạy. Thay đổi giá trị này bằng lệnh **top** hoặc **renice**
2. Tín hiệu tương đương của **^Z** gửi đến một tiến trình là gì? (Hiển thị tất cả các tín hiệu với **kill -l**)
3. Tín hiệu nào được định nghĩa lại cho phần lớn các daemon và yêu cầu đọc lại file cấu hình?
4. Tín hiệu mặc định gửi đến một tiến trình là gì khi sử dụng **kill** hoặc **killall**?
5. Tín hiệu nào được trực tiếp xử lý bằng nhân hệ thống (kernel) và không thể định nghĩa lại?
6. Trước hết bạn hãy đăng nhập vào thiết bị đầu cuối ảo (tty1 to tty6). Chúng ta sẽ thực hiện một script cho phép tiếp tục chạy khi chúng ta thoát ra khỏi hệ thống dùng tiến trình cha nohup

Trong thư mục **/tmp** tạo một file gọi là **print-out** với nội dung sau đây

```
#!/bin/bash
count=0
while (true) do
    echo this is iteration number $count
    let count+=1
done
```

Chúng ta trước hết thực hiện các bước sau (không dùng **nohup**)

```
cd /tmp
./print-out &
exit
```

QUẢN LÝ TIẾN TRÌNH

Chúng ta có thể không nhìn thấy dòng lệnh khi gõ exit nhưng câu lệnh này sẽ làm bạn thoát ra khỏi hệ thống. Khi bạn đăng nhập lại hãy kiểm tra print-out đã được dừng

```
ps ux | grep print-out
```

Tiếp theo bắt đầu với lệnh

```
nohup /tmp/print-out &  
exit
```

Đăng nhập lại và kiểm tra những lệnh sau

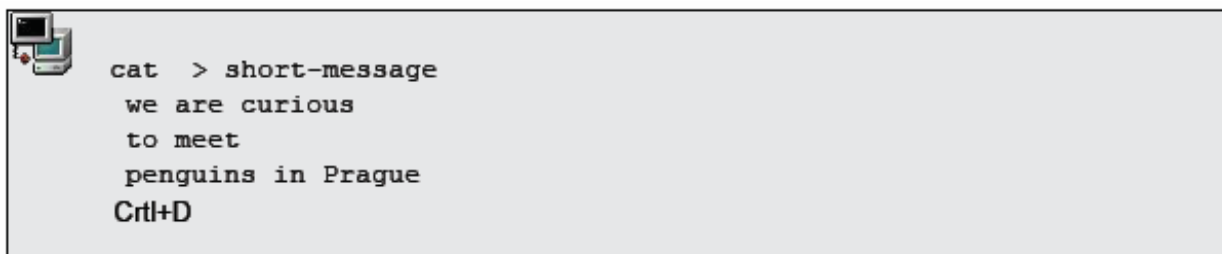
```
ps ux |grep print-out  
tail -f ~/nohup.out  
Ctrl+C  
killall print-out  
ps ux|grep print-out  
tail -f ~/nohup.out
```

XỬ LÝ VĂN BẢN

cat the Swiss Army Knife

Dùng cat để soạn văn bản

Tiện ích cat có thể dùng như một chương trình soạn thảo đơn giản



```
cat > short-message
we are curious
to meet
penguins in Prague
Ctrl+D
```

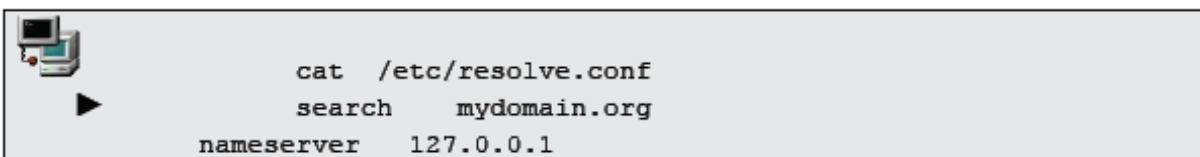
Chú ý cách dùng Ctrl+D. Lệnh này được dùng để kết thúc nhập input.

Dùng cat để đọc văn bản

Thông thường hơn **cat** được dùng để đưa văn bản ra *stdout*. Các lựa chọn thường được dùng là

- n đánh số mỗi dòng của output
- b chỉ đánh số dòng output không trống
- A hiển thị ký hiệu xuống dòng

Ví dụ



```
cat /etc/resolv.conf
search mydomain.org
nameserver 127.0.0.1
```

Lệnh tac sẽ đọc văn bản từ cuối lên đầu

Lệnh này giống như cat ngoại trừ nội dung văn bản được đọc từ dòng cuối lên đầu



```
tac short-message
▶ penguins in Prague
  to meet
  we are curious
```

Các công cụ đơn giản

Sử dụng head hoặc tail


Các tiện ích head hoặc tail thường được dùng để phân tích các logfile. Chúng sẽ xuất đưa ra mặc định 10 dòng văn bản. Sau đây là cách dùng

Hiển thị 20 dòng đầu tiên của **/var/log/messages**:



```
head -n 20 /var/log/messages
head -20 /var/log/messages
```

Hiển thị 20 dòng cuối cùng của **/etc/aliases**:



```
tail -20 /etc/aliases
```

Tiện ích tail có thêm một lựa chọn cho phép hiển thị nội dung văn bản bắt đầu từ dòng đưa vào cho đến hết.

Hiển thị nội dung văn bản bắt đầu từ dòng 25 trong **/var/log/messages**:



```
tail +25 /etc/log/messages
```

Câu hỏi: nếu một văn bản có 90 dòng, chúng ta sẽ sử dụng lệnh **tail** và **head** như thế nào để hiển thị các dòng từ 50 tới 65? Có thể có nhiều hơn một cách để thực hiện điều này?

Cuối cùng tail có thể đọc liên tục một file bằng lựa chọn -f. Lựa chọn này rất có ích khi chúng ta mong muốn một file được thay đổi trong thời gian thực

Đếm số dòng, số từ và byte

Tiện ích **wc** sẽ đếm số lượng các byte, các từ, và các dòng trong file. Một vài lựa chọn cho phép chúng ta thay đổi giá trị output của **wc**

Các lựa chọn cho **wc**

-l	Đếm số dòng
-w	Đếm số các ký tự hoặc từ
-c hoặc -m	Đếm số các byte hoặc ký tự


Lưu ý:

Nếu không có tham biến, **wc** sẽ đếm dựa trên nội dung được gõ vào *stdin*

Đánh số các dòng


Tiện ích **nl** có tác dụng giống như **cat -b**

Đánh số tất cả các dòng gồm cả các dòng trống



```
nl -ba /etc/lilo.conf
```

Đánh số các dòng văn bản không trống



```
nl -bt /etc/lilo.conf
```

Thay thế tab bằng space

Lệnh **expand** cho phép thay thế TAB bằng các dấu cách (space). Chúng ta có thể dùng lệnh **unexpand** để thay thế ngược lại.

Xem các file nhị phân

Có một số công cụ để thực hiện điều này. Công cụ phổ biến nhất là **od** (octal dump) và **hexdump**.

Xử lý văn bản

Các công cụ sau đây thay đổi bố trí văn bản

Lựa chọn các trường (field) và các ký tự với cut

Tiện ích cut có thể lấy ra một vùng các ký tự hoặc các trường từ mỗi dòng của văn bản.

Lựa chọn -c được dùng để xử lý ký tự.

Cú pháp:

```
cut -c {range1,range2}
```


Ví dụ:



```
cut -c5-10,15- /etc/passwd
```


Ví dụ trên sẽ đưa ra các ký tự từ vị trí 5 đến 10 và từ 15 đến cuối dòng của mỗi dòng trong `/etc/passwd`

Chúng ta có thể xác định dấu phân cách các trường (dấu cách, dấu phẩy,...) của một file cũng như các trường đối với output. Các lựa chọn được thiết lập với cờ hiệu `-d` và `-f` tương ứng

Cú pháp:

```
cut -d {delimiter} -f {fields}
```

Ví dụ:



```
cut -d: -f 1,7 --output-delimiter=" " /etc/passwd
```

Ví dụ này sẽ đưa ra các trường từ trường đầu tiên đến trường thứ bảy của `/etc/passwd` được phân cách bằng dấu cách. Mặc định của *output-delimiter* là giống như các phân cách input ban đầu. Lựa chọn **`--output-delimiter`** cho phép chúng ta thay đổi giá trị mặc định này.

Kết nối và dán văn bản

Tiện ích đơn giản nhất là **paste** sẽ ghép hai file bên cạnh nhau

Cú pháp

```
paste text1 text2
```

Với tiện ích **join** chúng ta có thể xác định cụ thể hơn những trường mà chúng ta đang quan tâm

Cú pháp

```
join -j1 {field_num} -j2{field_num} text1 text2 or
join -1 {field num} -2{field num} text1 text2
```

Văn bản được gửi đến stdout chỉ khi những trường cụ thể đối sánh. Quá trình so sánh được thực hiện trên một dòng tại mỗi thời điểm cho đến khi có sự trùng nhau được tìm thấy và quá trình sẽ được dừng ngay lập tức mặc dù có thể có những sự trùng nhau nằm ở phía sau.

Sắp xếp output

Lệnh sort sẽ sắp xếp mặc định một văn bản theo thứ tự abc. Lựa chọn -n sẽ thực hiện việc sắp xếp theo thứ tự số.

Định dạng output

Chúng ta có thể thay đổi số lượng các ký tự trong mỗi dòng của output bằng lệnh fmt. Mặc định fmt sẽ liên kết các dòng và đưa ra 75 ký tự cho mỗi dòng

Các lựa chọn *fmt*

- w số lượng các ký tự trên mỗi dòng
- s tách những dòng dài nhưng không điền đầy dòng còn lại
- u đặt một dấu cách ở giữa mỗi từ và 2 dấu cách ở cuối mỗi câu

Thay thế các ký tự

Tiện ích **tr** sẽ thay thế một tập hợp các ký tự bằng tập hợp ký tự khác.

Ví dụ thay đổi các chữ cái viết hoa bằng chữ cái thường

```
tr '[AB]' '[ab]' <file.txt
```

Thay thế các dấu phân cách trong **/etc/passwd**:



```
tr ':' ' ' < /etc/passwd
```

Chú ý: **tr** chỉ có 2 tham biến! Và file không được tính là tham biến.

Thực hành

1. Sử dụng **cat** để gõ văn bản sau vào một file có tên là *message*

```
cat >> message
line 1
^D
```

Thực hiện tương tự nhưng dùng từ khoá **STOP** thay thế điều khiển kết thúc file (^D)

```
cat >> message << STOP
line 2
STOP
```

Tiếp theo, thêm văn bản sau vào *message* sử dụng **echo**

```
echo line 3 >> message
```

2. Tạo một file có tên là *index* với 2 trường *REFERENCE* và *TITLE* được phân cách bằng dấu cách

```
ví dụ 001 Using_Linux
```

Tạo file thứ hai có tên là *pricing* với 2 trường *REFERENCE* và *PRICE* được phân cách bằng dấu cách

```
ví dụ 001 9.99
```

Sử dụng lệnh **join** để hiển thị các trường *reference*, *title*, và *price*.

3. Sử dụng **tr** để thay thế toàn bộ dấu hai chấm bằng dấu chấm phẩy trong */etc/passwd*

Thực hiện tương tự dùng lệnh **cut**

- Sử dụng lệnh **head** và **tail** để hiển thị dòng 75 đến 80 của */var/log/messages*
- Sử dụng tiện ích **cut** cùng với **grep** và **ifconfig** để in ra duy nhất địa chỉ IP của giao diện mạng eth0
- Trong **/tmp** tạo một thư mục với tên là files

```
mkdir /tmp/files
```

Tạo 50 file trong thư mục này

```
# ! /bin/bash
count=0
while [ $count -lt 50 ] do
touch /tmp/files/$count.txt
let count+=1
done
```

Các bạn sẽ gõ các dòng lệnh sau để thay đổi các file có phần mở rộng **txt** sang **dat**

```
-----
for FILES in $(ls *.txt)
do
FILENAME=$(echo $FILES | cut -d. -f1)
mv $FILES $FILENAME.dat
done
```

CÀI ĐẶT PHẦN MỀM

CÀI ĐẶT PHẦN MỀM

Giới thiệu

Hãy bắt đầu cùng với một đoạn mã nguồn ngắn. Ví dụ này sẽ giúp giúp chúng ta tìm hiểu vấn đề mà không cần có kiến thức sâu về ngôn ngữ lập trình C

Tệp main.c:

```
#include<stdlib.h>
int main(){
Hello();
}
```

Tệp Hello.c:

```
#include<stdio.h>
void Hello(){
printf("Hi ! \n");
}
```

Chú ý: main.c là chưa hoàn thành nếu hàm Hello() là chưa được định nghĩa. Cũng như vậy đối với Hello.c nếu không có khái báo main. Do đó, các tệp này là phụ thuộc nhau. Tuy nhiên từng hàm riêng biệt vẫn có thể được dịch theo một đối tượng kiểu Object (.o) đây là kiểu files được sử dụng để xây dựng các ứng dụng.

Dịch các file đối tượng

```
gcc -c main.c
gcc -c Hello.c
```

CÀI ĐẶT PHẦN MỀM

Câu lệnh trên sẽ tạo ra 2 tệp main.o và Hello.o được sử dụng để xây dựng các ứng dụng **app**.

Dịch app

Lựa chọn `-o` xác định tên của mã nguồn đã được dịch. Nếu không có tên tệp nào được chỉ ra tệp dịch sẽ được đặt tên mặc định **a.out**

Tất cả các bước trên có thể được chạy tự động bằng cách sử dụng a Makefile. Dưới đây là một ví dụ nhỏ dùng Makefile để tạo ứng dụng **app**

Makefile

```
SHELL = /bin/sh
CC = /usr/bin/gcc
app: main.o Hello.o
$(CC) -o app main.o Hello.o
main.o: main.c
$(CC) -c main.c
Hello.o Hello.c
$(CC) -c Hello.c
```

Thư viện tĩnh và thư viện chia sẻ

Các hàm chức năng thường dùng được lưu lại trong các thư viện. Trong thời gian dịch chương trình các thư viện này có thể được link tới mã nguồn nơi có sử dụng lệnh gọi thư viện chức năng. Thư viện có thể được link tới mã nguồn một cách tĩnh hoặc động.

Lệnh dịch gcc có thể link thư viện trong các cách khác nhau. Tuy nhiên theo chế độ mặc định nó sẽ là link các tệp(files) được khai báo trong dòng lệnh không có phần mở rộng .c (chỉ có các tệp có mở rộng .c là được hiểu như mã nguồn).

Listing 1. Kết nối mặc định (linking)

CÀI ĐẶT PHẦN MỀM



```
gcc main.c Hello.o
```

Dòng lệnh trên sẽ tạo tệp chạy a.out cùng tệp đối tượng Hello.o được link tĩnh tới nó.

- **Thư viện tĩnh**

Các thư viện tĩnh được lưu trữ trong file **.o**. Các lưu trữ được tạo ra bởi công cụ **ar** và có phần mở rộng **.a**.

Hình 2: thêm một file đối tượng vào phần lưu trữ



```
ar rcs libfoo.a file1.o file2.o
```

- **Thư viện động / Thư viện chia sẻ**

Thư viện chia sẻ là một thư viện sẽ được tải bởi chương trình khi nó được thực thi. Mặt khác chúng ta cũng có thể nói nó là *thư viện mà được tải động* (dynamically loaded)

Hình 3: Tạo thư viện chia sẻ:

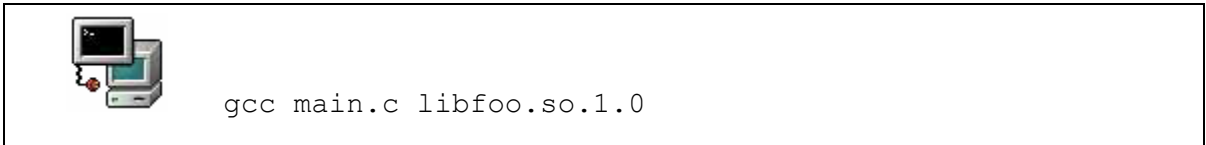


```
gcc -c -fPIC Hello.c      tạo file đối tượng  
gcc -shared -Wl, soname, libfoo.so.1 -o libfoo.so.1.0  
Hello.o
```

Cờ hiệu (flag) `-fPIC` sẽ kích hoạt vị trí mã nguồn độc lập

CÀI ĐẶT PHẦN MỀM

Hình 4: Dịch chương trình với thư viện chia sẻ:



Dòng lệnh trên sẽ tạo file chạy **a.out**. Tuy nhiên nếu bạn thử chạy file này máy tính sẽ thông báo lỗi dưới đây.

Thông báo lỗi không tìm thấy thư viện chia sẻ

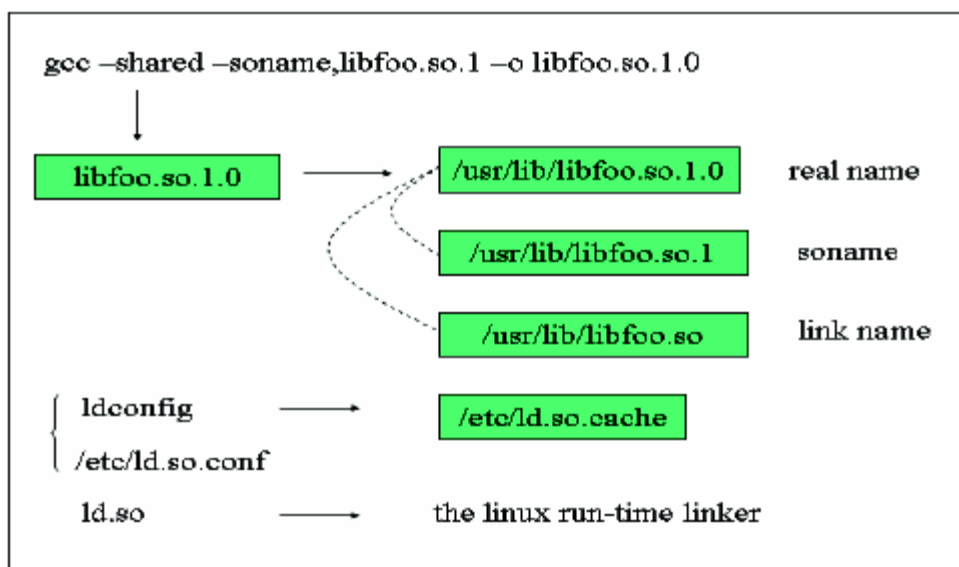
```
./a.out: error while loading shared libraries:  
libfoo.so.1.0:cannot open shared object file: No such file or  
directory
```

Trong phần tiếp theo chúng ta sẽ tìm hiểu cách để sửa lỗi này.

- **Đặt tên Thư viện chia sẻ và tải dynamic**

Chúng ta sử dụng ví dụ trong phần trước để tìm hiểu Thư viện Linux được bảo trì(maintain) thế nào.


CÀI ĐẶT PHẦN MỀM



Hình 1: Tên thư viện chia sẻ

Sử dụng công cụ `ldd` để xem Thư viện chia sẻ nào một file chạy cần trong thời gian thực thi.

Ví dụ:



```
ldd a.out
libfoo.so.1.0 => not found
libc.so.6 => /lib/libc.so.6 (0x40028000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

Chú ý chúng ta sẽ không tìm thấy file `libfoo.so.1.0` vì **a.out** cần tải(load) động thư viện này và kết nối động **ld.so** không biết có sự tồn tại của thư viện này.

Có thể làm theo một trong các cách sau để khắc phục lỗi này.

CÀI ĐẶT PHẦN MỀM

1. Nếu tệp nhị phân cần ở chế độ tạm thời, kiểm tra định nghĩa biến LD_LIBRARY_PATH như sau:



```
export LD_LIBRARY_PATH =$(pwd)
```

2. Copy libfoo.so.1.0 vào thư mục /usr/lib và chạy ldconfig để nâng cấp ld cache

Cài đặt nguồn

CÀI ĐẶT PHẦN MỀM

Quản lý gói Redhat (Redhat Package Manager RPM)

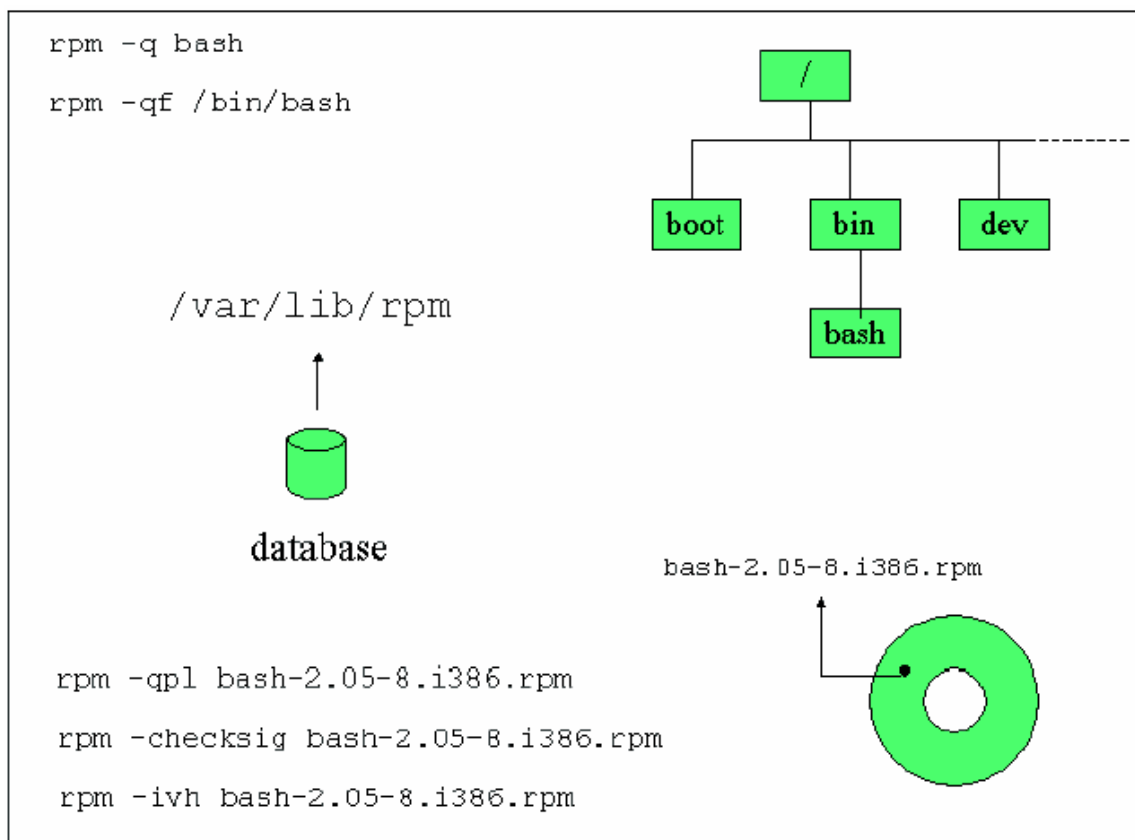


Figure1: Các chức năng của Quản lý Gói (Package Manager)

Đặt tên package

Rpm được đặt tên theo cách sau

name-version-release.architecture.rpm

Chế độ(mode) chính

Tắt	Đầy đủ	Mô tả
-i	- install	Cài đặt gói

CÀI ĐẶT PHẦN MỀM

-U	<code>-update</code>	Cập nhật hoặc cài đặt gói
-F	<code>--freshen</code>	Cập nhật chỉ những gói đã install
-V	<code>--verify</code>	Cỡ file, MD5, quyền, kiểu ...
-q	<code>--query</code>	Yêu cầu gọi các gói và các file đã cài đặt/đã gỡ bỏ
-e	<code>--erase</code>	Gỡ bỏ gói

Chế độ thứ cấp

Tắt	Mô tả
a	áp dụng cho tất cả các gói đã cài đặt
c	cùng với q đưa ra các file cấu hình
d	cùng với q đưa ra các file tài liệu
h	chạy bảng băm (hash) trong khi xử lý
i	cùng với q đưa ra thông tin về gói
l	cùng với q đưa ra tất cả file và thư mục trong một gói
p	cùng với q chỉ ra truy vấn nào được thực hiện đối với file
v	verbose

Các mode yêu cầu (query mode)

Chúng ta xem xét ví dụ với gói `routed-0.17.i386.rpm`. Bạn có thể truy vấn gói này và đưa ra nội dung của nó trước khi cài đặt cùng với lựa chọn `l` như sau:

CÀI ĐẶT PHẦN MỀM



```
rpm -qpl routed-0.17.i386.rpm
```

Khi gói này được cài đặt bạn có thể truy vấn gói đã cài đặt như sau:



```
rpm -ql routed-0.17      or  
rpm -ql routed
```

Cuối cùng nếu chúng ta muốn tìm gói nào đã cài đặt file /usr/sbin/routed dữ liệu rpm có thể được yêu cầu cùng:



```
rpm -qf /usr/sbin/routed
```

Ba kiểu truy vấn (query): uninstalled packages, installed packages và file

Kiểu truy vấn	Tùy chọn
Package File	-qp
Installed Package	-q
File	-qf

Một tùy chọn mở rộng sẽ cho phép bạn lấy thông tin trong tất cả các files đã cài đặt – **l**, tài liệu đã cài đặt – **d**, file cấu hình – **c**, v.v...

Các Tùy chọn đặc biệt

CÀI ĐẶT PHẦN MỀM

- nodeps** cho phép cài đặt không phụ thuộc
- force** ép buộc nâng cấp
- test** không cài đặt hoặc nâng cấp, chỉ in ra stdout
- requires** chỉ ra các yêu cầu của gói

Mã nguồn cho rất nhiều packages RPM cũng có thể được để dưới dạng package RPM và sẽ được sử dụng để xây dựng một package nhị phân. Tên kết hợp sẽ là:

`name-version-release.src.rpm` (tên-phiên_bản-ngày_xuất-bản.src.rpm)

Các gói như vậy sẽ chứa ít nhất 2 file, tarball cùng mã nguồn và một spec file. spec file chứa đựng chỉ dẫn để vá (patch), dịch và xây dựng RPM package. Nếu mã nguồn cần được vá trước khi dịch thì miếng vá sẽ nằm trong package nguồn.

Có ba cách khác nhau để xây dựng một package RPM. Giả sử rằng bạn có một package với tên gọi: `name-version-release.src.rpm`.

*Đối với các phương thức này, trước tiên bạn cần cài đặt gói **rpm-build***

Cách 1:

Cài đặt package nguồn RPM với:

```
rpm -ivh name-version-release.src.rpm
```

Lệnh trên sẽ copy các file vào thư mục sau:

`/usr/src/redhat/SPECS`

`/usr/src/redhat/SOURCES`

CÀI ĐẶT PHẦN MỀM

Trong thư mục `/usr/src/redhat/SPECS` có một file với tên **name.spec** (trong đó ‘name’ là tên của package). Để bắt đầu xây dựng package dịch, tên **name-version-release.i386.rpm**, gõ trong cửa sổ lệnh:

```
rpm -ba name.spec
```

Dòng lệnh trên sẽ kích hoạt một loạt các scripts. tarball trong `/usr/src/redhat/SOURCES` sẽ được mở (unpackage) tại `/usr/src/redhat/BUILD`

Nếu quá trình dịch thành công thì package nhị phân sẽ được lưu trong `/usr/src/redhat/RPMS/`.

Có một số các thư mục thứ cấp khác nhau tương ứng với một số models/thế hệ của CPU. Nếu quá trình dịch không liên đới tới các đặc tính đặc biệt từ các chip thì package đó sẽ được lưu vào thư mục `noarch`.

Cách 2:

Cách này cũng tương tự như cách thứ 1 nhưng bắt đầu với lệnh đơn sau đây:

```
rpm --rebuild name-version-release.src.rpm
```

Cách 3:

Trong một vài trường hợp nhà phát triển sẽ phân phối tarball cùng với nhau trong một file spec. Nếu tarball được gọi tên `name-version-release.tar.gz` bạn có thể tìm một file `.spec` với lệnh sau:

```
tar tzvf name-version-release.tar.gz | grep .spec
```

Nếu tarball có một file spec thì bạn có thể xây dựng một package RPM bằng cách gõ:

CÀI ĐẶT PHẦN MỀM

```
rpm --bt name-version-release.tar.gz
```

Công cụ Alien

Công cụ này sẽ chuyển đổi packages Debian sang Redhat và ngược lại. Bạn có thể tải xuống tại: <http://kitenet.net/programs/>

CÀI ĐẶT PHẦN MỀM

Thực hành

Trong các ví dụ sau tải một file RPM nguồn (vd. bash-2.05-8.src.rpm với Redhat 7.2) từ www.rpmfind.net

1. Cài đặt tarball

Bung các thành phần của gói RPM mà không dịch bất cứ file nào:

```
rpm -ivh bash-2.05-8.src.rpm
```

Trong thư mục `/usr/src/redhat/SOURCES`, mở gói tarball với:

```
tar xvzf bash-2.05-8.tar.gz
```

Tùy chọn(khuyến nghị): Miếng vá có thể được áp dụng. Cú pháp sẽ thay đổi phụ thuộc vào bạn đang ở thư mục nào

Từ `/usr/src/redhat/SOURCES`:

```
patch -p0 -b <file.patch
```

Từ `/usr/src/redhat/SOURCES/bash-2.05-8`

```
patch -p1 -b <file.patch
```

Cuối cùng dùng:

```
./configure
```

```
make
```

Nếu bạn chắc chắn bạn muốn cài đặt package này hãy dùng `make install` nhưng nhớ rằng nó sẽ không cài đặt phần mềm sử dụng package manager.

2. Xây dựng lại RPM package manager.

```
rpm -rebuild package.src.rpm
```

Package nhị phân đã dịch sẽ ở trong `/usr/src/redhat/RPMS`

- Kiểm tra thành phần của package với tùy chọn `-qpl`
- cài đặt package, và chạy truy xuất với các package cài đặt

CÀI ĐẶT PHẦN MỀM

- Gỡ bỏ(uninstall) package

THAO TÁC VỚI VĂN BẢN NÂNG CAO

Tìm kiếm một từ hoặc một cụm từ trong một văn bản được lưu trữ sử dụng **grep**, **fgrep** hoặc **egrep**. Các từ khoá sử dụng trong quá trình tìm kiếm là một tổ hợp của các ký tự được gọi là biểu thức chính quy (regular expressions-regex). Biểu thức chính quy được nhận dạng bởi rất nhiều ứng dụng như **sed**, và **vi**.

Các biểu thức chính qui

Bảng 1. Danh sách regex chính

Ký tự	Tìm kiếm tương ứng
x (hoặc bất cứ ký tự nào)	Cacs chuỗi chứa đựng ‘x’
\<KEY	Các từ bắt đầu bằng ‘KEY’
WORD\>	Các từ kết thúc bằng ‘WORD’
^	Bắt đầu của một dòng
\$	Kết thúc một dòng
[Range]	Giới hạn của bảng mã ASCII
[^c]	Không phải ký tự ‘c’
\[Dịch ký tự ‘[’ theo gốc
“cat*”	Chuỗi chứa đựng ‘ca’ hoặc ‘cat’ và các ký tự bất kỳ tiếp theo
“.”	Tìm kiếm các ký tự đơn

THAO TÁC VỚI VĂN BẢN NÂNG CAO

Biểu thức chính quy mở rộng (extended regex- eregex): Các ký tự chính của eregex là: . ?,() và |

Bảng2: Danh sách eregex chính

Ký tự	Tìm kiếm tương ứng
“A1 A2 A3”	Chuỗi chứa đựng ‘A1’ hoặc ‘A2’ hoặc ‘A3’
“cat+”	Chuỗi chứa đựng ít nhất cat và các ký tự bất kỳ tiếp theo
“cat?”	Chuỗi chứa đựng ‘ca’ hoặc ‘cat’ và các ký tự bất kỳ tiếp theo.

Họ grep

Tính năng grep hỗ trợ biểu thức chính quy regex như đã mô tả ở bảng1.

egrep

Công cụ egrep hỗ trợ biểu thức chính quy mở rộng eregex như mô tả trong bảng2.

fgrep

fgrep biểu diễn cho grep nhanh và **fgrep** dịch chuỗi gốc (không có hỗ trợ của regex hoặc eregex)

Làm việc với grep

Cú pháp của grep:

grep PATTERN FILE

Grep	Main Options
-c	Đếm số lượng dòng trùng với PATTERN
-f	Tim PATTERN từ file
-i	bỏ qua các trường hợp nhạy cảm

THAO TÁC VỚI VĂN BẢN NÂNG CAO

-n	chỉ ra số dòng của file
-v	xuất ra tất cả các dòng từ những dòng chứa PATTERN
-w	Tìm kiếm chính xác tuyệt đối PATTERN


Ví dụ đưa ra danh sách của tất cả các dòng không trống trong /etc/lilo.conf:



```
grep -v "^$" /etc/lilo.conf
```

egrep và fgrep

Tiện ích fgrep không nhận biết được ngữ nghĩa đặc biệt của một biểu thức chính quy. Ví dụ



```
fgrep "cat*" FILE
```

Dòng lệnh trên chỉ tìm kiếm các từ chứa đựng 'cat'. Khả năng của fgrep được bổ sung thêm nhờ lựa chọn LIST. Cú pháp như sau :

```
fgrep -f LIST FILE
```

Tiện ích **egrep** sẽ thực hiện với mọi biểu thức chính quy mới. Nó cũng có thể tìm kiếm một vài từ khoá nếu chúng được bắt đầu với dòng lệnh được chia bởi pipes. Ví dụ:



```
egrep "linux|^image" /etc/lilo.conf
```

Bộ soạn thảo Stream – sed

Tiện ích sed thông thường được sử dụng để tìm kiếm và thay đổi pattern trong văn bản. Nó hỗ trợ phần lớn các biểu thức chính quy (regex).

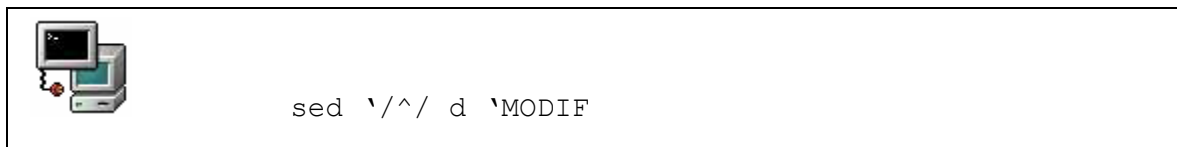
Làm quen với sed

Cú pháp :

```
sed [option] 'lệnh' [INPUTFILE]
```

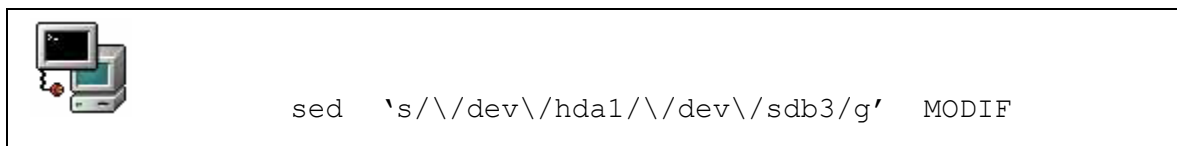
file input là tùy ý vì **sed** cũng làm việc trong các *thư mục file* và *pipes*. Đây là một vài ví dụ giả sử chúng ta làm việc trong một file gọi là MODIF.

Xoá tất cả các dòng chú thích :



Chú ý rằng pattern được tìm kiếm nằm giữa hai gách chéo //.

Thay thế /dev/hda1 bởi /dev/sdb3:



Ký tự **s** trong dòng lệnh biểu diễn cho ‘substitute’. Ký tự ‘**g**’ biểu diễn cho ‘globally’ và ép substitution trên mọi dòng

Nếu dòng chứa đựng từ khoá KEY thì thay thế ‘:’ với ‘;’

THAO TÁC VỚI VĂN BẢN NÂNG CAO



```
sed '/KEY/ s/:;/g' MODIF
```

sed nâng cao

Bạn có thể sử dụng một vài lệnh bắt đầu với **-e** tại dòng lệnh. Ví dụ, (1) xoá tất cả dấu trống khi (2) thay thế 'OLD' bằng 'NEW' trong file MODIF



```
sed -e '/^$/ d' -e 's/OLD/NEW/g' MODIF
```

Các lệnh trên có thể được viết vào một file, ví dụ COMMANDS. Khi đó mỗi dòng được dịch như một dòng lệnh để chạy.



```
sed -e '/^$/ d' -e 's/OLD/NEW/g' MODIF
```

```
1 s/old/new
```

```
/keyword/ s/old/new/g
```

```
23,25 d
```

Cú pháp sử dụng cùng COMMANDS file là:

```
sed -f COMMANDS MODIF
```

Việc này tiện lợi hơn rất nhiều việc phải đánh liên tục những dòng lệnh dài.

Tóm tắt lựa chọn cho sed

Cờ dòng lệnh

THAO TÁC VỚI VĂN BẢN NÂNG CAO

-e Thực hiện các lệnh tiếp sau đó
-f Đọc các lệnh từ một file
-n Không in ra các dòng không được sửa đổi

Tùy chọn của lệnh
d Xoá một dòng
r Đọc một file và xuất ra file output
s Thay thế
w Ghi kết quả ra vào một file

Thực hành

1. Tạo một file mới có tên FILE với nội dung sau:

```
Using grep,  
fgrep and  
egrep  
to grep for 99% of the cats  
% these are two  
% commented lines
```

Sử dụng grep để xuất ra chỉ những dòng lệnh không phải là dòng chú thích

Tìm kiếm các dòng chứa đựng các từ bắt đầu với 'a'

2. Biểu thức chính quy. Thêm các dòng sau vào file trên:

```
ca  
cat  
cats  
catss  
cat+  
cat*  
cat?  
car  
carriage
```

Xem kết quả của các lệnh sau khi sử dụng grep, egrep và fgrep:

```
grep 'cat+' FILE  
grep 'cat?' FILE  
grep 'cat.' FILE  
grep 'cat*' FILE
```

3. Sử dụng **sed** để thực hiện các thay đổi sau trong FILE

(sử dụng file COMMAND, sau đó làm các bước sau trên dòng lệnh)

- trong dòng đầu thay thế ‘grep’, với ‘soap’
- xoá ‘fgrep’ trong dòng thứ hai
- thay thế ‘egrep’ với ‘water’
- trong dòng thứ tư thay thế ‘grep for’ với ‘wash’

Save kết quả vào một file sử dụng tùy chọn **w**

SỬ DỤNG TRÌNH SOẠN THẢO VI

Dường như vi được sử dụng là trình soạn thảo chính trong Linux. Nó được coi như là một công cụ hữu ích như grep hoặc cat và được tổ chức tại thư mục /bin

Các chế độ Vi

Để thực hiện các thao tác phức tạp như là copy/paste, trình soạn thảo vi có thể thực hiện bằng nhiều chế độ khác nhau

- Chế độ dòng lệnh (Command Mode)

Đây là chế độ soạn thảo và đánh dấu thường sử dụng một chữ cái. Ví dụ dùng chữ cái j để nhảy xuống dòng tiếp theo

Như là quy tắc ngón tay cái (rule of thumb), nếu bạn muốn thực hiện một thao tác nhiều lần, bạn có thể điền số lần thực hiện trước khi gõ câu lệnh. Ví dụ: dùng lệnh 10j để nhảy đến 10 dòng tiếp theo.

- Chế độ dòng (hoặc cột) cuối cùng

Bạn có thể sử dụng chế độ này ở màn hình dòng lệnh (command line mode) bằng cách đánh dấu hai chấm. Cột sẽ hiển thị ở góc bên trái cuối cùng của màn hình. Trong chế độ này, bạn có thể thực hiện các thao tác đơn giản như tìm kiếm, ghi dữ liệu, thoát hoặc chạy một câu lệnh shell.

- Chế độ chèn

Cách đơn giản nhất để thực hiện chế độ này trong màn hình dòng lệnh (command Mode) là dùng chữ cái i hoặc a. Đây là chế độ trực quan nhất và thường được sử dụng để chèn văn bản vào một tài liệu.

Phím Esc sẽ thoát chế độ chèn và quay trở về màn hình dòng lệnh

Các mục văn bản

Các mục văn bản như là từ (words) hoặc đoạn văn bản (paragraph) được định nghĩa trong chế độ dòng lệnh (command mode) cho phép soạn thảo các lệnh sử dụng trong các tài liệu văn bản mà không cần dòng đến thiết bị chuột.

Từ, câu và đoạn (Words, sentences and paragraphs)

e reps. b	Chuyển đến cuối / đầu từ hiện thời
(reps.)	Chuyển đến cuối / đầu câu hiện thời
{ reps. }	Chuyển đến cuối / đầu đoạn hiện thời
w	trung tự như e nhưng thêm một dấu cách sau từ hiện thời

Đầu và cuối (Beginning and End)

^	Đầu dòng
\$	Cuối dòng
1G	Đầu tệp
G	Cuối tệp

Tất cả các mục văn bản trên có thể được sử dụng để đánh dấu một chữ (**w**) hoặc một đoạn văn bản (**{**) một lần, di chuyển đến đầu dòng (**^**) hoặc đầu tệp (**G**), vv... cũng như được sử dụng để thực hiện các câu lệnh như xoá hoặc copy.

Chèn văn bản

Trong chế độ dòng lệnh, **i** cho phép bạn chèn thêm văn bản vào tài liệu. Các đặc tính khác của trình soạn thảo **vi** cũng được thực hiện tương tự như vậy. Bảng sau đây sẽ liệt kê toàn bộ các đặc tính chèn văn bản của **vi**.

Các câu lệnh chèn

a	Chèn văn bản với con trỏ tại ký tự cuối cùng của dòng
----------	-------------------------------------------------------

SỬ DỤNG TRÌNH SOẠN THẢO VI

A	Chèn văn bản với con trỏ tại ký tự cuối cùng ở cuối dòng
i	Chèn văn bản tại vị trí con trỏ hiện tại
o	Chèn văn bản vào dòng mới
O	Chèn văn bản vào dòng mới phía trên
s	Xoá ký tự hiện thời và chèn văn bản
S	Xoá dòng hiện thời và chèn văn bản

Xoá văn bản

Nếu bạn muốn xoá một ký tự đơn trong chế độ dòng lệnh thì dùng **x** và để xoá dòng hiện tại thì dùng **dd**.

Chú ý: Gần như tất cả các câu lệnh trong vi có thể được lặp lại bằng cách gõ thêm số lần lặp lại ở phía trước. Bạn cũng có thể cách này đối với các mục văn bản (như từ, câu, đoạn văn bản, ...) bằng cách thay thế thực thể (entity) sau câu lệnh.

Bảng 4: Các từ và ký tự

w	Chữ đơn
l	Ký tự đơn

Ví dụ:

Xoá một từ

`dw`

Xoá văn bản từ vị trí con trỏ đến cuối dòng hiện tại

`d$`

Xoá văn bản từ vị trí con trỏ đến cuối đoạn hiện tại

`d}`

SỬ DỤNG TRÌNH SOẠN THẢO VI

Bạn có thể xoá cùng lúc một mục văn bản đồng thời chuyển sang chế độ chèn với lệnh **c**. Như thường lệ bạn có thể sử dụng câu lệnh này với một mục văn bản như **w** hoặc **{**.

Copy / Paste

Thao tác copy trong **vi** là câu lệnh **y** (thay cho yank), và thao tác chèn là **p**.

Nếu một dòng được copy thì sẽ được chèn vào dòng tiếp theo phía dưới con trỏ.

Việc lựa chọn văn bản được thực hiện với các mục văn bản thông dụng như **w**, **l**, **}**, **\$**, ... Một số ngoại lệ được mô tả trong ví dụ dưới đây.

Ví dụ:

Sao chép văn bản từ vị trí hiện tại đến cuối dòng hiện thời

y\$

Sao chép toàn bộ dòng hiện thời

yy

Sao chép 3 dòng

3yy

Mục xoá cuối cùng thông thường được đưa vào bộ đệm và có thể được chèn với câu lệnh **p**. Điều này tương đương với thao tác copy và chèn.

Tìm kiếm

Do việc tìm kiếm đòi hỏi phải khớp theo mẫu do đó một lần nữa chúng ta lại đề cập đến các biểu thức chính qui (regular expressions – regex). Như một số công cụ thao tác với văn bản của UNIX như **grep** hoặc **sed**, **vi** cũng tuân thủ các biểu thức chính qui này.

Để thực hiện tìm kiếm, đầu tiên phải chuyển về chế độ dấu hai chấm. Câu lệnh / sẽ tìm kiếm từ vị trí hiện tại xuống cuối và câu lệnh ? sẽ tìm kiếm theo hướng ngược lại.

Để có thể thực hiện thao tác tìm kiếm và thay thế. Cú pháp tương tự như đối với **sed**.

Ví dụ:

Tìm từ bắt đầu từ chữ ‘comp’ trong toàn bộ văn bản

```
/\<comp>
```

Tìm dòng bắt đầu từ chữ cái z

```
/^z
```

Tìm trong toàn bộ văn bản với từ khoá ‘VAR’ và thay thế bằng ‘var’

```
:% s/VAR/var
```

Làm lại (Undo)

Chúng ta luôn có thể huỷ bỏ các thao tác vừa thực hiện (trong chế độ dòng lệnh) với câu lệnh **u**, và có thể sử dụng đối với tệp khi chưa thao tác ghi chưa được thực hiện.

Ghi văn bản

Câu lệnh ghi dữ liệu là `w`. Bằng cách này tài liệu sẽ mặc định được ghi lại. Người dùng cũng có thể xác định tên cho tệp cần ghi. Từng đoạn (portion) văn bản có thể được ghi lại sang tệp bản bản khác trong khi các tệp văn bản khác đang được đọc hoặc chèn tại tài liệu hiện thời. Ví dụ sau sẽ thể hiện điều này.

Ví dụ:

Ghi tài liệu hiện tại ra tệp có tên là 'newfile'

```
:w newfile
```

Ghi dòng 15 đến dòng 24 sang tệp có tên là 'extract'

```
:w 15,24 extract
```

Đọc từ tệp 'extract'. Văn bản sẽ được chèn vào vị trí con trỏ hiện tại

```
:r extract
```

Chú ý: trong ngữ cảnh *ché độ cột* (column mode) chúng ta phải thực hiện như sau

. là dòng hiện thời

\$ là cuối tài liệu

Thực hành

Tại root cp /var/log/messages to /tmp. Sử dụng chức năng tìm kiếm và thay thế của vi để tạo ra tất cả các dòng bắt đầu với “and end with”;

Gõ “u” để huỷ bỏ tất cả các thay đổi.

Copy /etc/lilo.conf tới /tmp, soạn thảo tệp này và thử copy/paste yy/p và cut/paste với dd/p

Kiểm tra kết quả của :x, ZZ, :quit, :wq, và :q! (câu lệnh nào sẽ ghi dữ liệu và câu lệnh nào không)

Kiểm tra thử kết quả sau khi sử dụng một số chế độ chèn văn bản như: A, a, O, o, S và s

Lựa chọn: Nếu bạn cài đặt gói **vim-enhanced** thì chương trình **vimtutor** sẽ cho thấy một số lựa chọn thông dụng của **vi**.