

Disclaimer

No liability for the contents of this document can be accepted. Use the concepts, examples and other content at your own risk. There may be errors and inaccuracies, that may of course be damaging to your system. Although this is highly unlikely, you should proceed with caution. The author does not accept any responsibility for any damage incurred.

All copyrights are held by their respective owners, unless specifically noted otherwise. Use of a term in this document should not be regarded as affecting the validity of any trademark or service mark.

Naming of particular products or brands should not be seen as endorsements.

UNIX is a registered trademark of The Open Group.

License

Copyright © 2003 — 2006 Gareth Anderson. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with no Invariant Sections, with no Front-Cover Texts, and with no Back-Cover Texts. A copy of the license can be found at the GNU Documentation License Site.

GNU Free Documentation License

Version 1.1, March 2000

Copyright (C) 2000 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is

not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the

machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition.

Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps,

when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version

to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in

the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on.

These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the

Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See Copyleft.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

Original Version

GNU/Linux Command-Line Tools Summary

Gareth Anderson <somecsstudent(at)gmail.com>

Chris Karakas – Conversion from LyX to DocBook SGML, Index generation

Bản dịch tiếng Việt

Tóm tắt các lệnh trên GNU/Linux

Nguyễn Đình Trung <nguyendinhtrung141@gmail.com>

Mục lục

Chương 1. Các quy ước về bố cục của tài liệu.....	3
Chương 2. Tư tưởng chung của UNIX.....	3
Chương 3. Các mẹo khi làm việc với hệ vỏ.....	4
3.1. Một số mẹo hay dùng.....	4
3.2. Những lệnh đã thực thi.....	7
3.3. Các tổ hợp phím khác.....	8
3.4. Các đầu cuối ảo và lệnh screen.....	9
Chương 4. Các trợ giúp trên dòng lệnh.....	10
Chương 5. Định hướng vào/ra.....	12
5.1. Khái niệm.....	12
5.2. Cách dùng.....	13
5.3. Thay thế lệnh.....	14
5.4. Thực hiện nhiều lệnh cùng lúc.....	15
Chương 6. Làm việc với hệ tập tin.....	15
6.1. Di chuyển trong hệ tập tin.....	16
6.1.1. Tìm kiếm tập tin.....	17
6.2. Làm việc với các tập tin và thư mục.....	19
6.3. Các công cụ sao chép, đổi tên và liên kết hàng loạt.....	23
Chương 7. Xem thông tin về hệ thống.....	25
7.1. Ngày giờ, thời gian và lịch.....	28
7.2. Thông tin về các phân vùng.....	28
Chương 8. Điều khiển hệ thống.....	29
8.1. Gắn kết và tháo gắn kết.....	29
8.2. Tắt và khởi động lại hệ thống.....	31
8.3. Điều khiển tiến trình.....	32
8.4. Điều khiển các dịch vụ.....	36
Chương 9. Quản lý người dùng.....	37
9.1. Người dùng và nhóm.....	37
Chương 10. Xử lý văn bản.....	38
10.1. Soạn thảo văn bản.....	38
10.2. Xem văn bản.....	39
10.3. Thông tin về văn bản.....	40
10.4. Xử lý văn bản.....	41
10.5. Tìm tập tin chứa văn bản mình cần.....	44
Chương 11. Các lệnh toán học.....	45
Chương 12. Mạng.....	46
12.1. Cấu hình mạng.....	47
12.2. Internet.....	48
12.3. Quản trị từ xa.....	50
Chương 13. Bảo mật.....	51

13.1. Một số lệnh về bảo mật.....	52
13.2. Phân quyền cho tập tin và thư mục.....	52
Chương 14. Nén và giải nén.....	54
14.1. Các định dạng nén khác.....	56
Chương 15. Đồ hoạ.....	57
Chương 16. Các lệnh cho MS-DOS.....	59
Chương 17. Đặt kế hoạch chạy lệnh trong chế độ nền.....	60
Chương 18. Các ký tự đại diện.....	61
18.1. Ký tự đại diện chuẩn.....	62
18.2. Biểu thức chính quy.....	63
18.3. Một số ký tự hay dùng (theo chuẩn POSIX).....	64

Chương 1. Các quy ước về bố cục của tài liệu

> Ghi chú

Phần này cung cấp cho bạn thông tin quan trọng đối với nội dung đang đề cập.

◆ Mẹo

Các tùy chọn, cú pháp hay dùng đối với lệnh ở bên trên

→ Quan trọng

Mục này chứa các thông tin rất quan trọng. Đây là những ghi chú cực kỳ quan trọng!

x Cảnh thận

Thông tin trong phần này có thể ảnh hưởng nghiêm trọng đến hệ thống của bạn. Hãy cẩn thận!

✓ Cảnh báo

Phần này chứa thông tin mà bạn không nên thực hiện, vì nó có thể làm ảnh hưởng nghiêm trọng tới hệ thống.

Phần này minh họa mã dùng làm ví dụ cho các lệnh.

Phần này chứa cú pháp cho câu lệnh.

Chương 2. Tư tưởng chung của UNIX

Trên UNIX, một lệnh, hay một công cụ, là một chương trình nhỏ để thực hiện một nhiệm vụ nhất định nào đó. Từ khi lệnh *pipe* “|” ra đời, cho phép đưa đầu ra của một lệnh làm đầu vào cho một lệnh khác, tất cả các công cụ trên UNIX đều tuân theo một tư tưởng chung: “nhiều chương trình nhỏ có thể kết hợp, phối hợp với nhau để thực hiện một tác vụ phức tạp, thay vì một chương trình đồ sộ và phức tạp có thể thực hiện nhiều chức năng”. Tư tưởng này của UNIX giúp cho các lập trình viên trên hệ thống có thể tận dụng các chương trình đã được viết sẵn bởi những lập trình viên khác trên toàn thế giới, mà không phải mất công viết lại. Đồng thời, nó cũng giúp cho tất cả các chương trình trên UNIX có thể phối hợp làm việc với nhau.

Tư tưởng này tiếp tục được kế thừa và phát huy trên các hệ điều hành GNU/Linux và các hệ điều hành gốc UNIX khác như FreeBSD, NetBSD, OpenBSD...

Chương 3. Các mẹo khi làm việc với hệ vỏ

Phần này cung cấp cho bạn những mẹo nhỏ, bao gồm tổ hợp phím gõ tắt, thực hiện lại các lệnh đã làm trước đó, và các cửa sổ dòng lệnh (đầu cuối) ảo.

> Nếu bạn không thể khởi động vào trong hệ thống

Nếu bạn không thể khởi động vào trong hệ thống của mình, hãy thử truy cập vào hệ vỏ và thực hiện các thao tác chỉnh sửa cần thiết để khôi phục hoạt động của hệ thống.

Bạn có thể truy cập vào hệ vỏ trong quá trình khởi động bằng cách đưa tùy chọn “init=/bin/sh” vào dòng lệnh khởi động khi GRUB nạp nhân Linux.

Xin xem chương Bảo mật để biết thêm chi tiết — cách làm là giống nhau, nhưng thay vì đặt tùy chọn “single”, bạn đặt tùy chọn “init=/bin/sh”.

3.1. Một số mẹo hay dùng

Tự động viết nốt lệnh

Nhấn TAB: Khi bạn đang gõ một vài chữ đầu tiên của lệnh, nếu trong hệ thống không có lệnh nào bắt đầu bằng những chữ mà bạn vừa nhập, thì bạn có thể để BASH tự động viết nốt tên lệnh cho mình bằng cách nhấn phím **TAB**. Nếu có nhiều lệnh có phần đầu trùng với những ký tự bạn đã nhập, BASH sẽ liệt kê tất cả các trường hợp có thể. Cách này cũng cho phép bạn dễ dàng viết nốt tên tập tin, thư mục, dùng trong việc di chuyển, sao chép và quản lý hệ tập tin.

Ngoài ra, còn một số tổ hợp phím ít dùng khác để viết nốt một số giá trị đặc biệt:

ESC- $\%$ ($\%$: ký tự đặc biệt): Viết nốt lệnh, hoặc liệt kê các trường hợp có thể, tương ứng với một phần câu lệnh mà bạn nhập. Nếu không có lệnh nào phù hợp, BASH sẽ nháy màn hình, hoặc kêu bíp để báo cho bạn biết.

CTRL-X- $\%$ ($\%$: ký tự đặc biệt): Liệt kê các trường hợp phù hợp với phần lệnh mà bạn vừa nhập vào. Nếu không tìm ra lệnh nào phù hợp, BASH sẽ nháy màn hình hoặc kêu bíp.

Các ký tự đặc biệt $\%$ có thể là:

- ~ (dấu ngã, ở phía trên phím TAB) : viết nốt tên người dùng.
- @ (dấu a còng) : viết nốt tên máy.
- \$ (dấu dollar) : viết nốt tên biến môi trường.
- ! (dấu chấm than) : tương ứng với việc nhấn **TAB**.

alias

Lệnh *alias* liệt kê các bí danh có trong hệ thống của bạn. Bí danh cho phép bạn thay thế một lệnh này bằng một lệnh khác. Thường thường, ta hay đặt bí danh cho một lệnh để thêm vào các tùy chọn mặc định của chính lệnh đó. Ví dụ, ta có thể thêm 3 lệnh *alias* vào tập tin *~/.bashrc* để mỗi khi khởi chạy hệ thống, các lệnh đó được thực hiện:

```
# các bí danh
alias cp='cp -vi' # luôn chạy cp ở chế độ tương tác, thông báo.
alias rm='rm -i' # luôn chạy rm ở chế độ tương tác.
alias mv='mv -i' # luôn chạy mv ở chế độ tương tác.
```

Đối với Mandriva Linux, bạn có thể thay đổi bí danh cho tất cả mọi người dùng trên máy bằng cách thêm vào tập tin */etc/profile.d/alias.sh* nội dung trên.

Ngoài ra, đối với một phiên làm việc, bạn có thể đặt bí danh ngay trên dòng lệnh hệ vỏ. Các bí danh này sẽ được sử dụng cho tới khi bạn đăng xuất khỏi hệ thống.

set -x

set là một lệnh có sẵn trong BASH. Nó có rất nhiều tùy chỉnh khác nhau, mỗi tùy chỉnh lại có một tác dụng khác. Xin tham khảo trang manpage của *set*.

Lệnh *set -x* yêu cầu BASH in ra mỗi lệnh chuẩn bị được chạy trước khi thực sự thực thi nó. Nhờ vậy, ta có thể biết chính xác những gì sắp diễn ra trước khi thực hiện lệnh. Ví dụ, đối với các lệnh

có chứa đối số là các ký tự đại diện (như *, ?), sau khi ta chạy `set -x`, BASH sẽ thay thế tất cả các ký tự đại diện bằng tên tập tin hoặc thư mục khớp với mẫu. Muốn tắt chế độ xem thử này, bạn gõ lệnh `set +x`.

Ví dụ: Sau khi chạy `set -x`, ta chạy lệnh `ls`:

```
ls
```

Đầu ra của lệnh sẽ được in ra trước khi lệnh chạy:

```
+ ls -F --color=auto
```

Nhờ thông tin này, ta biết rằng thực ra lệnh `ls` trong hệ thống đã được dùng để thay thế cho lệnh `ls -F -color=auto`.

\ (dấu xoạc ngược)

Khi ta dùng dấu “\” trước lệnh cần chạy, ta sẽ chạy lệnh gốc thay vì lệnh bí danh dùng thay thế nó. Ở ví dụ trên, nếu gõ

```
\ls
```

ta sẽ thực thi lệnh `ls` đơn giản, và không có đối số nào, chính xác như những gì ta gõ trên dòng lệnh.

x Khi dùng `rm`

Xin lưu ý rằng bí danh dùng cho lệnh xoá tập tin yêu cầu bạn xác nhận những tập tin cần xoá trên máy. Điều này giúp bạn không xoá nhầm những tập tin quan trọng, đặc biệt là các tập tin trên hệ thống.

Chỉ nên dùng `\rm` nếu bạn biết chính xác mình đang làm gì. Việc phục hồi các tập tin đã xoá khỏi hệ thống Linux không hề đơn giản!

Ngoài ra, ký tự “\” có thể được đặt trước một số ký tự đặc biệt (như là các ký tự đại diện hay dấu cách) để BASH không hiểu chúng là các chức năng quy ước. Ví dụ, để truy cập vào một thư mục *My Documents* thay vì gõ

```
cd My Documents
```

ta sẽ phải gõ lệnh sau:

```
cd My\ Documents
```

◆ Phím TAB

Khi dùng phím **TAB**, BASH sẽ tự động thêm dấu “\” khi cần thiết. Ví dụ, khi ta gõ tên thư mục *My Documents*, ta chỉ cần gõ *My* và nhấn **TAB**, và BASH sẽ tự động điền nốt thành *My\ Documents*.

script

Lệnh `script` cho ta biết tất cả các lệnh đã thực thi từ khi đăng nhập vào hệ thống đến thời điểm hiện tại. Ta có thể ghi lại tất cả những lệnh đã chạy, cũng như đầu ra của chúng vào một tập tin để xem lại khi cần.

~ (dấu ngã trên phím TAB)

Ký tự này được dùng để thay thế cho thư mục chính của ta. Ví dụ, nếu tên đăng nhập vào hệ thống của ta là “fred”, để di chuyển về thư mục chính */home/fred/* ta chỉ cần gõ lệnh

```
cd ~
```

Hoặc để chuyển đến thư mục */home/fred/tmp/*, ta gõ:

```
cd ~/tmp/
```

◆ Gõ tắt để chuyển đến thư mục chính

Dấu `~` còn có thể được dùng kết hợp với tên người dùng khác để chuyển tới thư mục chính của người dùng đó. Ví dụ, nếu trên máy, ngoài người dùng “fred” ở trên còn có người dùng “andrew”, ta có thể nhập `~andrew` để chuyển tới thư mục */home/andrew*. Lưu ý rằng ta phải

gõ chính xác tên người dùng, mà không được dùng ký tự đại diện như * và ?.

set bell-style none

Đây là lệnh dùng để tắt loa hệ thống (loa nhỏ thường nằm phía trước máy). Khi cần tắt loa hệ thống trong môi trường đồ họa X, ta có thể dùng lệnh `xset -b`. Bằng cách đưa những lệnh trên vào tập tin `~/.bashrc` hoặc `~/.bash_profile`, ta có thể tắt hoàn toàn tính năng báo bíp của máy mỗi khi có lỗi.

reset

Lệnh này khởi tạo lại đầu cuối hiện thời mà bạn đang làm việc. Chạy `reset` mỗi khi thấy văn bản trong đầu cuối quá tràn lan và ta không thể theo dõi kịp.

exit

Trong môi trường đồ họa X, lệnh này đóng cửa sổ chương trình Thiết bị cuối. Trong giao diện dòng lệnh, chạy lệnh này để đăng xuất (tương đương với việc nhấn **CTRL-D**).

logout

Đăng xuất khỏi hệ thống, tương đương với tổ hợp phím **CTRL-D**.

echo

Lệnh này lặp lại những gì bạn nhập vào phía sau nó. Ví dụ, khi gõ

```
echo "Hello, world!"
```

BASH sẽ in ra dòng `Hello, world!` ngay bên dưới. Nếu gõ lệnh

```
echo rm -R *
```

BASH sẽ thay thế dấu * bằng toàn bộ các thư mục và tập tin có trong thư mục hiện thời, cũng chính là những đối số truyền cho lệnh `rm -R`. Ta có thể xem thử tác dụng của một lệnh trước khi thực hiện nó, bằng cách đặt `echo` lên đầu câu lệnh.

Khi dùng `echo` với tùy chọn `-e` ta có thể sử dụng ký tự đặc biệt như `\t`, `\n`, `\` để định dạng đầu ra với khoảng tab và xuống dòng.

◆ Dùng `echo` để tránh nhầm lẫn

Gõ lệnh `echo <lệnh>` để thử trước tác dụng của lệnh đứng sau `echo` luôn là một cách rất hay để tránh những sai sót không mong muốn có thể xảy ra, vì `echo` sẽ thay thế các ký tự đại diện bằng các mẫu khớp tương ứng, và cho bạn biết chính xác các đối số bạn vừa truyền cho lệnh đứng sau.

3.2. Những lệnh đã thực thi

Dùng các lệnh đã thực thi

Khi nhấn **mũi tên lên** và **xuống** tại dấu nhắc dòng lệnh, ta có thể chọn được lệnh đã thực hiện trước đó. Dùng **mũi tên trái phải** để di chuyển trong câu lệnh hiện có, và sửa lại, ta sẽ tiết kiệm được rất nhiều thời gian trong khi làm việc với hệ vỏ.

history

Lệnh `history` hiển thị bản ghi hệ thống của BASH lưu lại tất cả các lệnh đã thực hiện mà ta đã chạy trước đây.

Để liệt kê `n` lệnh cuối cùng đã chạy, ta gõ:

```
history n
```

Để thực thi lệnh đánh số `n`, ta gõ `!n`, và gõ `!!` để thực thi lệnh cuối cùng vừa chạy.

`!n` sẽ thực hiện lệnh đã chạy cách đây `n` lệnh (nghĩa là `!!` tương đương với `!-1`).

`!chuỗi` sẽ thực hiện lệnh gần đây nhất bắt đầu bằng "chuỗi", và `!?chuỗi?` sẽ thực hiện lệnh gần

đây nhất có chứa “chuỗi”. Ví dụ:

```
!cd
```

sẽ thực hiện lệnh `cd` đã chạy gần nhất.

Gõ “*lệnh !**” để thực thi lệnh mới với các tham số của lệnh vừa thực hiện trước đó. Cú pháp này rất thuận tiện trong trường hợp bạn vừa gõ sai tên lệnh, nhưng tham số thì đúng. Ví dụ, bạn vừa gõ

```
emacs ~/my_text_file.txt
```

BASH báo lỗi, rằng không có chương trình `emacs!` Nhưng bạn không cần gõ lại dòng lệnh trên, mà chỉ cần gõ

```
emacs !*
```

để dùng lại đối số `~/my_text_file.txt`.

Tìm kiếm một lệnh đã chạy

Dùng tổ hợp **CTRL-R** để tìm kiếm trong bản ghi các lệnh đã chạy. Ví dụ, khi cần tìm lại lệnh `snort` đã dùng trước đây, ta nhấn **CTRL-R** rồi sau đó gõ “`snort`”. Nhấn **CTRL-R** tiếp để chuyển đến đúng lệnh mình cần. Sau đó, nếu muốn, bạn có thể dùng **mũi tên trái phải** để sửa đổi lại câu lệnh, và nhấn **ENTER** để thực thi lại nó.

3.3. Các tổ hợp phím khác

Dưới đây là các tổ hợp phím để bạn thao tác nhanh chóng trên cửa sổ dòng lệnh:

CTRL-D

Nhấn tổ hợp này để đăng xuất khỏi đầu cuối. Đối với các trình soạn thảo văn bản, đây chính là ký tự end-of-file (EOF - kết thúc tập tin) dùng để báo hiệu việc nhập liệu cho tập tin đã hoàn tất.

CTRL-Z

Tổ hợp này dùng một tiến trình đang chạy và đưa nó vào chạy trong chế độ nền. Ví dụ, khi bạn sửa đổi nội dung một văn bản bằng `vi` hoặc `emacs` và muốn tạm thời chuyển ra dòng lệnh BASH, bạn có thể nhấn **CTRL-Z** để đưa `vi` hoặc `emacs` vào chạy ở chế độ nền. Khi cần quay trở lại soạn thảo tiếp, bạn gõ lệnh `fg`. Chi tiết về lệnh `fg` xin xem trong phần Điều khiển tiến trình.

◆ Nếu `fg` không chạy

Bạn phải thử dùng lệnh `jobs` để biết cụ thể các tiến trình nào đang chạy ở chế độ nền, sau đó chạy lại `fg` với đối số là tên hoặc số định danh của tiến trình.

CTRL-A và CTRL-E

Tổ hợp **CTRL-A** đưa con trỏ về đầu dòng lệnh (giống phím **HOME**) và **CTRL-E** đưa con trỏ về cuối dòng lệnh (giống phím **END**).

CTRL-K và CTRL-W

CTRL-K cắt hoặc xóa tất cả những ký tự đứng sau con trỏ.

CTRL-W cắt hoặc xóa toàn bộ dòng lệnh bạn đang gõ.

CTRL-Y

Tổ hợp này dán những ký tự bạn xóa bằng **CTRL-K** hoặc **CTRL-W** vào vị trí con trỏ.

3.4. Các đầu cuối ảo và lệnh `screen`

Khi ở giao diện dòng lệnh, ta có thể nhấn **ALT-F*** để chuyển qua lại giữa các đầu cuối ảo. Thông thường, có 6 đầu cuối tương ứng với các phím **F1** đến **F6**. Riêng **F7** là đầu cuối ảo chạy giao diện đồ họa X. Nếu muốn chuyển tới giao diện dòng lệnh từ môi trường đồ họa X, ta phải nhấn **CTRL-ALT-F***. Ví dụ, chuyển về đầu cuối ảo thứ nhất từ môi trường X bằng tổ hợp **CTRL-ALT-F1**.

screen

Lệnh này khởi tạo nhiều đầu cuối ảo trên một đầu cuối ảo mà bạn đang sử dụng. Có thể nói, đây chính là “trình quản lý cửa sổ” ở giao diện dòng lệnh. Giả sử bạn đăng nhập từ xa thông qua *ssh* vào một máy tính khác để thực hiện vài tác vụ quản trị trên đó. Khi ấy, dĩ nhiên bạn không thể chạy các đầu cuối ảo (vì những đầu cuối ảo còn lại chỉ hoạt động trên máy bạn), nhưng vẫn muốn chạy 2 hoặc 3 lệnh cùng một lúc. Ta có thể dùng *screen* để “giả lập” nhiều đầu cuối ảo trên 1 đầu cuối ảo đang làm việc.

Sau khi gõ

```
screen
```

ta ở trong 1 đầu cuối ảo, muốn tạo ra đầu cuối ảo thứ hai, ta gõ **CTRL-A** rồi nhấn **C**. Để chuyển qua lại giữa các đầu cuối ảo, dùng **CTRL-N** để tới đầu cuối ảo kế, và **CTRL-P** để tới đầu cuối ảo trước. Ngoài ra, **CTRL-A** giúp ta chuyển qua lại giữa 2 đầu cuối ảo.

Xin xem thêm phần manpage của *screen* để biết thêm những cách dùng khác.

Chương 4. Các trợ giúp trên dòng lệnh

Phần này hướng dẫn bạn cách xem các thông tin trợ giúp sẵn có trên hệ thống GNU/Linux. Thông thường mỗi lệnh cài đặt trên GNU/Linux đều có tài liệu trợ giúp đi kèm.

man

Lệnh này hiển thị các thông tin tổng quát lấy từ các trang trợ giúp (manpage) cài đặt cùng chương trình. Ví dụ, gõ *man man* để đọc thông tin trợ giúp về lệnh *man*. Để thoát khỏi chế độ xem trợ giúp, bạn nhấn phím **q** hoặc **Q**.

Cú pháp của lệnh *man*:

```
man tên_lệnh
```

◆ Mở rộng

Ta có thể chỉ định thêm phân loại chứa trang hướng dẫn cần xem. Có nhiều lệnh trên Linux thực hiện nhiều tác vụ khác nhau, do đó trang hướng dẫn của chúng cũng được chia ra và đặt ở nhiều phân loại. Các phân loại được đánh số từ 1 đến 9, tương ứng với nội dung của chương trình như sau:

1. Các chương trình hoặc lệnh của hệ vỏ, có thể thực thi được.
2. Lệnh gọi hàm do hệ nhân cung cấp.
3. Lệnh gọi hàm do các thư viện cung cấp.
4. Các tập tin đặc biệt (thường nằm trong thư mục */dev*).
5. Các định dạng và chuyển đổi định dạng cho tập tin (ví dụ như */etc/passwd*).
6. Trò chơi.
7. Các nội dung lật vật.
8. Các lệnh quản trị hệ thống (thường dành riêng cho quản trị viên).
9. Chương trình con của hệ nhân (mở rộng).

Ta dùng cú pháp

```
man phân_loại tên_chương_trình
```

Ví dụ như lệnh *time* có 2 trang hướng dẫn ở phân loại 1 và 2, mặc định là trang hướng dẫn ở phân loại 1 được hiển thị. Nếu muốn xem trang ở phân loại 2, ta gõ

```
man 2 time
```

man -K

Lệnh này tìm kiếm tất cả các trang hướng dẫn và hiển thị trang nào chứa từ khoá được cung cấp ở sau. Cú pháp:

```
man -K từ_khoá
```

Để tìm các cụm từ, ta có thể dùng dấu “.

x Chú ý

Lệnh này sẽ sử dụng tương đối nhiều tài nguyên hệ thống, vì Linux sẽ tìm chuỗi từ khoá ta cung cấp trong **tất cả** các trang hướng dẫn trên máy.

man -f

Lệnh này hiển thị thông tin chi tiết gắn với tên lệnh sẽ cung cấp cho nó. Cú pháp:

```
man -f tên_lệnh
```

Người quản trị phải chạy lệnh *makewhatis* trước khi người dùng thông thường có thể thực hiện lệnh này.

➤ **Lệnh này tương đương với lệnh *whatis***

info

Một số lệnh được cài đặt kèm theo hướng dẫn chi tiết ở dạng siêu văn bản. Ta có thể xem thông tin hướng dẫn này bằng lệnh *info*.

Cú pháp của lệnh như sau:

```
info tên_lệnh
```

whatis

Lệnh này cung cấp thông tin mô tả ngắn gọn (1 dòng) về chức năng của một lệnh trên BASH. Tên lệnh dùng làm đối số cho *whatis* phải được nhập chính xác, nếu không *whatis* sẽ không đưa ra bất kỳ thông tin nào. Lệnh này lấy thông tin trợ giúp từ cơ sở dữ liệu *whatis* do người quản trị tạo ra.

Cú pháp của câu lệnh:

```
whatis tên_lệnh
```

makewhatis

Lệnh này tạo ra cơ sở dữ liệu (CSDL) *whatis*, dùng để tra cứu trợ giúp thông qua các lệnh *apropos*, *whatis* và *man -f*.

➤ Quyền quản trị

Việc tạo CSDL *whatis* tương đối mất thời gian, và bạn cần đăng nhập với tư cách quản trị (người dùng *root*) để thực thi lệnh này.

apropos

Tra cứu CSDL *whatis* và lọc ra một chuỗi từ khoá mà ta nhập, tương tự như lệnh *whatis* nhưng thay vì chỉ in ra thông tin dựa vào tên lệnh, nó in ra tất cả các thông tin có chứa chuỗi đối số mà ta nhập. Đầu ra của lệnh này cũng phụ thuộc vào CSDL *whatis* mà quản trị hệ thống tạo ra.

Cú pháp lệnh:

```
apropos chuỗi_cần_tìm
```

➤ Lưu ý

Quản trị viên phải chạy lệnh *makewhatis* để tạo CSDL *whatis* thì các lệnh *whatis*, *man -f* và *apropos* mới hoạt động được.

◆ Xem trợ giúp ngắn gọn

Ngoài các trang hướng dẫn và CSDL whatis, các chương trình trên GNU/Linux thường có sẵn tùy chọn `?`, `-h`, `--h` hoặc `--help` để in ra các thông tin ngắn gọn về cách dùng và chức năng của lệnh, tích hợp trong lệnh đó. Ví dụ, ta có thể xem thông tin ngắn về cách dùng lệnh `ls` bằng cách gõ:

```
ls --help
```

Chương 5. Định hướng vào/ra

Chương này sẽ đề cập đến những phương thức bạn có thể dùng để gửi đầu ra của một lệnh ra một tập tin hoặc làm đầu vào của một lệnh khác. Đây là một thao tác rất hữu ích mà bạn sẽ thường xuyên sử dụng khi làm việc với hệ vỏ BASH.

5.1. Khái niệm

Ba khái niệm dưới đây đều được gọi là “Luồng tập tin” (File stream). Chúng lưu thông tin từ một nguồn gửi tới, hoặc gửi thông tin tới một nơi khác. Trên UNIX, dữ liệu nhập vào từ bàn phím (đầu vào chuẩn), thông tin được in ra màn hình (đầu ra chuẩn) và thông báo lỗi (cũng được in ra màn hình) được coi là những luồng tập tin độc lập.

Đầu ra chuẩn

Đầu ra chuẩn là tất cả các thông tin mà chương trình in ra màn hình, trừ các thông báo lỗi. Thông tin gửi ra đầu ra chuẩn sẽ được lưu tạm trong bộ đệm, và sau đó được gửi ra màn hình.

Đầu vào chuẩn

Đây là dữ liệu mà người dùng nhập vào, thông thường là từ bàn phím.

Đầu ra lỗi chuẩn

Đây là thông báo lỗi mà chương trình in ra khi gặp sự cố. Các thông báo này cũng được gửi ra màn hình, và thường trộn với các thông tin đầu ra chuẩn. Tuy nhiên, các thông báo lỗi đều được in trực tiếp ra màn hình mà không lưu vào bộ đệm, nên ngay khi có sự cố xảy ra, thông báo lỗi sẽ lập tức được in ra màn hình.

5.2. Cách dùng

>

Ký hiệu lớn hơn được dùng khi ta cần gửi thông tin tới một nơi nào đó (ví dụ như tới một tập tin văn bản).

Ví dụ:

```
cat file1 file2 > file1_file2.txt
```

Lệnh trên in liên tiếp nội dung của 2 tập tin là `file1` và `file2`, rồi gửi toàn bộ thông tin in ra vào tập tin `file1_file2.txt`. Nếu tập tin `file1_file2.txt` đã tồn tại và có nội dung, nó sẽ bị xóa đi và ghi nội dung mới vào.

<

Ký hiệu nhỏ hơn được dùng để chèn thông tin từ nơi nào đó (ví dụ như một tập tin) giống như khi bạn tự nhập vào. Thông thường ta dùng nó để chạy cùng các lệnh yêu cầu có thông tin được nhập từ đầu vào chuẩn.

Ví dụ, khi chạy lệnh `tr`:

```
tr '[A-Z]' '[a-z]' < file.txt > fileNew.txt
```

Câu lệnh trên đưa nội dung của tập tin *file.txt* vào đầu vào của lệnh *tr*, và đưa đầu ra khi lệnh chạy vào tập tin *fileNew.txt*.

>>

Tương tự như ký hiệu *>*, nhưng thay vì tạo mới hoặc đề lên tập tin đã có, *>>* sẽ thêm đầu ra của lệnh vào cuối tập tin nếu nó đã có nội dung nào đó, hoặc tạo tập tin nếu chưa có.

<<

Ký hiệu *<<* hay được dùng với các lệnh lấy thông tin nhập vào từ đầu vào chuẩn. Nó rất hay được dùng trong các văn lệnh hệ vỏ (shell script).

Giả sử bạn gõ *<< word* (word có thể là một chuỗi bất kỳ) phía sau lệnh, khi ấy lệnh sẽ lấy thông tin từ bàn phím cho tới khi bạn gõ *word* để báo hiệu kết thúc việc nhập liệu. Dùng *<<* cũng giống như khi nhấn tổ hợp **CTRL-D** trừ việc *<<* dùng một chuỗi để báo hiệu kết thúc việc nhập liệu thay vì ký tự EOF.

Ví dụ, gõ “cat” (không cho thêm tùy chọn nào phía sau), lệnh này sẽ lấy dữ liệu từ bàn phím tới khi ta nhấn tổ hợp **CTRL-D**. Tuy nhiên, khi gõ “cat *<< KẾTTHÚC*”, và sau khi đã nhập dữ liệu xong xuôi, thay vì nhấn **CTRL-D**, ta gõ KẾTTHÚC để kết thúc việc nhập liệu. Chuỗi KẾTTHÚC sẽ không được lưu lại.

2>

Đổi hướng đầu ra lỗi chuẩn. Ví dụ, khi muốn đổi hướng đầu ra lỗi tới */dev/null* nhằm bỏ qua các thông báo lỗi, ta chỉ việc thêm “2> */dev/null*” vào phía cuối dòng lệnh. Ví dụ:

```
make some_file 2> /dev/null
```

Dòng lệnh trên chạy lệnh *make* cho tập tin *some_file* và gửi tất cả các thông báo lỗi tới */dev/null*

Lệnh “ống” cho phép ta đưa đầu ra của một lệnh tới đầu vào của một lệnh khác, giống như nối liền 2 lệnh bằng 1 “ống dẫn liệu” vậy! Ví dụ về cách dùng lệnh này:

```
cat file1.txt file2.txt | less
```

Lệnh *cat* nối liền 2 tập tin *file1.txt* và *file2.txt*. Sau đó, thay vì gửi dữ liệu ra màn hình, nó gửi dữ liệu tới lệnh *less*, nhờ đó ta có thể xem nội dung tổng hợp của 2 tập tin dễ dàng hơn.

tee

Lệnh này gửi đầu ra của một lệnh tới một tập tin và tới đầu ra chuẩn. Nó giống như đoạn ống hình T trong đường ống nước: chia dữ liệu đầu vào thành 2 đường đầu ra. Ví dụ:

```
ls /home/user | tee my_directory.txt
```

Lệnh *ls* liệt kê nội dung thư mục */home/user*, rồi gửi tới đầu vào của lệnh *tee*. Lệnh *tee* sau đó sẽ vừa gửi dữ liệu đầu vào tới đầu ra chuẩn (màn hình) vừa gửi tới tập tin *my_directory.txt*.

&>

Đổi hướng đầu ra chuẩn và đầu ra lỗi chuẩn tới một vị trí nào đó. Ví dụ:

```
make &> /dev/null
```

Lệnh trên sẽ gửi tất cả các thông tin lỗi cũng như thông tin đầu ra của lệnh *make* tới */dev/null* để bỏ đi, do đó ta sẽ không nhận được bất kỳ thông tin đầu ra nào khi chạy lệnh.

5.3. Thay thế lệnh

Thay thế lệnh, xét đến cùng, cũng chỉ là một cách khác để thực hiện lệnh “ống” / và ta có thể dùng thay thế lệnh hoặc ống tùy thích.

Có hai cách thay thế lệnh như sau:

Dùng dấu trích dẫn (`)

Ta có thể dùng ký hiệu ``` để thực hiện trước 1 lệnh, rồi dùng đầu ra của lệnh đó làm đầu vào của 1 lệnh khác. Cú pháp:

```
lệnh1 `lệnh2 -tùy_chọn`
```

◆ Phím `

Nằm cùng phím với dấu ngã ~, bên trên phím TAB.

Dùng ký hiệu dollar (\$)

Tương tự như trên, nhưng thay cặp ngoặc trích dẫn `...` bằng dấu \$:

```
lệnh1 $(lệnh2 -tùy_chọn)
```

5.4. Thực hiện nhiều lệnh cùng lúc

Chạy lệnh thứ 2 chỉ khi thực hiện thành công lệnh đầu

Ta dùng phép logic VÀ để làm điều này. Cú pháp:

```
lệnh1 && lệnh2
```

Nếu *lệnh1* chạy hoàn hảo và không có lỗi nào xảy ra, *lệnh2* mới được thực thi. Nếu có lỗi và *lệnh1* bị đình trệ, *lệnh2* sẽ không được thực hiện.

Thực hiện lệnh thứ 2 chỉ khi lệnh đầu bị lỗi

Ta dùng phép logic HOẶC cho trường hợp này. Cú pháp:

```
lệnh1 || lệnh2
```

theo đây, *lệnh2* sẽ chỉ được thực hiện nếu *lệnh1* bị lỗi. Nếu *lệnh1* không lỗi, *lệnh2* sẽ bị bỏ qua.

Thực hiện 2 lệnh liên tiếp

Ta không quan tâm đến việc các lệnh chạy ra sao, có xảy ra lỗi không, mà chỉ cần đảm bảo là chúng được chạy liên tiếp:

```
lệnh1; lệnh2
```

Ở đây, *lệnh2* sẽ được thực hiện sau khi *lệnh1* kết thúc.

◆ Chạy nhiều lệnh hơn

Ta có thể chạy bao nhiêu lệnh liên tiếp nhau cũng được, miễn là ngăn cách chúng bằng dấu chấm phẩy (;).

Chương 6. Làm việc với hệ tập tin

Chương này sẽ đề cập đến các lệnh để di chuyển trên cây cấu trúc hệ tập tin và xử lý các tập tin trên đó, đặc biệt là thao tác tìm kiếm tập tin và đổi tên hàng loạt.

6.1. Di chuyển trong hệ tập tin

cd

Di chuyển sang thư mục khác.

Để đến thư mục cha, ta gõ

```
cd ..
```

Để đến thư mục vừa dùng trước đây, ta gõ

```
cd -
```

Ta cũng có thể nhập thư mục cần đến dưới dạng đường dẫn tương đối và đường dẫn tuyệt đối. Đường dẫn tuyệt đối luôn bắt đầu bằng dấu /. Ví dụ, nếu muốn tới thư mục /boot/grub:

```
cd /boot/grub
```

đường dẫn tuyệt đối đi từ gốc cây thư mục xuống tới nhánh cần chuyển tới một cách tuần tự. Đường dẫn tương đối không bắt đầu bằng dấu /. Giả sử như ta đang ở thư mục /root, và muốn chuyển tới thư mục /root/music/ABBA/, ta chỉ việc gõ

```
cd music/ABBA
```

ls

Liệt kê các tập tin và thư mục. Lệnh *ls* (không có tham số đi kèm) sẽ liệt kê tất cả các tập tin và thư mục hiện có trong thư mục hiện hành, nhưng bỏ qua các tập tin ẩn (có tên bắt đầu bằng dấu chấm, ví dụ như *.bashrc*). Ngoài ra, *ls* còn có một số tùy chọn như sau:

- *-l* : liệt kê chi tiết quyền hạn, kích thước, ngày sửa đổi và người sở hữu.
- *-a* : liệt kê cả các tập tin ẩn (có dấu *.* ở đầu tên)
- *-d* : chỉ liệt kê tên thư mục, thay vì cả nội dung của thư mục đó.
- *-F* : thêm các ký hiệu vào một số tập tin nhất định, ví dụ như thêm dấu *** cho các tập tin thực thi được.
- *-S* : sắp xếp danh sách in ra theo kích cỡ giảm dần.
- *-R* : liệt kê đệ quy toàn bộ các tập tin và thư mục, bao gồm cả nội dung của các thư mục con.

Cú pháp của lệnh này như sau:

```
ls -tùy_chọn
```

Ta cũng có thể cung cấp một chuỗi chứa các ký tự đại diện để liệt kê những mục khớp:

```
ls -tùy_chọn chuỗi
```

Về các ký tự đại diện, xin xem chương để biết thêm.

Ta có thể dùng lệnh *ls -d* để chỉ liệt kê các thư mục khớp với một mẫu nào đó. Ví dụ, khi gõ "*ls -d */*" BASH sẽ in ra tất cả các thư mục con nằm trong thư mục hiện hành, nhưng bỏ qua nội dung của chúng. Đây là một thao tác rất hay dùng, và bạn nên dùng lệnh alias để đặt *lsd* tương đương với "*ls -d */*". Dưới đây là một số ví dụ:

```
ls -d */
```

Liệt kê tất cả các thư mục con trong thư mục hiện hành.

```
ls -d string*
```

Liệt kê các thư mục có tên bắt đầu bằng "string".

```
ls -d /usr/**/doc
```

Liệt kê tất cả các thư mục nằm dưới thư mục */usr* 2 cấp, và có chứa thư mục con tên là *doc*. Đây cũng là một lệnh rất hữu dụng.

> Đặt alias cho *ls*

Thông thường, người ta hay dùng lệnh *alias* để đặt *l="ls"*, *la="ls -a"* và *ll="ls -l"* vào tập tin *~/.bashrc* để tiện sử dụng.

pwd

In ra đường dẫn tuyệt đối tương ứng với thư mục hiện hành. Lệnh này cho bạn biết đường dẫn đầy đủ của thư mục mà bạn đang làm việc trong đó. Cú pháp lệnh:

```
pwd
```

tree

Lệnh này đưa ra một cấu trúc cây dưới dạng văn bản, với gốc cây chính là thư mục hiện thời hoặc thư mục được chọn. Tất cả các nội dung bên trong một thư mục sẽ được liệt kê dưới dạng cây cấu trúc ra màn hình. Lệnh *tree* có rất nhiều tùy chọn. Xin xem trang hướng dẫn của nó để biết thêm chi tiết.

Cú pháp:

```
tree
```

hoặc

```
tree -tuỳ_chọn /thư/mục/cần/liệt/kê
```

6.1.1. Tìm kiếm tập tin

find

Lệnh *find* tìm một hoặc nhiều tập tin khớp với một mẫu nào đó. Đây là một lệnh tương đối phức tạp. Dưới đây là một vài ví dụ về cách dùng *find*:

```
find / -name file_name
```

Lệnh trên sẽ tìm tất cả các tập tin có tên là "*file_name*" trên toàn bộ hệ tập tin. Tùy chọn "*-name*" sẽ yêu cầu tập tin phải có cùng dạng chữ hoa/thường với mẫu tìm kiếm. Để bỏ qua phân biệt chữ hoa/thường, ta thay "*-name*" bằng "*-iname*".

Các tùy chọn "*-regex*" và "*-iregex*" cho phép bạn tìm các tập tin có biểu thức chính quy nào đó (có và không phân biệt chữ hoa/thường).

Tùy chọn "*-exec*" thực hiện một lệnh nào đó đối với mỗi tập tin mà *find* tìm ra. Cú pháp này có dạng sau: gõ lệnh *find* để tìm ra tập tin mình cần, rồi thêm tùy chọn *-exec* vào cuối, sau đó là mẫu:

```
lệnh_cần_chạy ① '{}' ② tham_số_cho_lệnh ③
```

Chú thích:

- ① đây là lệnh mà bạn muốn chạy (ví dụ, nếu muốn sao chép các tập tin tìm thấy, ta gõ *cp*).
- ② cặp ngoặc nhọn {} đại diện cho tập tin được tìm ra (ví dụ, nếu tìm thấy tập tin tên là *shopping.doc* thì cặp ngoặc sẽ được thay thế bằng chuỗi "*shopping.doc*"). Thường thường ta dùng dấu \ hoặc cặp dấu ngoặc đơn ' để BASH chỉ hiểu các ký tự khoảng trống, tab là tên tập tin.
- ③ ký hiệu dùng để báo hiệu kết thúc lệnh, cũng nằm trong dấu \ hoặc '.

Ví dụ về lệnh *find* dùng tùy chọn *-exec*:

```
find / -name '*.doc' -exec cp '{}' /tmp/ ';' 
```

Lệnh trên tìm tất cả các tập tin có đuôi là ".doc" và sao chép sang thư mục */tmp/*.

Tùy chọn *-path* được dùng để bỏ qua một thư mục nào đó. Lưu ý là không thêm dấu "/" vào cuối tên thư mục đặt sau *-path*. Tùy chọn *-prune* bỏ qua tất cả các thư mục con nếu có tùy chọn *-path* đứng trước nó. Ví dụ:

```
find / -path '/mnt/win_c' -prune -o -name "string" -print
```

Lệnh trên tìm toàn bộ hệ tập tin và đưa ra các tập tin có tên là "string", nhưng không thực hiện tìm kiếm trong thư mục */mnt/win_c* và tất cả các thư mục con bên trong nó. Khi dùng tùy chọn *-path* bạn có thể dùng các ký tự đại diện và viết nhiều tùy chọn *-path* liên tiếp để bỏ qua các thư mục không cần tìm.

Lệnh *find* có rất nhiều tùy chọn hữu ích khác. Xin xem thêm trang hướng dẫn của nó để biết thêm.

slocate

Lệnh *slocate* liệt kê tất cả các tập tin trong hệ thống khớp một mẫu nào đó. Nếu tên tập tin đầy đủ (đường dẫn + tên tập tin) có chứa mẫu cần khớp, *slocate* sẽ in tập tin đó ra.

whereis

Lệnh *whereis* tìm chính xác tên một chương trình trong tất cả các tập tin nhị phân, mã nguồn và trang hướng dẫn. Lưu ý là ta phải nhập tên chính xác tên chương trình cần tìm. Nếu chỉ biết một phần tên chương trình, hãy dùng lệnh *slocate*.

Cú pháp:

```
whereis tên_chương_trình
```

which

Tìm chính xác tên một chương trình trong tất cả các tập tin thực thi được, nằm trong các thư mục ghi trong biến môi trường *\$PATH* của người dùng.

Dùng thêm tùy chọn *-a* để liệt kê tất cả các kết quả (khi có nhiều hơn 1 tập tin thực thi được khớp với tên_chương_trình).

Cú pháp:

```
which tên_chương_trình
```

6.2. Làm việc với các tập tin và thư mục

mkdir

Tạo một thư mục. Dùng tùy chọn `-p` để tự động tạo các thư mục con nếu cần. Ví dụ:

```
mkdir -p /home/matt/work/maths
```

Lệnh trên sẽ tạo lần lượt thư mục `work` và `maths` bên trong thư mục chính `/home/matt`.

rm

Xoá tập tin hoặc thư mục. Ta có thể dùng ký tự đại diện cho tên thư mục, tập tin cần xoá.

```
rm -tùy_chọn tập_tin_hoặc_thư_mục
```

Tùy chọn `-R` hoặc `-r` sẽ thực hiện xoá đệ quy, tức là xoá hoàn toàn nội dung bên trong các thư mục con chứa trong thư mục mà ta dùng làm đối số cho lệnh. Tùy chọn `-f` sẽ thực hiện việc xoá thư mục/tập tin mà không hỏi lại.

rmdir

Xoá một thư mục rỗng. Nếu cần xoá một thư mục không rỗng, ta dùng lệnh

```
rm -R thư_mục
```

Lệnh `rmdir` chỉ xoá thư mục khi bên trong nó không có gì! Cú pháp:

```
rmdir thư_mục
```

mv

Di chuyển một tập tin hay một thư mục tới vị trí mới hoặc đổi tên nó. Ví dụ:

```
mv filename1 filename2
```

Lệnh trên đổi tên tập tin `filename1` thành `filename2`.

```
mv tập_tin_gốc thư_mục_mới
```

Lệnh này di chuyển tập tin `tập_tin_gốc` sang thư mục “`thư_mục_mới`”. Ta có thể dùng các ký tự đại diện để di chuyển nhiều tập tin cùng một lúc.

> Di chuyển và đổi tên

Ta cũng có thể vừa di chuyển vừa đổi tên cho một tập tin cùng một lúc, bằng việc thêm vào tên mới của tập tin trong đối số bên phải (đích). Ví dụ:

```
mv /etc/config configuration.txt /home/joe/backupconfig
```

sẽ di chuyển tập tin “`configuration.txt`” tới thư mục `/home/joe/` và đổi tên nó thành “`backupconfig`” .

cp

Sao chép tập tin. Tùy chọn `-R` (hoặc `-r`) cho phép sao chép đệ quy toàn bộ các thư mục và thư mục con sang vị trí mới. Cú pháp lệnh:

```
cp -tùy_chọn tập_hoặc_thư_mục vị_trí_mới
```

Các ví dụ:

```
cp file1 file2
```

Lệnh trên sao chép tập tin `file1` thành `file2` trong thư mục hiện hành.

```
cp /tmp/file1 ~/file2 /mnt/win_c
```

Sao chép 2 tập tin là `/tmp/file1` và `~/file2` vào trong thư mục `/mnt/win_c`. Lưu ý là đối số cuối cùng chính là thư mục cần được chép đến.

```
cp -R thư_mục thư_mục_đích
```

Sao chép toàn bộ nội dung thư mục `thư_mục` vào bên trong thư mục `thư_mục_đích`.

Ta có thể dùng các ký tự đại diện để thực hiện sao chép nhiều tập tin và thư mục cùng một lúc. Ngoài ra, nếu thêm tùy chọn `-u` lệnh `cp` sẽ chỉ sao chép các tập tin mới hơn vào vị trí mới thay vì ghi đè.

ln

Tạo liên kết tới một tập tin. Có hai loại liên kết trên UNIX:

- **Liên kết cứng :**

Là con trỏ chỉ tới tập tin (số được gắn với tập tin khi chạy lệnh `ls -l`). Mỗi một liên kết cứng là một tham chiếu tới một tập tin. Ta chỉ có thể xoá tập tin gốc khi **tất cả** các liên kết cứng tới tập tin đã bị xoá. Cú pháp lệnh:

```
ln tập_tin_nguồn tên_liên_kết
```

Khi muốn xoá tập tin nguồn, ta phải xoá cả liên kết (`tên_liên_kết`) lẫn tập tin nguồn (`tập_tin_nguồn`).

- **Liên kết tượng trưng :**

Liên kết tượng trưng tham chiếu đến tập tin hoặc thư mục. Khi xoá thư mục hoặc tập tin gốc, liên kết tượng trưng sẽ bị hỏng. Liên kết tượng trưng giống như khái niệm “lối tắt” (shortcut) trên hệ điều hành Windows. Cú pháp lệnh:

```
ln -s nguồn tên_liên_kết
```

Lệnh trên tạo liên kết tượng trưng tên là `tên_liên_kết` trỏ tới thư mục hoặc tập tin nguồn. Nếu ta xoá tập tin hoặc thư mục nguồn, liên kết sẽ bị hỏng và không làm việc.

shred

Xoá hoàn toàn một tập tin bằng cách ghi đè lên đĩa cứng. Sau khi *shred*, các thông tin đã xoá sẽ không thể lấy lại được, kể cả bằng phần cứng hay phần mềm.

Ví dụ:

```
shred -n 2 -z -v /dev/hda1
```

Câu lệnh này lấy dữ liệu ngẫu nhiên (tùy chọn `-n`) ghi đè phân vùng `/dev/hda1` hai lần (số 2 đứng sau `-n`), rồi sau đó tiếp tục điền số 0 vào (tùy chọn `-z`), và thông báo quá trình làm việc của chương trình cho bạn biết (tùy chọn `-v`). Mỗi một lần ghi dữ liệu lên toàn bộ phân vùng như vậy tương đối mất thời gian, nên ta đã chỉ định số lần ghi chỉ có 2 lần thay vì mặc định là 25 lần!

Vì *shred* ghi dữ liệu ở cấp thấp, nên bất kể hệ tập tin một phân vùng có là loại nào (FAT, NTFS, ext3 ...) thì toàn bộ dữ liệu trên đó sẽ bị “nghiền” và không thể khôi phục lại được nữa. Sau khi hoàn tất lệnh này, ta có thể yên tâm tắt máy rồi bán hoặc đưa ổ đĩa cho người khác mà không sợ thông tin cá nhân của mình bị khôi phục và sử dụng.

➤ Không phải hệ tập tin nào *shred* cũng chạy được

Lưu ý rằng như đã nói trong trang hướng dẫn của *shred*, chương trình làm việc không hiệu quả trên các hệ thống tập tin cấu trúc bản ghi hoặc journal như JFS, ReiserFS, XFS, ext3 và các cấu trúc hệ tập tin hiện đại.

◆ Các phần mềm tương tự

shred có nhược điểm lớn khi chạy trên hệ tập tin, đó là ta phải cài nó lên hệ tập tin, nên ta không thể chạy trên hệ tập tin của hệ điều hành mình có, và cũng không thể dùng *shred* trên máy windows vì ta không thể cài *shred* lên máy này được.

Có nhiều phần mềm để phục vụ việc nghiền dữ liệu trên ổ cứng khác,

như *chattr*.

du

Hiện thông tin về kích thước tập tin. Dùng *du tên_tập_tin* để biết kích thước của tập tin nào đó. Nếu đối số là một thư mục, *du* sẽ hiện thông tin kích cỡ của từng tập tin và thư mục con trong thư mục đó.

Các tùy chọn của lệnh *du*:

- *-c* : in ra kích cỡ tổng cộng sau khi đã quét qua tất cả các đối tượng.
- *-s* : in ra kích cỡ tổng cộng cho từng đối tượng (các thư mục con chẵn hạn).
- *-h* : in ra dưới dạng Megabyte, kilobyte để người dùng dễ theo dõi hơn.

Khi dùng lệnh *du -hs* ta sẽ biết kích thước tổng cộng của thư mục và tất cả các thư mục con trong đó.

Cú pháp lệnh như sau:

```
du -tùy_chọn thư_mục_hoặc_tập_tin
```

Ví dụ:

```
du -hs *
```

Lệnh này liệt kê kích thước của tất cả các tập tin có trong thư mục hiện hành, cũng như kích cỡ các thư mục con dưới dạng dễ đọc.

file

Cho biết kiểu tập tin của một tập tin do ta cung cấp, ví dụ, kiểu nhị phân, ảnh, văn bản...

Cú pháp lệnh:

```
file tên_tập_tin
```

stat

Thông tin chi tiết về một tập tin, bao gồm số inode, ngày truy cập (ngày tạo). Cú pháp:

```
stat tập_tin
```

dd

Sao chép dữ liệu cấp thấp, thường được dùng trong việc sao đĩa cứng (để cập trong phần) hoặc tạo tập tin ảnh đĩa CD. *dd* còn có chức năng chuyển đổi tập tin và thay đổi kích cỡ khối dữ liệu dùng lưu trữ các tập tin.

Trong cú pháp của lệnh *dd* thì các tham số *bs* và *count* là không bắt buộc và ta có thể áp dụng lệnh cho tập tin thay vì ổ đĩa lưu trữ...

➤ Lưu ý

dd là một lệnh khó dùng, yêu cầu người sử dụng phải hiểu rõ các chức năng và công dụng của nó. Do vậy, hãy cẩn thận khi sử dụng lệnh này.

Cú pháp của lệnh này như sau:

```
dd if=/dev/xxx of=/dev/xxx bs=xxxx count=x
```

Trong câu lệnh *dd*, có hai tham số bắt buộc là *if* (tập tin đọc) và *of* (tập tin cần ghi ra). Các tham số tùy chọn là *bs*, thiết lập số byte sẽ được đọc và ghi ứng với một khối dữ liệu được sao chép, và tham số *count* chính là số khối dữ liệu cần sao chép.

✓ Cảnh báo

Lệnh *dd* làm việc với dữ liệu cấp thấp. Nó có khả năng ghi đè các thông tin quan trọng, như thông tin MBR (vùng đầu đĩa cứng, chứa dữ liệu khởi động) hoặc các phần quan trọng trên đĩa cứng. Do đó, hãy cẩn thận khi dùng lệnh này (đặc biệt khi làm việc với các thiết bị lưu trữ thay vì các tập tin).

touch

Lệnh *touch* thường được dùng để tạo một tập tin trống và cập nhật lại ngày tháng truy cập cuối cùng của một tập tin đang tồn tại. Cú pháp sau sẽ tạo ra tập tin mới, không chứa nội dung gì bên trong, tên là *filename*:

```
touch filename
```

Câu lệnh sau sẽ thay đổi thời gian tạo ra tập tin *myfile.txt* thành 9:15 ngày 7 tháng 5:

```
touch -t 05070915 myfile.txt
```

Chuỗi số phía sau tùy chỉnh *-t* ở đây có dạng MM-DD-hh-mm (tháng - ngày - giờ - phút).

Hoặc, ta cũng có thể dùng định dạng ngày tháng tiếng Anh như sau:

```
touch -d '5 May 2000' myfile.txt
```

Định dạng ngày tháng được đặt sau tùy chỉnh *-d* giống như trong lệnh *date* (chương Ngày giờ, thời gian và lịch).

split

Lệnh *split* chia một tập tin thành nhiều tập tin. Trong câu lệnh, tùy chọn *-b xx* chia nhỏ tập tin ra thành nhiều phần có kích thước *xx* byte, *-k xx* chia ra thành các tập tin cỡ *xx* KB và *-m* đối với MB. Ta thường dùng *split* để chia nhỏ tập tin ra nhiều phần, và dùng *cat* để ghép lại. Cú pháp lệnh như sau:

```
split -tùy_chọn tập_tin
```

Lệnh này sẽ chuyển 1000 dòng của tập tin được chia vào thành 1 phần nhỏ hơn. Các tập tin thành phần sẽ có dạng "tập_tina", "tập_tinb", "tập_tinc"...

6.3. Các công cụ sao chép, đổi tên và liên kết hàng loạt

Trên GNU/Linux có rất nhiều cách để thực hiện việc sao chép, đổi tên nhiều tập tin cùng một lúc. Nhiều văn lệnh PERL cho phép ta đổi đuôi của các tập tin một cách dễ dàng. Xin tham chương .

Dưới đây là 3 cách để đổi tên hàng loạt tập tin bằng lệnh *mmv*, *rename* (một văn lệnh PERL) và một vài văn lệnh hệ vỏ BASH.

mmv

mmv là một công cụ di chuyển, sao chép và đổi tên hàng loạt, sử dụng các ký tự đại diện chuẩn. Tuy nhiên, phân hướng dẫn sử dụng của lệnh này tương đối khó hiểu. Theo đó, ký tự ";" được dùng đại diện cho các tập tin bất kỳ nằm trong cây thư mục bên dưới thư mục hiện hành (tức là chương trình sẽ tìm kiếm các tập tin nằm trong tất cả các thư mục con của thư mục hiện tại).

Ví dụ:

```
mmv \*.JPG \#1.jpg
```

Mẫu "*.JPG" khớp với tất cả các tập tin có đuôi là .JPG, và #1 ở mẫu thứ 2 chính là phần số khớp tương ứng với dấu * ở mẫu thứ nhất, do vậy lệnh này sẽ đổi đuôi của các tập tin ".JPG" thành ".jpg".

Mỗi khi ta dùng mẫu dạng \(\ký_tự_đại_diện), ta có thể dùng #x để lấy lại giá trị khớp với ký tự đại diện đã dùng; x là một số dương, lớn hơn hoặc bằng 1.

◆ Thông tin thêm về *mmv*

Lưu ý rằng các tùy chỉnh dùng với *mmv* cũng có hiệu lực đối với các lệnh trong bộ công cụ, như *mcp* (sao chép hàng loạt) *mad* (thêm nội dung tập tin nguồn vào một tập tin đích) *mln* (tạo nhiều liên kết cùng lúc tới một tập tin).

Một công cụ viết bằng Java khác có chức năng tương đương, chạy trên cả Linux và Windows đó là *Esomaniac*.

rename

rename là một văn lệnh PERL cho phép đổi tên nhiều tập tin khớp với biểu thức chính quy nào đó. Ví dụ, để đổi đuôi “.JPG” thành “.jpg”, ta làm như sau:

```
rename 's/\.JPG$/ .jpg/' *.JPG
```

> Cài đặt *rename*

Ta có thể lên CPAN để lấy *rename* về. CPAN là một mạng chia sẻ các thư viện PERL trên khắp thế giới. Để sử dụng CPAN, ta có thể chạy lệnh *cpan* trên hệ vỏ, và cài đặt *rename* từ trên Internet.

Văn lệnh BASH

Dùng văn lệnh BASH cũng là cách để ta đổi tên nhiều tập tin cùng lúc. Ta phải viết một loạt các lệnh, ghi vào một tập tin, trao quyền thực thi tập tin đó và chạy nó. Dưới đây xin lấy một ví dụ về văn lệnh BASH:

```
for i in *.JPG; do mv $i `basename $i JPG`jpg; done
```

Dòng lệnh trên thực hiện vòng lặp đối với tất cả các tập tin có đuôi JPG trong thư mục hiện hành, sau đó đổi tên chúng dựa vào lệnh *basename* (bỏ qua phần mở rộng của tập tin). Lệnh *basename* sẽ loại bỏ đường dẫn đứng trước tên tập tin nếu có, cũng như phần đuôi phía sau của tập tin, nếu được chỉ định. Ví dụ: *basename /usr/share/myfile.txt.txt .txt* sẽ in ra *myfile.txt*.

◆ Lưu ý

Câu lệnh trên chỉ làm việc với tên tập tin không chứa khoảng trắng, như dấu cách hoặc TAB.

Chương 7. Xem thông tin về hệ thống

time

Lệnh *time* dùng để đo khoảng thời gian cần có để thực hiện một chương trình. Nó cũng đo luôn hiệu suất sử dụng CPU và hiển thị các thông tin thống kê khác.

Dùng tùy chọn *-v* để *time* liệt kê chi tiết hơn nữa về hoạt động của một chương trình.

Cách dùng lệnh này:

```
time tên_chương_trình tùy_chọn
```

/proc

Các tập tin trong thư mục */proc* (thông tin về tiến trình) chứa nhiều thông tin về hệ tập tin. Ta có thể coi đây là một cửa sổ chứa thông tin mà hệ nhân sử dụng. Ví dụ, lệnh

```
cat /proc/cpuinfo
```

sẽ liệt kê thông tin về CPU, hoặc lệnh

```
less /proc/modules
```

sẽ liệt kê thông tin về các mô-đun được nạp vào hệ nhân trong quá trình khởi động.

dmesg

Đây là một lệnh rất quan trọng, in ra bản ghi các sự kiện diễn ra trong quá trình khởi động hệ thống. Nó in ra tất cả các thông báo mà hệ nhân gửi ra màn hình khi ta khởi động hệ thống.

Thông thường, để tiện theo dõi, ta hay kết hợp *dmesg* với *less* như sau:

```
dmesg | less
```

df

Hiển thị thông tin về dung lượng của các phân vùng đĩa cứng cũng như các thiết bị lưu trữ đang được gắn kết. Dùng tùy chọn *-h* để yêu cầu *df* hiển thị thông tin ở dạng “dễ đọc”, tức là quy 1024 kilobyte thành ra 1M và hiển thị dưới dạng 1M.

Cú pháp lệnh:

```
df -tuỳ_chọn /dev/sdx
```

Nếu ta không cung cấp thiết bị cần xem dung lượng, *df* sẽ in ra dung lượng của tất cả các thiết bị đã gắn kết trong hệ thống.

who

Lệnh này cho biết người dùng nào đã đăng nhập vào trong hệ thống, và thời gian họ đăng nhập. Cú pháp:

```
who
```

w

Lệnh này cho biết thông tin về người dùng nào đã đăng nhập và họ đang làm gì (các tiến trình nào được họ chạy). Nó gần giống như lệnh *who* nhưng cung cấp thêm vài thông tin khác. Cú pháp:

```
w
```

users

Lệnh này cũng tương tự như *who* nhưng chỉ in ra tên người dùng đang có mặt trên hệ thống. Cú pháp:

```
users
```

last

Hiển thị bản ghi thông tin về việc đăng nhập và đăng xuất của người dùng, cùng với thời gian hệ thống khởi động lại. Cú pháp:

```
last
```

lastlog

Hiển thị danh sách người dùng cùng với thời gian họ đăng nhập vào trong hệ thống. Cú pháp:

```
lastlog
```

whoami

Báo cho người sử dụng biết tên đăng nhập mà họ đã dùng là gì, trong trường hợp người sử dụng dùng lệnh *su* để chuyển tài khoản. Cú pháp:

```
whoami
```

free

Cung cấp thông tin bộ nhớ chính (RAM tổng cộng, còn trống, đã dùng, kích cỡ bộ đệm và phân vùng trao đổi). Dùng với tùy chọn *-t* để tổng hợp các thông số lại, và dùng tùy chọn *-m* để dùng đơn vị đo là MB. Ví dụ:

```
free -tm
```

sẽ hiển thị thông tin bộ nhớ tính bằng MB, và tổng dung lượng bộ nhớ của mỗi thành phần.

uptime

Trả về thời gian hệ thống hoạt động, tính tới thời điểm này. Nó cũng cung cấp thông tin về số người dùng và các công việc đã xử lý (mức độ làm việc của CPU).

◆ Lệnh *w*

Lệnh *w* cũng in ra thông tin về thời gian máy đã chạy, giống như *uptime*.

uname

Liệt kê các thông tin của hệ thống, như hệ điều hành, phiên bản hạt nhân... Một số tùy chọn của *uname*:

- *-a* : in ra tất cả các thông tin.
- *-m* : in thông tin về cấu trúc máy tính (PPC, i386, i686...)

- -n : in tên máy dùng trong mạng.
- -r : in ra phiên bản của hệ nhân đang dùng.
- -s : in ra tên hệ điều hành.
- -p : in tên bộ xử lý.

Cú pháp lệnh:

```
uname -tùy_chọn
```

xargs

Đây là một lệnh tương đối phức tạp, khó dùng. Nó cho phép ta chạy một lệnh khác bao nhiêu lần cũng được. Khi ta gõ một lệnh nào đó, rồi thêm vào phía sau "*| xargs lệnh2*" thì đối với mỗi một dòng đầu ra của lệnh 1, ta sẽ chạy lệnh 2 và dùng dòng đó làm đối số. Ví dụ về cách dùng *xargs*:

```
ls | xargs grep work
```

Ở câu lệnh trên, đối với mỗi dòng mà lệnh *ls* đưa ra tương ứng với một tập tin, BASH sẽ chạy lệnh *grep* để tìm trong tập tin đó chuỗi "work" và in ra màn hình. Đầu ra sẽ có dạng

```
tên_tập_tin: kết_quả_grep_trả_về
```

xargs có một vài tùy chọn như sau:

- -nx : nhóm x lệnh lại với nhau.
- -lx : *xargs* sẽ chạy lệnh mỗi khi nhận được x dòng dữ liệu đầu vào.
- -p : hỏi xem có chạy đầu vào này không.
- -t : in ra các lệnh sẽ chạy trước khi thực hiện.
- -i : giống như tùy chọn *-exec* trong lệnh *find*, thực hiện một lệnh nào đó.

Ví dụ:

```
ls dir1 | xargs -i mv dir1/{}' dir2/{}'
```

Lệnh trên di chuyển tất cả các tập tin trong thư mục *dir1* sang thư mục *dir2*.

```
ls *.wav | xargs -i lame -h '{}' '{}'.mp3
```

Lệnh trên dùng bộ giải mã *lame* để chuyển đổi tất cả các tập tin âm thanh định dạng wave sang định dạng mp3. Tuy nhiên, nó không bỏ đuôi *.wav* đi mà chỉ thêm đuôi *.mp3* vào cuối tập tin.

7.1. Ngày giờ, thời gian và lịch

date

In ra ngày giờ hiện tại, hoặc cho phép bạn đặt lại ngày giờ trên hệ thống.

Để đặt lại ngày, gõ *date MM:DD:YYYY* trong đó MM là tháng, DD là ngày và YYYY là năm. Ví dụ:

```
date 04:25:2008
```

sẽ đặt lại ngày hiện tại là ngày 25/4/2008. Để đặt giờ, ta dùng cú pháp *date -s hh:mm:ss* như trong ví dụ:

```
date -s 03:08:22
```

Lệnh trên đặt giờ về 3 giờ 8 phút 22 giây, buổi sáng.

Ngoài ra, ta cũng có thể dùng tùy chọn *-d "string"* hoặc *-date="string"* để in ra ngày cách đây x hôm hoặc sau đây x hôm. Ví dụ:

```
date -date="3 months 1 day ago"
```

in ra ngày tháng trước đây 3 tháng 1 ngày, và lệnh

```
date -d "3 days"
```

in ra ngày sau đây 3 ngày.

cal

In lịch ra màn hình, với một số tùy chọn như sau:

```
cal -y year
```

In ra lịch của năm year, ví dụ, 2008 chẳng hạn:

```
cal -y 2008
```

7.2. Thông tin về các phân vùng

Có nhiều cách để xem thông tin về các phân vùng đĩa cứng trên máy, ngoài cách dùng *df* như đã đề cập ở trên.

Dùng thư mục /proc

Ta có thể biết thông tin về phân vùng trong phần tương ứng của hệ tập tin *proc*, trong thư mục */proc/ide* hoặc */proc/ide?/hd?*, trong đó dấu hỏi đầu tiên là một con số và dấu thứ 2 là một chữ cái (bắt đầu bằng 'a').

Ví dụ:

```
cd /proc/ide0/hda
```

Trong thư mục này là rất nhiều thông tin về ổ cứng và đĩa CD đã gắn kết với hệ thống.

fdisk

Dùng lệnh *fdisk -l* để liệt kê các thông tin về ổ cứng có trên máy và các phân vùng trên ổ cứng đó.

> Quyền quản trị

Để chạy lệnh này, ta phải đăng nhập với tư cách người dùng *root*.

Chương 8. Điều khiển hệ thống

Chương này dành ra để đề cập tới các lệnh mà bạn sẽ dùng để tương tác với các thiết bị, tiến trình và các trình nền trên hệ thống của mình.

8.1. Gắn kết và tháo gắn kết

◆ Cho phép người dùng gắn kết các thiết bị

Theo mặc định, người dùng bình thường có quyền tháo gắn kết các thiết bị lưu trữ trên hệ thống. Tuy nhiên, nếu muốn người dùng bình thường có thể gắn kết các thiết bị vào hệ thống, người quản trị phải cấp quyền đó cho người dùng.

Những lệnh sau đây sẽ không làm việc nếu người dùng không được cấp quyền gắn kết thiết bị.

Nếu bạn phân phối Linux mà bạn dùng không cho phép người dùng gắn kết thiết bị lưu trữ, ta có thể sửa đổi lại việc này bằng cách đăng nhập vào tài khoản *root* và sửa lại tập tin */etc/fstab* như sau:

```
Thay thế từ "defaults" bằng "user" hoặc thêm từ "user" vào cuối danh sách tùy chọn tương ứng với thiết bị lưu trữ mà ta muốn người dùng thông thường được quyền gắn kết.
```

eject

Lệnh này dùng để mở ổ đĩa CD hoặc DVD. Ví dụ, trên máy bạn, ổ CD được quy chiếu thành tập tin */dev/cdrom*, ta sẽ gõ lệnh sau khi cần lấy đĩa ra khỏi ổ:

```
eject /dev/cdrom
```

mount

Lệnh *mount* dùng để gắn kết một thiết bị lưu trữ. Trước khi truy cập vào dữ liệu được lưu trong một thiết bị, bạn phải gắn kết nó trước. Ví dụ về lệnh *mount*:

```
mount -t ext3 /dev/fd0 /mnt/floppy
```

Gắn kết ổ đĩa mềm, được định dạng bằng hệ tập tin *ext3*, được quy chiếu bằng tập tin */dev/fd0*, vào thư mục */mnt/floppy* trên hệ tập tin. Ta có thể truy cập vào nội dung đĩa mềm bằng cách chuyển đến thư mục */mnt/floppy*.

```
mount -t iso9660 /dev/hdb /mnt/cdrom
```

Gắn kết ổ đĩa CD, nằm ở giao tiếp IDE Secondary Master, vào thư mục */mnt/cdrom*. Trên các bản phân phối hiện nay, thông thường */dev/cdrom* được liên kết tới tập tin tương ứng với ổ CD. Do vậy, thông thường ta không cần biết chính xác ổ CD được gắn vào giao tiếp IDE nào, mà chỉ cần thay */dev/hdx* bằng */dev/cdrom*.

```
mount -t iso /tmp/image_file /mnt/iso_file/ -o loop
```

Lệnh *mount* còn có thể gắn kết tập tin ảnh đĩa CD (*.iso) để ta xem và thay đổi các tập tin trên đó.

➤ Tùy chọn -t

Đối với phiên bản nhân Linux hiện nay, ta không cần phải nhập tùy chọn *-t* để xác định kiểu hệ tập tin cần gắn kết nữa!

umount

Lệnh *umount* dùng để tháo gắn kết cho thiết bị. Trước khi lấy đĩa mềm, đĩa CD/DVD hay thẻ nhớ USB ra khỏi cổng, ổ đĩa, bạn phải tháo gắn kết cho thiết bị ấy trước. Cú pháp:

```
umount /thư/mục/gắn/kết
```

Ví dụ, nếu trước đây ta gắn kết ổ CD vào thư mục */mnt/cdrom*, ta có thể chạy lệnh *umount /mnt/cdrom* để tháo gắn kết cho ổ CD.

smbmount

Lệnh *smbmount* nằm trong gói *samba* cho phép gắn kết một ổ đĩa FAT hay NTFS trên hệ điều hành Windows thông qua mạng LAN hoặc mạng Internet. Cú pháp câu lệnh:

```
smbmount //computer/c /mnt/win
```

Trong đó ta phải thay "computer" bằng tên máy hoặc địa chỉ IP của máy Windows mà ta muốn kết nối, và "c" là tên ổ đĩa hoặc thư mục trên đó.

➤ Lưu ý

Tập tin */etc/hosts* liệt kê địa chỉ IP và tên máy tương ứng với một máy tính trên mạng cục bộ. Nếu trong tập tin không có chứa thông tin về tên máy và địa chỉ IP ứng với nó, ta không thể dùng các lệnh *ping* hay *smbmount* hay *ssh* để truy cập vào máy tính thông qua tên máy được, mà phải cung cấp địa chỉ IP chính xác của máy đó. Ví dụ, trong tập tin */etc/hosts* có dòng:

```
192.168.1.100 my_brother
```

ta có thể dùng lệnh

```
smbmount //my_brother/c /mnt/winc
```

nhưng nếu không có dòng trên trong */etc/hosts*, ta sẽ phải gõ lại như sau:

```
smbmount //192.168.1.100/c /mnt/winc
```

8.2. Tắt và khởi động lại hệ thống

shutdown now

Tắt máy ngay lập tức nhưng không tắt điện. Lưu ý rằng trên hệ thống UNIX, lệnh này sẽ chuyển hệ thống từ chế độ đa người dùng thành chế độ đơn người dùng, trong đó chỉ có người quản trị (root) mới có thể truy cập và sửa đổi các thông số trên hệ thống. Lệnh này thường được dùng khi cần sửa chữa hay nâng cấp hệ thống.

shutdown -h now

Tắt máy và tắt điện ngay lập tức. Ta có thể thêm một thông điệp trong dấu ngoặc kép ở cuối câu lệnh để báo cho tất cả những người dùng khác đang ở trên hệ thống biết. Ví dụ:

```
shutdown -h now "Chú ý: Tắt máy để bảo dưỡng"
```

Lệnh trên sẽ tắt máy, và thông báo "Chú ý: Tắt máy để bảo dưỡng" sẽ được hiển thị trên dòng lệnh của những người dùng khác.

◆ Đặt lịch tắt máy

Ta có thể thay "now" bằng "+x minutes" hoặc "hh:mm" để yêu cầu tắt máy trong x phút nữa, hoặc tắt máy lúc hh:mm. Ví dụ:

```
shutdown -h 11:50
```

➤ shutdown -h và poweroff

Nhiều khi lệnh *shutdown -h* và *halt* không tắt hẳn nguồn điện trên máy tính (bạn phải nhấn công tắc của máy thêm lần nữa!). Lúc đó, thay vì dùng lệnh *halt* hoặc *shutdown -h* ta dùng lệnh *poweroff*.

halt

Tương đương với lệnh *shutdown -h now*. Lệnh này không lấy tham số nào khác.

shutdown -r now

Lệnh này khởi động lại hệ thống ngay lập tức. Giống như *shutdown -h* ta có thể đặt thời gian khởi động lại, cũng như thông điệp gửi tới các người dùng khác đang làm việc trên hệ thống.

reboot

Giống như lệnh *shutdown -r now*. Lệnh này cũng không lấy tham số nào khác.

CTRL-ALT-DEL

Trên giao diện dòng lệnh hệ vờ, tổ hợp phím này sẽ buộc hệ thống khởi động lại ngay lập tức, ngay cả khi người dùng chưa đăng nhập vào hệ thống.

◆ Thay đổi chức năng ALT-CTRL-DEL

Để tắt chức năng khởi động lại máy của tổ hợp ALT-CTRL-DEL, ta có thể sửa tập tin */etc/inittab* với tư cách người dùng *root*:

```
# Trap
CTRL-ALT-DEL
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
```

Ta thêm dấu # vào trước dòng *ca::ctrlaltdel:/sbin/shutdown -t3 -r now* để chuyển dòng đó thành dòng chú thích, và đồng thời tắt chức

năng khởi động lại máy của tổ hợp ALT-CTRL-DEL.

8.3. Điều khiển tiến trình

ps

Liệt kê các tiến trình đang chạy trên hệ thống. Khi chạy *ps* và không cung cấp tham số nào phía sau, BASH sẽ liệt kê các tiến trình người dùng hiện tại đang chạy và đang làm việc trên đầu cuối điều khiển. Các tùy chọn bổ sung:

- *-aux* : liệt kê tất cả các tiến trình đang chạy (bởi tất cả các người dùng)
- *-a* : liệt kê tất cả các tiến trình của tất cả các người dùng.
- *-u username* : liệt kê các tiến trình, kèm theo tên người dùng, hiệu suất CPU và không gian nhớ mà tiến trình sử dụng.
- *-x* : liệt kê các tiến trình đang chạy nhưng không thuộc đầu cuối nào (các ứng dụng X chẳng hạn!)
- *-l* : liệt kê các thông tin chi tiết cho mỗi tiến trình.
- *--forest* : liệt kê các tiến trình đang chạy dưới dạng cây, để bạn biết chính xác mối tương quan giữa chúng.

pstree

Liệt kê các tiến trình dưới dạng cây, tương tự như lệnh *tree* để liệt kê cấu trúc cây thư mục vậy. Dùng thêm tùy chọn *-p* để chương trình hiển thị cả ID của tiến trình.

```
pstree -p
```

pgrep

Giống như lệnh *grep*, lệnh *pgrep* giúp ta tìm ra ID của một tiến trình khi ta biết một phần trong tên của tiến trình đó. Dùng tùy chọn *-l* ta có thể liệt kê thêm tên của tiến trình, và *-u* để tìm kiếm dựa trên người dùng chạy tiến trình đó. Thông thường, *pgrep* sẽ chỉ trả về số định danh tiến trình pid. Ví dụ:

```
kill $(pgrep mozilla)
```

Lệnh trên sẽ buộc tiến trình có tên chứa mozilla (như mozilla_firefox hay mozilla_thunderbird chẳng hạn) kết thúc.

```
pgrep -l tên_tiến_trình
```

Lệnh trên liệt kê cả ID của tiến trình và tên của nó.

top

Lệnh *top* cho biết những tiến trình nào trên hệ thống đang sử dụng nhiều hiệu năng CPU nhất. Ngoài ra, *top* còn có rất nhiều thông tin hữu ích khác để bạn giám sát hệ thống.

kill

Để tắt một tiến trình, ta phải biết id hoặc pid của nó. Cách dùng *kill* như sau:

```
kill pid
```

Dùng *ps* và *pstree* để biết tiến trình cần tắt có pid là gì.

```
kill %id
```

Dùng lệnh *jobs* để biết tiến trình cần tắt có id là gì.

```
kill -kill pid
```

Buộc tắt một tiến trình theo pid. Toàn bộ dữ liệu chưa lưu trên tiến trình này sẽ bị mất. Ngoài ra, còn có rất nhiều tùy chọn khác mà ta có thể tham khảo trên trang hướng dẫn của *kill*.

killall

Không giống như *kill* tắt tiến trình có pid và id nào đó, *killall* tắt một tiến trình có tên chính xác được nhập vào. Ví dụ, để tắt tiến trình *mozilla*, ta gõ:

```
killall mozilla
```

Lệnh *killall* còn có tùy chọn *-v* để thông báo lại cho người dùng biết việc tắt tiến trình thành công hay không, và tùy chọn *-i* để hỏi lại trước khi tắt tiến trình.

pkill

pkill tắt tiến trình có tên khớp với biểu thức chính quy cung cấp cho nó. Dùng với tùy chọn *-u* để tắt tiến trình theo tên và người dùng. Ví dụ, ta muốn tắt tiến trình tên là *mozilla-bin*. Với *killall*, ta phải gõ chính xác tên tiến trình:

```
killall mozilla-bin
```

nhưng với *pkill*, ta chỉ cần gõ

```
pkill mozilla
```

Hơn nữa, nếu chỉ tắt tiến trình *mozilla-bin* của 2 người dùng tên là *fred* và *anon*, ta gõ:

```
pkill -u fred anon mozilla
```

skill

Gửi một tín hiệu tới người dùng, lệnh hoặc giao diện đầu cuối nào đó. *skill* có nhiều tùy chọn như sau:

- *-L* : liệt kê các tín hiệu có thể gửi được.
- *-u username* : đặt tên người dùng cần gửi tín hiệu tới. Nếu cần gửi tín hiệu tới nhiều người dùng, hãy viết danh sách người dùng, ngăn cách nhau bằng dấu cách.
- *-p pid* : pid của tiến trình cần gửi tín hiệu tới.
- *-c command* : tên lệnh (giống như *killall*)
 - *-t tty*: tên giao diện đầu cuối (tty)
 - *-v* : thông báo lại tiến độ thực thi câu lệnh.
 - *-i* : hỏi lại mỗi khi gửi tín hiệu.

Ví dụ:

```
skill -STOP fred anon
```

Lệnh trên gửi tín hiệu *STOP* để tắt tất cả các tiến trình do người dùng *fred* và *anon* chạy. Giao diện dòng lệnh của *fred* sẽ bị ngưng lại cho tới khi ta gõ lệnh:

```
skill -CONT fred
```

Lưu ý rằng bạn phải đăng nhập với tư cách quản trị hệ thống.

CTRL-C

Tổ hợp phím này dùng để dừng công việc đang chạy lại. Ví dụ, khi đang chạy lệnh *ls* nếu đầu ra quá nhiều, ta có thể nhấn **CTRL-C** để ngừng việc *ls* xuất dữ liệu ra màn hình lại.

jobs

In ra các công việc đang chạy trên hệ vỏ, cùng với id của tiến trình tương ứng.

bg

Đưa một tiến trình về chạy ở chế độ nền. Để chạy một chương trình nào đó ở chế độ nền, ta thêm dấu "&" vào cuối câu lệnh. Ta có thể dùng tổ hợp **CTRL-Z** để tạm ngưng một tác vụ nào lại, và đưa tiến trình tương ứng vào nền, và dùng lệnh *bg* để chạy lại tác vụ đó trong nền. Cú pháp lệnh:

```
bg 4
```

để đưa tiến trình mang id là 4 vào chạy nền, và

```
bg job_name
```

để chạy tiến trình mang tên *job_name* vào chạy nền.

fg

Đưa tiến trình từ nền ra trước để ta có thể tương tác với nó. Tiến trình sẽ được đưa lên giao diện đầu cuối ta đang chạy. Dùng *fg* không kèm theo tham số nào khác sẽ đưa tiến trình được đưa vào nền gần đây nhất ra phía trước. Cú pháp lệnh của *fg* tương tự như *bg*:

```
fg id
fg tên_việc
```

nice

Đặt mức ưu tiên cho một tiến trình. *nice -20* đưa một tiến trình lên mức ưu tiên cao nhất (chỉ có người quản trị hệ thống *root* mới có quyền đặt mức ưu tiên âm cho tiến trình). *nice 20* đặt mức ưu tiên thấp nhất cho một tiến trình. Người dùng bình thường chỉ có thể hạ mức ưu tiên cho tiến trình của chạy.

Ví dụ:

```
nice -20 make
```

Lệnh trên chạy lệnh *make* với mức ưu tiên cao nhất.

renice

Thay đổi mức ưu tiên của tiến trình. *renice* được dùng với tùy chọn *-u anón* để đổi mức ưu tiên của tất cả các tiến trình do người dùng tên *anón* chạy, và *-g xyz* để áp dụng mức ưu tiên mới cho tiến trình do nhóm *xyz* chạy. Theo mặc định, *renice* nhận đối số là mức ưu tiên mới và pid của tiến trình. Ví dụ:

```
renice +20 2222
```

Câu lệnh trên chuyển tiến trình mang số pid là 2222 xuống mức ưu tiên thấp nhất.

snice

Thay đổi mức ưu tiên của tiến trình theo phương thức tương tự như *skill*. Ý nghĩa các tùy chọn:

- *-u username* : đổi mức ưu tiên cho các tiến trình do một hoặc nhiều người dùng nào đó. Nếu cần sửa đổi tiến trình của nhiều người dùng, ta nhập tên người dùng, phân cách bằng dấu cách, vào sau *-u*.
 - *-p pid*: đổi mức ưu tiên dựa vào pid của tiến trình.
 - *-c command*: đổi mức ưu tiên dựa vào tên tiến trình.
 - *-t tty*: đổi mức ưu tiên cho tất cả các tiến trình nằm trên đầu cuối nào đó.
 - *-v* : thông báo việc đổi tiến trình thành công hay thất bại.
 - *-i* : yêu cầu người dùng xác nhận trước khi đổi mức ưu tiên cho 1 tiến trình.

Ví dụ:

```
snice -10 -u root
```

Đặt lại mức ưu tiên là -10 cho tất cả các tiến trình mà người quản trị hệ thống chạy.

8.4. Điều khiển các dịch vụ

Các hệ thống UNIX dùng những văn lệnh để điều khiển “trình nền” (daemon), cung cấp cho người dùng một dịch vụ nào đó. Có rất nhiều dịch vụ trên hệ thống UNIX. Trình nền là một tiến trình hệ thống chạy ở nền (không cho phép người dùng tương tác với nó) và thực hiện một tác vụ nhất định. Các trình nền thường có chữ “d” ở cuối tên, và bắt các sự kiện xảy ra trên hệ thống để thực hiện một chức năng nào đó. Ví dụ, *sshd* nghe các yêu cầu kiểu hệ vỏ bảo mật (secure shell) một máy khác gửi tới, yêu cầu truy cập vào máy ta, và xử lý các thao tác quản trị từ xa. Thông thường các trình nền nhận trách nhiệm xử lý những tác vụ quan trọng trên hệ thống, như điều khiển không gian trao đổi trên đĩa cứng, quản lý bộ nhớ và tần số làm việc của CPU.

service

Lệnh *service* là một văn lệnh hệ vỏ trên Redhat và Mandrake/Mandriva Linux, để bật, tắt, khởi động lại hoặc in thông tin trạng thái cho một trình nền trên hệ thống. Nó có các tùy chọn sau:

- -s : in ra trạng thái hiện tại của tất cả các trình nền có trên hệ thống.
- -f : khởi động lại một trình nền nào đó, được cung cấp ở cuối câu lệnh.
- -R : khởi động lại tất cả các trình nền (lưu ý rằng lệnh `service -R` sẽ khởi động lại cả giao diện đồ họa X).

Ví dụ, để khởi động lại trình nền `sshd`, ta gõ:

```
service -f sshd
```

Chạy văn lệnh của trình nền

Thông thường, các trình nền đều đặt văn lệnh hệ vỏ điều khiển của mình trong thư mục `/etc/init.d`. Ta chỉ cần chuyển đến thư mục đó, và chạy văn lệnh tương ứng để bật hay tắt một trình nền nào đó. Các đối số thường dùng:

- restart : khởi động lại trình nền.
- start : kích hoạt trình nền.
- stop : tắt trình nền.
- status : hiển thị các thông tin về trình nền.

Ví dụ, ta có thể chạy lệnh sau để khởi động lại trình nền `sshd`:

```
/etc/init.d/ssh status
```

Chương 9. Quản lý người dùng

SU

Lệnh `su` được dùng để đổi người dùng. Gõ `su -` để chuyển sang tài khoản `root` hoặc gõ `su fred` để chuyển sang tài khoản người dùng tên là `fred`.

Lưu ý rằng khi muốn quay về tài khoản cũ của mình, thay vì dùng lệnh `su` một lần nữa, ta chỉ cần gõ `exit` hoặc nhấn **CTRL-D**.

Nếu chỉ gõ `su` thì bạn cũng sẽ thực hiện được một số tác vụ với tư cách quản trị hệ thống, nhưng các biến môi trường thì vẫn giữ nguyên như của người dùng thông thường, nên tốt nhất, nếu cần làm việc với tư cách `root` thì ta dùng lệnh `su -`.

◆ *sudo*

Thông thường ta không hay chuyển hẳn về người dùng `root` mà chỉ thực hiện một số tác vụ nào đó với tư cách `root` bằng lệnh `sudo`. Lệnh `sudo` còn có thể lưu lại các hành động bạn đã thực hiện trong bản ghi hệ thống, để sau này bạn có thể theo dõi lại mà không cần đăng nhập vào tư cách `root`.

Cách dùng `sudo` rất đơn giản: chỉ việc thêm `sudo` phía trước câu lệnh thông thường bạn cần thực hiện. Ví dụ, để cài đặt một gói rpm trên RedHat Linux, ta gõ:

```
sudo rpm -U myrpm.i386.rpm
```

9.1. Người dùng và nhóm

Tất cả các thông tin của người dùng được liệt kê trong tập tin `/etc/passwd`, và thông tin về nhóm được lưu trong `/etc/groups`. Nếu bạn phải sửa đổi lại thông tin trong hai tập tin trên, hãy dùng `vipw` để sửa lại tập tin `/etc/passwd` và `vigr` để sửa lại `/etc/groups`. Hai lệnh đặc biệt này tự động thực hiện việc khoá truy nhập đối với các tập tin trước và sau khi ta sửa lại nội dung của chúng.

chsh

Lệnh `chsh` dùng để đổi hệ vỏ mà bạn dùng để đăng nhập. Để xem có những hệ vỏ nào bạn có thể dùng, gõ:

```
chsh --list-shells
```

Sau đó, bạn gõ `chsh` rồi nhấn ENTER, và nhập vào tên hệ vỏ muốn dùng từ lần đăng nhập sau.

chfn

Thay đổi thông tin cá nhân như tên thật, địa chỉ, số điện thoại... về người dùng, được lưu trong tập tin `/etc/passwd`. Người quản trị có thể nhập thêm tên người dùng cần sửa đổi thông tin để sửa lại thông tin cá nhân cho người dùng đó. Cú pháp:

```
chfn tên_người_dùng
```

passwd

Đổi mật khẩu đăng nhập cho người dùng hiện tại. Người quản trị có thể nhập vào sau tên người dùng cần được đổi mật khẩu để chỉ định mật khẩu mới cho người đó, trong trường hợp người dùng quên mật khẩu. Cú pháp:

```
passwd tên_người_dùng
```

Chương 10. Xử lý văn bản

Đây là chương lớn nhất trong tài liệu này, vì trên GNU/Linux, ta rất hay phải thực hiện các tác vụ liên quan đến văn bản. Chương này sẽ đề cập đến một số trình soạn thảo văn bản hay dùng, và các lệnh xem cũng như xử lý văn bản hữu ích...

10.1. Soạn thảo văn bản

vim

Đây là một trình soạn thảo văn bản trên giao diện dòng lệnh, bắt nguồn từ trình soạn thảo nổi tiếng vi, có trong hầu hết các hệ thống UNIX hiện đại. Vi tốn rất ít tài nguyên hệ thống, và có rất nhiều chức năng soạn thảo văn bản, nhưng yêu cầu người dùng phải học một số tổ hợp phím và từ khóa. Cú pháp lệnh:

```
vim file1 file2
```

Để xem trợ giúp trên vim, ta gõ `:help` . Trên VIM có 2 chế độ: chế độ soạn thảo và chế độ xem. Muốn chuyển từ chế độ xem sang chế độ soạn thảo, ta có thể nhấn **i** (soạn thảo tại vị trí con trỏ) hoặc **a** (thêm văn bản vào sau vị trí con trỏ). Muốn quay lại chế độ xem, nhấn **ESC**. Muốn tìm kiếm, nhấn `/` sau đó gõ vào từ cần tìm. Để lưu tập tin, nhấn `:w` , và thoát nhấn `:q` . Để thoát và không lưu, gõ `:q!` .

emacs

Emacs là một công cụ duyệt web, chat, chơi trò chơi và cả soạn thảo văn bản. Nó có rất nhiều chức năng, và do vậy khá đồ sộ.

nano

Nano cũng là một trình soạn thảo văn bản trên dòng lệnh rất nổi tiếng. Nó cực kỳ nhỏ gọn, và so với vim thì dễ dùng hơn rất nhiều. Đặc biệt, nano luôn có tra cứu phím tắt ngay bên dưới, nên làm việc với nano rất thuận tiện và dễ dàng. Lưu ý ký hiệu `^` dùng để chỉ nhấn phím **CTRL**. Ví dụ, `^O` sẽ tương đương với tổ hợp **CTRL-O**.

10.2. Xem văn bản

head

```
head file.txt
```

Lệnh trên sẽ in ra 10 dòng đầu tiên của tập tin `file.txt`.

```
head -n 25 file.txt
```

Lệnh trên sẽ in ra 25 dòng đầu tiên của tập tin `file.txt`

```
head -F -n 25 file.txt
```

Lệnh trên in ra 25 dòng đầu của tập tin *file.txt*, nhưng nội dung tập tin sẽ được cập nhật liên tục. Nếu muốn thoát khỏi chế độ tương tác này, bạn phải nhấn tổ hợp **CTRL-C**.

tail

Tương tự như *head*, nhưng thay vì in ra một vài dòng đầu tiên của tập tin, *tail* in ra những dòng cuối. Cách dùng:

```
tail -n 20 file.txt
tail file.txt
tail -F -n 30 file.txt
```

less

Đây là trình xem văn bản, có khả năng cuộn lên (**PageUp**) và xuống (**PageDown**), mở cùng lúc nhiều tập tin. Khi xem nhiều tập tin cùng lúc, gõ `:"n"` và `:"p"` để chuyển tới tập tin kế tiếp hoặc tập tin trước. Để tìm một từ cố trong văn bản, gõ `"/word"`, và để chuyển tới kết quả kế tiếp tìm được bằng cách nhấn phím **n**. Cách dùng:

```
less file1.txt file2.txt
```

Để mở 2 tập tin *file1.txt* và *file2.txt* cùng lúc.

```
cat file.txt | less
```

Để đưa đầu ra của lệnh *cat* sang đầu vào của lệnh *less*.

cat

Ghép nhiều tập tin văn bản lại thành một, hoặc in ra nội dung của một tập tin. Một số tùy chỉnh của lệnh *cat* như sau:

- `-b` : đánh số tất cả các dòng không trống.
- `-n` : đánh số tất cả các dòng (kể cả dòng trống).

Ví dụ:

```
cat filepart1 filepart2 filepart3 filepart4 > file.txt
```

Lệnh trên ghép lần lượt nội dung của 4 tập tin theo thứ tự từ trái sang phải, và ghi vào tập tin *file.txt*.

tac

Chức năng tương tự như *cat* nhưng đảo ngược thứ tự các dòng trong văn bản gốc, trước khi nối lại theo thứ tự ghi trong câu lệnh. Ví dụ:

```
tac filepart1 filepart2 filepart3 filepart4 > file.txt
```

Lệnh trên sẽ in dòng cuối cùng trong tập tin *filepart1* lên đầu, và in dòng đầu thành dòng cuối, sau đó nối tiếp với dòng cuối của tập tin *filepart2*... dòng cuối cùng được ghi vào trong *file.txt* sẽ là dòng đầu tiên của tập tin *filepart4*.

10.3. Thông tin về văn bản

wc

Lệnh *wc* đếm số từ, số dòng và số byte của một tập tin văn bản. Dùng tùy chọn `-w` để in ra số từ, `-l` để in số dòng, `-c` để in số byte. Khi không chỉ định tùy chọn nào, *wc* sẽ in ra cả 3 thông tin trên. Cú pháp lệnh:

```
wc -tùy_chọn tập_tin.txt
```

cmp

So sánh 2 tập tin để xem chúng có giống nhau hay không. Lệnh *cmp* tương tự như lệnh *diff* nhưng làm việc được với mọi loại tập tin, kể cả các tập tin nhị phân.

diff

So sánh hai tập tin và in ra những chỗ khác nhau. Lệnh *diff* thường được dùng để tạo bản vá, dùng trong nâng cấp phần mềm. Ví dụ:

```
diff file1.txt file2.txt
```

Lệnh trên sẽ in ra dấu '>' kèm theo dòng nằm trong tập tin thứ nhất, nhưng không có trong tập tin thứ hai, và sẽ in dấu '<' kèm theo dòng nằm trong tập tin thứ hai, nhưng không có trong tập tin thứ nhất.

sdiff

Thay vì in ra sự khác biệt theo từng dòng, lệnh *sdiff* in các tập tin thành 2 cột kế nhau, ngăn cách bởi dấu cách.

diff3

Giống như *diff* nhưng lấy 3 tập tin đầu vào để so sánh.

comm

So sánh hai tập tin và in ra 3 cột, cột thứ nhất chứa các dòng chỉ có ở tập tin đầu, cột thứ hai chứa các dòng chỉ có ở tập tin thứ hai, và dòng thứ ba chứa các dòng chung của hai tập tin. Dùng tùy chọn *-1*, *-2* hoặc *-3* để bỏ qua một hoặc hai cột. Cú pháp:

```
comm file1 file2
```

look

In ra một danh sách các từ có trong từ điển hệ thống, bắt đầu bằng một chuỗi nào đó. Lệnh này hay được dùng để tìm ra các chuỗi bắt đầu với một cụm từ hoặc một tiếp đầu ngữ nào đó. Cú pháp lệnh:

```
look chuỗi_cần_tìm
```

10.4. Xử lý văn bản

sort

Sắp xếp lại văn bản theo thứ tự ABC. Thêm tùy chọn *-r* để đảo thứ tự sắp xếp, và thêm *-g* để sắp xếp số (đọc toàn bộ số thay vì chỉ đọc số đầu tiên). Ví dụ:

```
cat shoppinglist.txt | sort
```

Mỗi dòng trong tập tin *shoppinglist.txt* chứa một mặt hàng cần mua, và lệnh trên sẽ sắp xếp lại danh sách đó, và in ra màn hình.

Ngoài ra lệnh *sort* còn có một số tùy chỉnh ít dùng khác:

- *-t ,*: chỉ định dấu phẩy làm ký tự ngăn cách giữa các cột trong tập tin cần sắp xếp.
- *-k 4*: yêu cầu sắp xếp dựa trên cột thứ 4 (cột 1 là cột nằm trước ký tự ngăn cách).

Ví dụ:

```
sort -t : -k 4 -k 1 -g /etc/passwd | more
```

Lệnh trên sắp xếp lại nội dung tập tin */etc/passwd*, với ký tự ngăn cách là dấu ":". Trước hết, cột thứ 4 sẽ được sắp xếp, và sau đó sắp xếp lại lần nữa, dựa trên cột thứ nhất. Tùy chọn *-g* yêu cầu chương trình sắp xếp số; nếu không có tùy chọn *-g*, 4000 sẽ được đưa lên trước 50 (vì 4 < 5).

join

Đọc nội dung hai tập tin và kết hợp hai dòng lại tương ứng lại với nhau khi cả 2 dòng có chứa một từ nào đó. Bỏ qua nếu các dòng không có từ nào chung. Cú pháp:

```
join file1 file2
```

cut

In ra một phần trong mỗi dòng văn bản. Ta có thể dùng lệnh *cut* để trích lấy các nội dung mình cần. Các tùy chọn của lệnh *cut* mà ta có thể sử dụng:

- *-d ':'*: đặt ký tự ngăn cách dùng trong dòng (ở đây dùng dấu hai chấm). Ví dụ, trong tập tin */etc/passwd* (lưu thông tin mật khẩu của những người dùng), ký tự ":" được dùng để ngăn cách các trường. Ký tự ngăn cách mặc định là **TAB**.
- *-c 1-50*: chỉ lấy 50 ký tự đầu mỗi dòng.
- *-f*: xác định cột văn bản chứa nội dung cần lấy ra. Thường, ta dùng *-f* kết hợp với *-d* để trích lấy nội dung mình cần. Ví dụ, ta có dòng văn bản "cột1,cột2,cột3" chứa trong tập tin *file.txt*, nếu muốn in ra "cột1", ta gõ:

```
cut -d ',' -f 1 file.txt
```

Ngoài ra, ta có thể dùng mẫu "-5" để yêu cầu *cut* lấy các dòng từ 1 đến 5, và "20-" để lấy các dòng từ 20 đến hết. Lấy ví dụ về cách dùng *cut*:

```
cut -c 1-20, 30-, 9 file.txt
```

Lệnh trên sẽ lấy 20 ký tự đầu tiên, ký tự thứ 9 và 30 ký tự ở cuối trong tập tin *file.txt*.

ispell/aspell

Đây là hai lệnh dùng để kiểm tra chính tả, và gợi ý các từ có thể dùng để thay thế cho từ bị sai. Ví dụ:

```
ispell file.txt  
aspell file2.txt
```

chcase

Chuyển các chữ hoa trong một tập tin thành các chữ thường. Ta cũng có thể dùng lệnh *tr* để làm việc này. Ví dụ:

```
cat file.txt | tr '[A-Z]' '[a-z]' > file2.txt
```

Lưu ý là lệnh *chcase* là một văn lệnh PERL

fmt

Đây là một công cụ định dạng văn bản đơn giản. Dùng tùy chọn *-u*, ta có thể loại bỏ các khoảng trống thừa giữa hai từ bất kỳ trong văn bản, tức là nếu giữa hai từ có nhiều hơn một dấu cách thì *fmt -u* sẽ sửa lại thành 1 dấu, và khoảng cách giữa các hàng sẽ được chỉnh lại thành 2 dấu cách. Ví dụ:

```
fmt -u file.txt
```

Sẽ đảm bảo giữa 2 từ trong tập tin là 1 dấu cách, và giữa hai câu là 2 dấu cách. Đầu ra sẽ được in ra màn hình.

paste

Nối các dòng trong 2 tập tin lại với nhau, và ngăn cách chúng bằng ký tự TAB hoặc một ký tự do ta đặt. Ví dụ:

```
paste file1.txt file2.txt
```

Sẽ in ra dòng đầu tiên của *file1.txt*, dấu TAB, rồi dòng đầu tiên của *file2.txt*, và cứ như vậy.

```
paste --serial file1.txt file2.txt
```

Sẽ in lần lượt từng dòng của *file1.txt*, rồi tới *file2.txt*, tức là dòng đầu tiên của *file1.txt*, TAB, dòng thứ 2 của *file1.txt*,... cho tới dòng cuối cùng của *file2.txt*.

expand

Lệnh này chuyển ký tự TAB thành các dấu cách. Dùng *-t 4* để thay mỗi phím TAB thành 4 dấu cách. Ví dụ:

```
expand -t 4 file.txt
```

unexpand

Ngược với *expand*, chuyển nhiều dấu cách thành phím TAB.

```
unexpand file.txt
```

uniq

Xoá các dòng trùng lặp có trong một tập tin. Các tùy chọn của *uniq* như sau:

- *-c* : đếm số lần xuất hiện của một dòng.
- *-u* : liệt kê tất cả các dòng duy nhất.
- *-d* : liệt kê tất cả các dòng bị trùng lặp.

Ví dụ:

```
uniq -cd danhbađiệnthoại.txt
```

Lệnh trên sẽ liệt kê và đếm tất cả các nội dung trùng lặp có trong tập tin *danhbađiệnthoại.txt*.

tr

Lệnh *tr* đọc đầu vào chuẩn, thay thế các nội dung trong đó và in ra đầu ra chuẩn. Ví dụ:

```
cat file.txt | tr '3' '5' > new.txt
```

Lệnh trên thay thế toàn bộ các số '3' trong tập tin file.txt thành số '5' và ghi vào tập tin new.txt.

```
cat file.txt | tr '[A-Z]' '[a-z]' > new.txt
```

Lệnh trên chuyển tất cả các chữ hoa thành chữ thường.

nl

Đánh số dòng văn bản, và đẩy văn bản sang phải. Thông thường, ta dùng *cat -b* để đánh số cả các dòng trắng, và *cat -n* để đánh số các dòng có nội dung. Tuy nhiên, với *nl* ta có thể làm nhiều hơn thế:

```
grep keywords file.txt | nl
```

Lệnh trên dùng *grep* để tìm ra từ khoá keywords nằm trong tập tin file.txt, rồi đánh số các kết quả in ra bằng *nl*.

Tìm kiếm và thay thế bằng PERL

Để tìm và thay thế hàng loạt với PERL, ta có thể dùng lệnh sau:

```
perl -pi -e "s/cần_tìm/thay/g;" file*.txt
```

Lệnh trên sẽ tìm trong tất cả các tập tin dạng file*.txt, và thay thế từ "cần_tìm" bằng từ "thay".

◆ awk và sed

Ngoài lệnh *perl -pi -e* và *tr*, trên GNU/Linux còn có hai lệnh rất mạnh để xử lý văn bản là *sed* và *awk*. Tuy nhiên, cách dùng chúng tương đối phức tạp. Nếu quan tâm, bạn có thể tham khảo cuốn *Advanced Bash Scripting Guide* của Mendel Cooper.

10.5. Tìm tập tin chứa văn bản mình cần

grep

Chương trình này tìm một chuỗi văn bản trong tất cả các tập tin bạn chọn. Cú pháp:

```
grep chuỗi tậptin.txt
```

Các tùy chọn thường dùng:

- *-v* : in ra các dòng không chứa chuỗi cần tìm
- *-n* : đánh số các dòng được in ra
- *-w* : *grep* sẽ so khớp chính xác chuỗi được nhập vào
- *-A 5* : in ra 5 dòng sau dòng có chứa chuỗi cần tìm
- *-B 10* : in ra 10 dòng trước dòng có chứa chuỗi cần tìm
- *-r* : tìm đệ quy tất cả các tập tin bên trong những thư mục con của thư mục hiện thời.

Lệnh *grep* cho phép ta nhập một biểu thức chính quy để so khớp. Xin xem thêm phần .

Ví dụ, lệnh sau sẽ tìm trong tập tin "rpmlist.txt" và in ra tất cả các dòng chứa từ khoá "rpm":

```
grep rpm rpmlist.txt
```

Ta cũng có thể dùng *grep* kết hợp với *pipe* để trích lấy nội dung cần xem. Ví dụ, để liệt kê nội dung thư mục */usr/share*, nhưng chỉ in ra các thư mục hoặc tập tin có chứa từ "gnome", ta gõ:

```
ls /usr/share | grep gnome
```

rgrep

Tương tự như *grep -r*, tìm chuỗi hoặc biểu thức chính quy trong tất cả các tập tin và thư mục con của thư mục hiện thời.

fgrep

Lệnh này tương đương với *grep -F*. Khác với *grep* thông thường, cho phép dùng biểu thức chính quy, *fgrep* chỉ hiểu chính xác chuỗi nhập vào. Ví dụ:

```
fgrep 'a$b*' file.txt
```

Lệnh trên sẽ tìm chuỗi "a\$b*" bên trong tập tin *file.txt* và in ra dòng chứa chuỗi đó.

Chương 11. Các lệnh toán học

units

Lệnh *units* giúp ta chuyển đổi đơn vị đo từ centimet sang inch, lit sang gallon... Thường thường, ta chỉ cần chạy lệnh

```
units -v
```

Sau đó, ta nhập vào giá trị cần chuyển đổi, ví dụ như "60 meters" để xem chiều dài đó được quy đổi sang các đơn vị khác như thế nào.

Để thoát khỏi chương trình, nhấn **CTRL-D**.

python

Đây là một ngôn ngữ lập trình mạnh, và cũng có thể giúp ta thực hiện các phép tính phức tạp trên giao diện dòng lệnh. Lệnh *python* mở giao diện Python để ta có thể nhập các phép tính, và sau khi nhập xong, gõ Enter, Python sẽ trả về kết quả tính được.

Lưu ý là khi làm việc với phân số, nếu muốn lấy kết quả số thực, ta phải thêm vào mẫu số dấu chấm và số 0, nếu không Python sẽ trả về kết quả số nguyên. Ví dụ, "1/2+1/3" sẽ cho kết quả là 0 nhưng "1/2.0+1/3.0" sẽ cho kết quả là "0.833333".

Nhấn **CTRL-D** để thoát ra ngoài sau.

numgrep

Lệnh *numgrep* giống như lệnh *grep* nhưng dành riêng cho số. Dùng dấu "/" để đặt biểu thức. Dùng *m<n>* để tìm nhiều số *n* và dùng *f<n>* để tìm thừa số của *n*. Dùng dấu phẩy để ngăn cách các biểu thức, và dấu .. để biểu diễn phạm vi số. Ví dụ, để nhập từ bàn phím, ta có thể gõ:

```
numgrep
```

Để nhập từ một tập tin và tìm số giữa 1 và 1000, ta có thể gõ:

```
numgrep /1..1000/ file.txt
```

➤ Công cụ này nằm trong gói *num-utils*

Công cụ này nằm trong gói *num-utils*. Đây là một gói bao gồm nhiều lệnh phục vụ việc tính toán trên GNU/Linux.

Chương 12. Mạng

netstat

Hiển thị nội dung tập tin */proc/net*, cho biết trạng thái các cổng trên hệ thống (mở, đóng, chờ...) và rất nhiều thông tin khác.

tcpdump

tcpdump bắt các gói tin đi qua một card mạng, và thông dịch nó ra cho ta xem. *tcpdump* nhận ra nhiều định dạng gói tin khác nhau, và cho phép ta lưu các gói tin bắt được để xem xét.

ping

Lệnh *ping* gửi yêu cầu trả lời tới một máy trên mạng và in ra thời gian truyền qua lại. Cú pháp:

```
ping ip_hoặc_tên_máy
```

Để dừng *ping* lại, ta phải nhấn **CTRL-C**.

hostname

Cho người dùng biết tên máy mà họ đang đăng nhập vào.

tracert

Lệnh *tracert* cho biết đường đi của một gói tin. Nó liệt kê ra tất cả các máy mà gói tin đó đi qua để đến một đích nào đó. Cú pháp lệnh:

```
tracert tên_máy_hoặc_ip
```

tracert

Lệnh này tương tự như *tracert* nhưng không có nhiều tùy chọn phức tạp như *tracert*. Cú pháp lệnh:

```
tracert tên_máy_hoặc_ip
```

findsmb

Liệt kê thông tin về các máy trả lời truy vấn tên SMB (ví dụ như các máy chạy Windows có chia sẻ ổ cứng ra mạng). Cú pháp lệnh:

```
findsmb
```

Lệnh này sẽ gửi yêu cầu đến tất cả các máy có thể. Bạn có thể thêm vào địa chỉ subnet để hạn chế việc gửi thông tin đến các máy trong mạng thay vì tất cả.

nmap

Đây là một công cụ cao cấp cho phép gửi gói tin yêu cầu tới một máy để xem nó có hoạt động không, nó đang mở những cổng nào. *nmap* có rất nhiều tùy chọn. Một câu lệnh hay dùng:

```
nmap tên_máy
```

Lệnh trên liệt kê các cổng đang mở trên máy.

12.1. Cấu hình mạng

ifconfig

Lệnh này được dùng để cấu hình hoặc hiển thị cấu hình hiện thời một giao tiếp mạng. Ngoài việc bật và tắt một giao tiếp mạng bằng tham số "up" và "down", lệnh này còn cho phép ta đặt địa chỉ IP của giao tiếp, nếu ta không có văn lệnh *ifconfig* trên máy.

Để liệt kê thông tin về tất cả các giao tiếp mạng đã cấu hình, ta dùng lệnh:

```
ifconfig
```

Để tắt giao tiếp eth0 (tương ứng với card mạng đầu tiên trên máy, trên máy tính cá nhân thì đây chính là card mạng của bạn), để nó không gửi và nhận các gói tin nữa, ta dùng lệnh

```
ifconfig eth0 down
```

Để bật lại eth0, ta gõ

```
ifconfig eth0 up
```

Ngoài ra, ta còn có thể đặt địa chỉ IP cho máy mình bằng cú pháp

```
ifconfig thiết_bị địa_chỉ_ip
```

Ví dụ:

```
ifconfig eth0 192.168.1.4
```

Lệnh trên sẽ gán địa chỉ IP 192.168.1.4 cho card mạng eth0 trên máy.

ifup

Để bật thiết bị mạng eth0 bằng một văn lệnh chứa các thiết lập mặc định mà bạn có, ta dùng cú pháp:

```
ifup eth0
```

ifdown

Để tắt thiết bị mạng eth0, ta dùng cú pháp:

```
ifdown eth0
```

ifcfg

Lệnh *ifcfg* giúp ta cấu hình một giao tiếp mạng. Để xem trợ giúp về cách dùng lệnh này, bạn gõ *ifcfg* và chương trình sẽ in ra thông tin về cách dùng để ta tham khảo.

Để chuyển giao tiếp eth0 từ địa chỉ 192.168.0.1 về 192.168.0.2, ta có thể gõ:

```
ifcfg eth0 del 192.168.0.1
ifcfg eth0 add 192.168.0.2
```

Câu lệnh đầu tiên sẽ tắt eth0 và bỏ địa chỉ IP gán cho nó, và câu lệnh thứ hai sẽ bật lại nó với địa chỉ IP mới.

route

Lệnh *route* được dùng để thay đổi hoặc hiển thị bảng định tuyến. Thường thường, lệnh *route* cho phép ta thêm địa chỉ IP của máy trạm để kết nối lên mạng Internet:

```
route add default gw ip_hoặc_tên_máy
```

Lệnh trên sẽ báo cho hệ thống biết thiết lập mặc định khi muốn gửi gói tin ra ngoài mạng thì gửi nó tới địa chỉ hoặc máy nào.

12.2. Internet

Lưu ý rằng bạn phải cấu hình DNS hợp lý bằng cách sửa lại tập tin “/etc/resolv.conf” để truy cập lên Internet...

host

Tra cứu tên miền tương ứng với địa chỉ Internet thông qua DNS. Cú pháp:

```
host địa_chỉ_ip
```

hoặc

```
host tên_miền
```

dig

Lệnh này cung cấp rất nhiều thông tin liên quan đến DNS. Nếu ta cung cấp cho lệnh một tên máy, *dig* sẽ in ra thông tin về máy đó, bao gồm địa chỉ IP, tên máy, và nhiều thông tin liên quan khác. Ví dụ, để tra cứu thông tin về “www.amazon.com”, ta gõ:

```
dig www.amazon.com
```

Để tra cứu tên máy tương ứng với địa chỉ IP nào đó, ta gõ:

```
dig -x 208.67.222.222
```

Ngoài ra, *dig* còn có rất nhiều các tùy chỉnh khác nữa. Xin xem trang hướng dẫn để biết thêm.

whois

Lệnh *whois* tra cứu CSDL whois thường có trên các website lớn. CSDL này lưu thông tin liên hệ của người dùng, và thường hay bị các cracker chú ý tới, do đó, máy chủ whois thường được ẩn đi hoặc hạn chế sử dụng.

wget

wget dùng để tải các tập tin hoặc web từ trên Internet về máy. Để tải về toàn bộ một trang web, ta dùng tùy chọn *-m* hoặc *--mirror*. Để tránh việc ghi đè lên một tập tin đã có sẵn, ta dùng tùy chọn *-nc*. Để tải nốt một tập tin chưa tải xong, ta dùng tùy chọn *-c* hoặc *--continue*. Cú pháp đơn giản:

```
wget địa_chỉ_url
```

wget cũng cho phép ta lấy nhiều tập tin về máy thông qua các ký tự đại diện như *, [], ?. Lưu ý là cần thêm dấu trích dẫn " vào mẫu đại diện.

Ngoài ra, *wget* còn có thể trích lấy tất cả các liên kết có trong một trang văn bản và tải chúng về:

```
wget -spider -force-html -i bookmarks.html
```

Lưu ý: Đối với giao thức HTTP, *wget* không thể tải nhiều tập tin dựa vào mẫu đại diện nào đó. Tuy nhiên ta vẫn có thể thực hiện việc tải về nhiều tập tin theo mẫu thông qua FTP. Để lấy nhiều tập

tin trên HTTP, ta có thể dùng cú pháp:

```
wget -r ll --no-parent -A.gif http://www.website.com
```

Lệnh trên sẽ thực hiện tải về máy tất cả các tập tin có đuôi .gif (tùy chọn *-A.gif* tương đương với *-A *.gif*) một cách đệ quy với cấp đệ quy là 1, tức là chỉ lấy các tập tin trong thư mục truy cập tới mà bỏ qua nội dung của các thư mục con trong đó.

wget còn có rất nhiều tùy chọn khác nữa. Xin xem thêm trang hướng dẫn để biết thêm.

➤ Cách khác để tải về một trang web

Bạn có thể dùng trình *httrack*, một chương trình đồ hoạ viết trên Python cho phép tải toàn bộ nội dung của một trang web về.

curl

curl cũng là một chương trình trên GNU/Linux, hỗ trợ việc tải và đưa dữ liệu trên mạng một cách tự động. *curl* hỗ trợ nhiều giao thức khác nhau, truy cập được vào máy chủ từ điển (dict), ldap, ftp, http, gopher. Đây là một công cụ rất phức tạp. Để xem trợ giúp của *curl*, ta dùng lệnh:

```
curl -M
```

Thông thường, ta dùng *curl* tương tự như dùng *wget*. Để tải về một nội dung nào đó yêu cầu đăng nhập xác thực, ta có thể dùng cú pháp:

```
curl -u tên_dăng_nhập:mật_khẩu http://địa_chỉ_web.com/tên_tập_tin
```

Khi cần đưa một tập tin lên mạng thông qua giao thức FTP, ta chạy lệnh:

```
curl -T tập_tin ftp://ftp.địa_chỉ_web.com
```

Khi chưa tải xong một tập tin nào đó, ta có thể tải nốt nội dung của nó về bằng lệnh:

```
curl -C - -o tập_tin http://www.địa_chỉ.com
```

12.3. Quản trị từ xa

ssh

ssh là viết tắt của Secure shell (hệ vỏ bảo mật), cho phép người dùng đăng nhập vào máy tính có dịch vụ *ssh* (do trình nền *sshd* xử lý) từ một máy khác, thông qua mạng LAN hay Internet. Sau khi đã đăng nhập vào hệ vỏ, người dùng có thể thực hiện các tác vụ như sao chép, khởi động lại máy, như đang làm việc trên chính máy mình.

Để kết nối tới một máy qua mạng, ta phải cung cấp địa chỉ IP hoặc tên máy (nếu tên máy và IP được ghi trong tập tin */etc/hosts*). Ví dụ:

```
ssh dinhtrung-pc
```

Lệnh trên sẽ kết nối đến máy có tên là *dinhtrung-pc* bên trong mạng. Tất nhiên, địa chỉ IP của máy này phải được ghi trong tập tin */etc/hosts*. Nếu không, thay vì tên máy, ta phải nhập địa chỉ IP của máy vào:

```
ssh 192.168.1.122
```

Lưu ý là tên đăng nhập dùng kết nối tới máy vẫn là tên đăng nhập hiện tại tại máy người dùng trực tiếp làm việc. Để đăng nhập vào máy ở xa với tên đăng nhập khác, ta phải thêm tên đăng nhập vào trước tên máy, như sau:

```
ssh dinhtrung@dinhtrung-pc
```

Sau đó nhập mật khẩu cho tài khoản *dinhtrung* trên máy *dinhtrung-pc*.

scp

Secure copy - Sao chép bảo mật, cũng là một lệnh trong gói *ssh*. Lệnh *scp* cho phép ta sao chép các tập tin từ máy này sang máy khác. Khi dùng với tùy chọn *-r*, *scp* chép đệ quy toàn bộ nội dung của thư mục (cả các thư mục con). Cú pháp lệnh:

```
scp tên_máy_nguồn:/thư/mục/cần/chép tên_máy_đích:/thư/mục/ghi/vào
```

Nếu ta không chỉ định thư mục nào cần ghi dữ liệu vào, *scp* sẽ dùng thư mục chính của người dùng mà ta đăng nhập từ xa vào làm đích.

Ngoài ra, nếu ta đăng nhập thông qua *ssh* vào 2 máy tính khác, ta có thể sao chép nội dung từ

máy này sang máy khác bằng *scp*. Ví dụ, ta làm việc trên máy may1, kết nối ssh tới máy may2 và máy may3. Ta có thể gõ:

```
scp may2:/tmp/file.txt may3:/mnt/winc
```

Lệnh trên sẽ chép tập tin file.txt trong thư mục /tmp của máy may2 vào thư mục /mnt/winc của máy may3.

sftp

sftp - Secure FTP cũng là một chương trình trong gói ssh. Lệnh này tương tự như ftp, nhưng dùng một kết nối tunnel an toàn cho việc truyền tập tin. Cú pháp câu lệnh cũng tương tự như dùng ftp. Để truy cập vào máy chủ phục vụ SFTP, ta gõ:

```
sftp tên_máy
```

Trên dòng lệnh sftp, ta có thể xem trợ giúp bằng cách gõ *help*, gửi một tập tin lên máy chủ bằng lệnh *put* và lấy tập tin từ máy chủ xuống bằng lệnh *get*.

Chương 13. Bảo mật

Phần này sẽ đề cập đến các vấn đề bảo mật cơ bản trên GNU/Linux. Nếu muốn xem thêm, xin bạn tham khảo cuốn Linux Security howto của Kevin Fenzi và Dave Wreski.

Thay đổi mật khẩu root

Để thay đổi mật khẩu *root* ta chỉ việc thêm cụm từ "linux single" ở câu lệnh khởi động Linux trên GRUB hoặc Lilo.

- Với Grub : Chọn dòng lệnh khởi động tương ứng (chỉ định thư mục chứa hạt nhân kernel và các tùy chọn khi khởi động), nhấn **e** để sửa lại dòng lệnh, thêm 'single' vào cuối câu lệnh có từ 'kernel' và nhấn **b** để khởi động. Sau đó, chạy lệnh *passwd* để đổi lại mật khẩu *root*.
- Với Lilo : Nhấn **ESC** rồi gõ 'linux single' và nhấn **ENTER**. Chạy lệnh *passwd* để đổi lại mật khẩu.

✓ Chú ý

Nếu không muốn người khác dễ dàng sửa đổi mật khẩu root trên máy của mình, hãy đặt mật khẩu cho LILO hoặc GRUB.

umask

umask là một giá trị được hệ vỏ dùng để điều khiển quyền truy cập mặc định đối với các tập tin được tạo ra trong phiên làm việc. Thông thường, *umask* được đặt trong một tập tin cấu hình hệ thống, để người quản trị thay đổi lại nếu cần (v.d /etc/fstab, /etc/profile).

Giá trị *umask* tương ứng với các hạn chế về quyền sử dụng tập tin. Ta lấy 777 trừ đi giá trị số tương ứng quyền truy cập của tập tin thì ra giá trị *umask* của nó. Ví dụ, nếu muốn các tập tin trong phiên làm việc được phân quyền bằng lệnh *chmod 750*, ta có thể dùng lệnh:

```
umask 027
```

13.1. Một số lệnh về bảo mật

md5sum

Tính mã kiểm tra md5 (128 bit) của 1 tập tin để kiểm tra xem nó có bị đứt gãy hay sai khác gì không. Thông thường ta dùng *md5sum -c* để kiểm tra tập tin nào đó (thường có đuôi .asc). Lệnh *md5sum* hay được dùng để kiểm tra các tập tin .iso tải xuống. Cú pháp:

```
md5sum tập_tin
```

mkpasswd -l 10

Lệnh trên tạo một mật khẩu ngẫu nhiên, gồm 10 ký tự. Nếu muốn tạo mật khẩu khó đoán, bạn

có thể dùng lệnh này.

13.2. Phân quyền cho tập tin và thư mục

Khi dùng lệnh `ls -l` ta sẽ thấy bên trái tên tập tin là các thông tin dạng như sau:

```
lrwxrwxrwx 1 dinhtrung dinhtrung 26 2008-05-05 23:24 Examples ->
/usr/share/example-content
drwxr-xr-x 2 dinhtrung dinhtrung 4096 2008-05-05 23:33 Music
-rw-r--r-- 1 dinhtrung dinhtrung 482 2008-05-07 01:18 ForU.txt
```

Ký tự đầu tiên báo cho ta biết nội dung được liệt kê là thư mục (chữ `d` — trên ví dụ, `Music` là một thư mục), liên kết (chữ `l` — trên ví dụ, `Examples` là một liên kết trở tới thư mục `/usr/share/example-content`) hoặc tập tin (dấu `-`, như tập tin `ForU.txt`). Tiếp đó là 3 bộ gồm 3 ký tự, tương ứng với quyền đọc (`r`) ghi (`w`) hoặc thực thi (`x`) của người chủ đã tạo ra nội dung, nhóm sở hữu ứng với người chủ và người khác. Trên ví dụ, liên kết `Example` cho phép mọi người đọc và ghi cũng như xem nội dung của thư mục `/usr/share/example-content`, thư mục `Music` cho người dùng `dinhtrung` mọi quyền đọc ghi và thực thi, còn những người khác chỉ có quyền đọc và thực thi, cuối cùng, tập tin `ForU.txt` cho người dùng `dinhtrung` quyền đọc và ghi, nhưng những người khác `dinhtrung` thì chỉ có quyền đọc. Con số nằm bên phải các ký tự `rwx` là số liên kết cứng tới nội dung đó. Theo sau là tên người dùng đã tạo ra hoặc sở hữu tập tin, và nhóm gắn với nó.

chmod

Lệnh `chmod` dùng để thay đổi quyền truy cập cho tập tin nào đó. Để đổi quyền, ta có hai cách là dùng số và chữ. Ta sẽ lấy một vài ví dụ dưới đây:

```
chmod u+rw file.txt
```

Lệnh trên cấp (+) quyền đọc (`r`) và ghi (`w`) tập tin `file.txt` cho người sở hữu (`u`)

```
chmod o-rwx file
```

Lệnh trên xoá (-) quyền đọc (`r`) ghi (`w`) và thực thi (`x`) của những người khác không trong nhóm và không phải là sở hữu (`o` - others).

```
chmod g=rx file.sh
```

Lệnh trên cho phép những người thuộc cùng nhóm với người sở hữu tập tin (`g`) được đọc và thực thi tập tin `file.sh`, nhưng không thể sửa đổi nội dung của nó.

```
chmod a=rwx file
```

Lệnh trên cấp quyền đọc, ghi và thực thi cho mọi người trên máy, nhưng các quyền nằm trong giá trị chỉ định bởi lệnh `umask` sẽ không bị thay đổi. Ví dụ, `umask` là `002` thì quyền của tập tin sẽ là `rw-rwxr-x`.

```
chmod 664 file.sh
```

Lệnh trên thay đổi quyền của tập tin `file.sh` thành `rw-rw-r--`. Mỗi một số trong bộ `664` tương ứng với một bộ 3 ký tự trong chuỗi phân quyền của tập tin. Với quyền đọc, giá trị là 4, quyền ghi, giá trị là 2 và quyền thực thi, giá trị là 1. Vì $6 = 4 + 2$ nên số 6 đầu tiên của bộ `664` báo cho hệ thống biết là trao quyền đọc (4) và ghi (2) cho người sở hữu (vì là số đầu tiên). Tương tự, số 6 thứ hai cũng trao quyền đọc và ghi cho nhóm của người sở hữu tập tin. Số 4 ở cuối cùng báo cho hệ thống biết là chỉ cấp quyền đọc cho những người dùng khác. Vì vậy, để cho phép mọi người có toàn quyền với một tập tin nào đó, ta dùng lệnh:

```
chmod 777 file
```

Rõ ràng, $7 = 4 + 2 + 1$ tương ứng với cả 3 quyền đọc, ghi và thực thi.

```
chmod 1700 thư_mục
```

Khi thêm số 1 ở trước, ta bật chức năng bảo vệ, hệ thống chỉ chấp nhận để người dùng xoá tập tin do mình tạo ra, những người khác (ngay cả khi có quyền ghi nội dung) không thể xoá nó đi được. Lệnh trên cũng tương đương với lệnh

```
chmod +t thư_mục
```

Thông thường, thư mục /tmp trên hệ tập tin được dùng với tùy chọn này. Để tắt chức năng bảo vệ này đi, ta dùng cú pháp:

```
chmod 0700 thư_mục
```

hoặc

```
chmod -t thư_mục
```

chown

Thay đổi người và nhóm sở hữu tập tin. Lệnh này chỉ có thể hoạt động nếu bạn là người quản trị *root* trên hệ thống. Cú pháp:

```
chown người:nhóm tập_tin
```

Khi dùng với tùy chọn *-R*, ta sẽ thay đổi chủ sở hữu của tất cả các nội dung bên trong thư mục (đệ quy).

```
chown -R người:nhóm thư_mục
```

chattr

Thay đổi các thuộc tính mà định dạng lưu trữ hệ tập tin cung cấp (thường là ext2 hoặc ext3). Khi thêm tùy chọn *-R*, ta sẽ đổi thuộc tính toàn bộ các tập tin và thư mục nằm bên trong thư mục được chỉ định. *chattr* cho phép thay đổi rất nhiều thuộc tính, xin tham khảo trang hướng dẫn của lệnh. Một số thuộc tính đáng lưu ý là:

- *A* : nếu tập tin hoặc thư mục có thuộc tính này, khi người dùng truy cập vào nó lúc đọc và ghi nội dung, thời gian truy cập sẽ không được lưu lại. Việc này giữ được thời gian ghi trên inode của hệ tập tin là thời gian tạo ra tập tin đó, thay vì thời gian truy cập cuối cùng. Thường dùng cho các thư mục hay được truy cập.
- *a* : chỉ cho phép thêm nội dung vào cuối tập tin lúc mở nó ra để ghi. Đối với thư mục có thuộc tính này, ta chỉ có thể thêm nội dung vào mà không thể xoá các tập tin đã có trong đó. Chỉ có người quản lý *root* được phép thay đổi thuộc tính này.
- *s* : khi xoá tập tin hoặc thư mục có thuộc tính này, toàn bộ nội dung của nó trên đĩa cứng sẽ được điền bằng số 0, tương tự như khi chạy lệnh *shred* (xem lại phần Làm việc với các tập tin và thư mục).
- *i* : không cho phép mọi sự thay đổi. Khi muốn sửa lại tập tin hoặc thư mục có thuộc tính này, ta phải tắt nó đi đã.

Ví dụ về cách dùng lệnh:

```
chattr +i /boot/grub/menu.lst
```

Lệnh trên không cho ai có quyền thay đổi nội dung tập tin /boot/grub/menu.lst cả, trừ khi người quản trị tắt thuộc tính *i* đi bằng lệnh

```
chattr -i /boot/grub/menu.lst
```

lsattr

Liệt kê các thuộc tính đặc biệt của tập tin và thư mục, được đặt bằng lệnh *chattr*. Dùng với tùy chọn *-R* để liệt kê đệ quy mọi tập tin và thư mục bên trong thư mục nào đó.

```
lsattr /file/or/dir
```

Chương 14. Nén và giải nén

Chương này đề cập đến vấn đề nén và giải nén các tập tin trên dòng lệnh, thường dùng trong quá trình cài đặt các phần mềm thông qua mã nguồn.

tar

Lệnh *tar* là công cụ chính dùng để nén và giải nén dữ liệu trên dòng lệnh. *tar* là định dạng đơn giản lưu trữ nhiều tập tin, nhưng không nén chúng lại (kích thước giảm không đáng kể). Muốn giảm kích thước của tập tin nén, ta phải dùng các công cụ nén khác, thường là *gzip* hoặc *bzip2*.

Các tùy chỉnh chính của nó:

- `-c` : tạo tập tin nén
- `-v` : in ra các tập tin nào đang được xử lý (nén hoặc giải nén)
- `-f` : tạo hoặc giải nén từ một tập tin; tùy chọn này phải được đặt ở sau các tùy chọn khác.
- `-z` : chạy lệnh `gzip` hoặc `gunzip` trước.
- `-x` : giải nén các tập tin có trong tập tin nén `tarball`.
- `-p` : giữ nguyên ngày tạo và phân quyền của các tập tin được giải nén ra.
- `-j` : chạy `bzip2` để tối ưu kích thước nén.
- `--exclude=mẫu` : bỏ qua các tập tin khớp với mẫu được cung cấp.

Ví dụ về cách dùng `tar`:

```
tar -cjvpf file.tar.bz2 file1 file2 file3
```

Lệnh trên sẽ ghép 3 tập tin `file1 file2 file3` vào và tạo ra tập tin `file.tar.bz2` (nén bằng `bzip2`)

```
tar -zxvpf file.tar.gz
```

Lệnh trên sẽ giải nén tập tin `file.tar.gz`, in ra các tập tin đang được xử lý trong quá trình chạy.

rsync

`rsync` được viết để thay thế cho lệnh `rcp`. Nó có thể dùng `ssh` để mã hoá thông tin và sao chép dữ liệu giữa hai máy tính, hoặc trong cùng máy tính, và từ máy chủ `rsync`. Khi chạy, `rsync` dùng một thuật sai phân để tiết kiệm băng thông và thời gian sao chép bằng cách bỏ qua những thông tin giống nhau, và chỉ chép những phần mới hoặc đã thay đổi. Công cụ này khá phức tạp, nên ta sẽ chỉ đi tìm hiểu một số cú pháp thường gặp. Xin bạn tham khảo thêm trang hướng dẫn của `rsync` để biết thêm chi tiết.

Dưới đây là một số ví dụ về cách dùng `rsync`:

```
rsync -t *.c foo:src/
```

Lệnh này sẽ truyền tất cả các tập tin khớp với mẫu `*.c` từ thư mục hiện hành tới thư mục `src` của máy tên là `foo`. Nếu trên thư mục đích đã có sẵn một vài tập tin rồi, thì `rsync` sẽ chạy cơ chế cập nhật để chỉ gửi các thông tin mới hoặc thay đổi để tiết kiệm thời gian và băng thông.

```
rsync -avz foo:src/bar /data/tmp
```

Lệnh trên sẽ gửi đệ quy tất cả các tập tin có trong thư mục `src/bar` trên máy `foo` tới thư mục `/data/tmp/bar` về máy mình. Với tùy chọn `-a`, toàn bộ các liên kết mềm, tập tin thiết bị, thuộc tính và phân quyền của các tập tin gốc sẽ được giữ nguyên. Tùy chọn `-z` báo cho `rsync` biết là phải nén dữ liệu lại khi gửi để tiết kiệm thời gian truyền tin.

```
rsync -avz foo:src/bar/ /data/tmp
```

Khi thêm dấu `/` ở cuối đường dẫn đích, `rsync` sẽ hiểu là ta muốn nó tránh tạo thêm thư mục ở đích: "chỉ chép nội dung thư mục này" thay vì "chép thư mục này lẫn nội dung của nó". Ví dụ, hai câu lệnh sau sẽ tương đương nhau:

```
rsync -av /src/foo /dest
rsync -av /src/foo/ /dest/foo
```

Ta cũng có thể dùng `rsync` để chép thông tin giữa các thư mục trong máy mình, thay vì sao chép bằng lệnh `cp` vốn không linh hoạt và hiệu quả bằng.

14.1. Các định dạng nén khác

Có hai công cụ chính để nén dữ liệu trên GNU/Linux là `bzip2` và `gzip`. Đúng như tư tưởng trên GNU/Linux, `bzip2` và `gzip` đều được thiết kế để nén 1 tập tin riêng rẽ. Do vậy, trước khi nén dữ liệu ta phải gộp tất cả các tập tin cần nén lại bằng lệnh `tar`.

GNU zip (gzip)

`gzip` là chương trình nén thông dụng nhất trên GNU/Linux:

```
gzip file.tar
```

Lệnh trên sẽ nén một tập tin .tar (gồm nhiều tập tin được bó lại với nhau) và tạo ra tập tin file.tar.gz. Gzip có thể nén bất kỳ loại tập tin nào, không nhất thiết phải là đuôi .tar. Để giải nén, ta dùng lệnh:

```
gunzip file.gz
```

bzip2

bzip2 cho phép ta nén dữ liệu chặt hơn, nhưng thời gian xử lý cũng lâu hơn gzip.

```
bzip2 file.tar
```

Lệnh trên sẽ nén tập tin file.tar vào thành tập tin mới file.tar.bz

```
bunzip2 file.tar.bz2
```

Lệnh trên sẽ giải nén tập tin file.tar.bz2 thành tập tin file.tar đặt trong thư mục hiện thời.

zipinfo

Lệnh zipinfo cung cấp thông tin chi tiết về một tập tin nén ở dạng ZIP (như WinZip trên Windows)

```
zipinfo file.zip
```

zipgrep

Lệnh zipgrep cho phép ta tìm kiếm các tập tin trong một tập tin ZIP mà không cần giải nén nó ra. Cú pháp lệnh:

```
zipgrep mẫu file.zip
```

bzip2recover

Lệnh bzip2recover được dùng để khôi phục lại một tập tin nén bzip2 bị hỏng. Nó sẽ giải nén tất cả các khối dữ liệu có trong tập tin bzip2, và ta có thể dùng bzip2 -t để kiểm tra tính toàn vẹn của từng tập tin một và lấy lại các tập tin không bị hỏng.

bzme

Lệnh này chuyển đổi một tập tin nén bằng ZIP hoặc gzip sang định dạng bzip2. Cú pháp lệnh:

```
bzme file
```

◆ Mẹo

Cả gzip và bzip2 đều có các công cụ giúp ta làm việc ngay bên trong tập tin nén. Nó cho phép ta liệt kê các tập tin có trong tập tin nén, sau đó chạy less hoặc grep để tìm ra tập tin mình cần, rồi giải nén riêng tập tin đó ra.

Với dạng gzip, ta dùng các lệnh thường có chữ đầu là z, như zcat, zless và zgrep.

Với dạng bzip2, các lệnh có chữ đầu là bz, như bzcat, bzless và bzgrep.

Chương 15. Đồ hoạ

Chương này giới thiệu với bạn một số lệnh thực thi trên giao diện dòng lệnh để xử lý ảnh, cho phép ta tự động hoá hoặc lập trình hệ vỏ một số tác vụ cơ bản với các tập tin hình ảnh.

montage

Lệnh *montage* tạo một ảnh từ nhiều ảnh khác nhau bằng cách sắp xếp các ảnh gốc theo thứ tự bất kỳ. Ví dụ:

```
montage r32.jpg r34.jpg skylines* skyline_images.miff
```

Lệnh trên sẽ tạo ra tập tin *skyline_images.miff* từ các tấm ảnh bắt đầu bằng *skylines* và 2 tấm ảnh *r32.jpg*, *r34.jpg*.

➤ **Lưu ý**

Các ảnh gốc sẽ được chuyển về cùng một kích thước, rồi xếp lát một cách ngẫu nhiên để tạo ra ảnh phức.

convert

Để chuyển định dạng ảnh này sang định dạng ảnh khác, ta dùng lệnh *convert*. Ngoài ra, *convert* còn có thể xử lý một số tác vụ ảnh đơn giản. Ví dụ, để chuyển đổi một tấm ảnh từ định dạng JPEG sang định dạng PNG, ta làm như sau:

```
convert JPEG: file.jpg PNG: file.png
```

import

Chụp ảnh màn hình từ giao diện đồ hoạ X và lưu vào một tập tin. Cú pháp:

```
import tên_tập_tin
```

display

Lệnh *display* hiển thị ảnh lên màn hình. Sau khi mở một bức ảnh bằng *display* thậm chí ta còn có thể thực hiện một số chỉnh sửa đơn giản cho tấm ảnh. Ngoài ra, *display* còn có thể trình chiếu ảnh theo thứ tự, hoặc chụp ảnh một cửa sổ trên màn hình. Cú pháp lệnh:

```
display tên_tấm_ảnh
```

Lệnh trên sẽ hiển thị nội dung tấm ảnh *tên_tấm_ảnh* trên màn hình.

```
display *.jpg
```

Lệnh này trình chiếu lần lượt tất cả các ảnh *.jpg* trong thư mục.

identify

Lệnh này liệt kê định dạng, kích cỡ, độ sâu màu và các thông tin liên quan đến một bức ảnh. Dùng với tùy chọn *-v* để xem thông tin chi tiết hơn nữa. Cú pháp:

```
identify tên_ảnh
```

mogrify

Lệnh *mogrify* cho phép chuyển đổi định dạng, thay đổi kích cỡ, xoay, và một vài hiệu ứng khác lên tấm ảnh. Ta có thể dùng *mogrify* để xử lý cùng lúc nhiều ảnh. Một số thao tác hay dùng với *mogrify* là:

```
mogrify -format jpeg *.tiff
```

Lệnh trên chuyển tất cả các tập tin *.tiff* sang định dạng *jpeg*.

```
mogrify -geometry 120x120 *.jpg
```

Lệnh trên thay đổi kích thước của các tập tin *.jpg* thành 120x120 pixel.

showrgb

Lệnh *showrgb* cho phép ta biết chính xác các thành phần màu đỏ, lục, lam cấu tạo nên một màu được giao diện X sử dụng. Lệnh này không đòi hỏi tham số nào khác. Cú pháp:

```
showrgb
```

➤ **Lưu ý:**

Tất cả các lệnh được liệt kê ở trên, ngoài lệnh *showrgb*, đều thuộc bộ công cụ ImageMagick. Xin xem trang hướng dẫn của ImageMagick (gõ lệnh *man imagemagick*) để biết thêm thông tin về các lệnh khác nữa!

Chương 16. Các lệnh cho MS-DOS

Trên GNU/Linux, ta dùng các chương trình trong gói *mtools* để làm việc với các tập tin nền MS-DOS. Khi chạy lệnh *mtools* tại dòng lệnh, các lệnh trong gói sẽ được liệt kê ra để ta tra cứu khi cần. Có rất nhiều lệnh trong gói này!

> Dùng dấu xoạc

Với các lệnh trong gói *mtools*, ta có thể tùy ý dùng dấu xoạc ngược “\” giống như trên Windows, hay dấu “/” như trên UNIX để chỉ đường dẫn trong ổ đĩa MS-DOS.

mformat

Lệnh *mformat* định dạng ổ đĩa thành đĩa mềm MS-DOS. Lưu ý rằng ta không được gắn kết ổ đĩa vào hệ thống. Cách dùng lệnh này tương tự như lệnh *format* trong DOS. Ta chỉ cần gõ lệnh sau để định dạng ổ mềm:

```
mformat a:
```

mcopy

Chép các tập tin có trong ổ đĩa MS-DOS khi ta không gắn kết nó vào hệ thống. Lệnh *mcopy* tương tự như lệnh *copy* trên MS-DOS, nhưng cao cấp hơn. Cú pháp lệnh như sau:

```
mcopy a:/tập_tin /thư/mục/dích
```

mmount

Gắn kết ổ đĩa MS-DOS, không cần thông qua lệnh *mount*. Ví dụ:

```
mmount a: /mnt/floppy
```

Lệnh trên sẽ gắn kết ổ đĩa mềm a: vào trong thư mục */mnt/floppy*.

mbadblocks

Quét một ổ đĩa MS-DOS để tìm những khối dữ liệu lỗi và đánh dấu để hệ thống không động tới chúng khi sao chép dữ liệu mới lên đĩa. Cú pháp:

```
mbadblocks a:
```

dosfsck

Lệnh này kiểm tra và sửa lại hệ tập tin MS-DOS. Dùng với tùy chọn *-a* để tự động sửa lỗi, *-t* để đánh dấu các rãnh đĩa không đọc được và *-v* để thông báo tiến độ công việc khi thực hiện kiểm tra. Ví dụ:

```
dosfsck -at /dev/fd0
```

Lệnh trên sẽ kiểm tra ổ đĩa mềm liên kết với tập tin */dev/fd0* và tự động sửa lỗi cũng như đánh dấu các rãnh đĩa hỏng nếu phát hiện được.

Chương 17. Đặt kế hoạch chạy lệnh trong chế độ nền

Có hai công cụ chính thường dùng để đặt kế hoạch chạy một số tác vụ nào đó trên GNU/Linux là *at* và *cron*. Nếu máy tính của ta không chạy liên tục, ta có thể dùng thêm *anacron*, vì *cron* chỉ làm việc tốt khi máy tính được bật liên tục (*anacron* có thể lưu trữ lịch chạy các chương trình khác ngay cả khi máy tính được tắt đi và thực hiện các tác vụ đã định khi máy được bật).

at

at thực thi một lệnh nào đó 1 lần tại một thời điểm xác định trước. Tại dòng lệnh, ta gõ:

```
at 21:30
```

sau đó nhập vào các lệnh cần chạy, giống như khi viết văn lệnh hệ vỏ vậy. Sau khi đã đặt xong các lệnh sẽ thực thi, nhấn **CTRL-D** (ký tự EOF) để thoát. *at* sẽ thực thi những lệnh bạn vừa viết vào lúc 21:30 cùng ngày. Ngoài ra, ta còn có thể thay giờ cụ thể (ở đây là 21:30) bằng *now + hh:mm:ss* để yêu cầu *at* chạy các lệnh ta nhập trong thời gian tới. Ví dụ:

```
at now + 1 hour
```

sẽ thực thi các lệnh ta nhập sau đó trong 1 giờ sau.

Ngoài ra, ta cũng có thể viết sẵn một văn lệnh hệ vỏ và dùng tùy chọn *-f* để thực thi nó với *at*.

```
at -f shell_script.sh now + 1 hour
```

Với câu lệnh trên, *at* sẽ gọi văn lệnh *shell_script.sh* trong 1 giờ sau.

atq

Liệt kê các công việc đang đợi *at* thực thi của người dùng hiện thời, nếu người quản trị chạy lệnh *at* thì *atq* sẽ liệt kê tất cả các công việc đang đợi được thực thi bởi trình nền *at*. Lệnh này không cần tham số hay tùy chọn nào hết.

atrm

Xoá một công việc khỏi danh sách các việc đang đợi *at* thực hiện. Dùng lệnh *atq* ta có thể biết được số ID của công việc đang đợi, và ta có thể dùng số đó trong lệnh *atrm*. Ví dụ:

```
atrm 2
```

sẽ xoá công việc xếp thứ 2 trong hàng đợi của *at*.

cron

cron cho phép người dùng đặt lịch thực hiện một tác vụ nào đó trong từng phút, từng giờ, ngày, tuần hay tháng. Thông thường ta dùng *crontab* để thực hiện việc sửa lịch đã đặt, vì lệnh này tự động xử lý trình nền *cron* và dễ dùng hơn đối với người dùng thông thường.

◆ **anacron**

anacron là một lệnh khác tương đương với *cron* nhưng dành riêng cho các máy tính không chạy liên tục, như các máy tính để bàn cá nhân. Nếu *cron* bỏ qua các lệnh được đặt lịch chạy trong thời gian máy tính không hoạt động, thì *anacron* sẽ thực hiện chúng trong lần khởi động kế.

crontab

Lệnh *crontab* được dùng để chỉnh sửa, đọc và xoá các tập tin mà trình nền *cron* sẽ đọc. Các tùy chọn dành cho *crontab* như sau:

- *-e* : sửa tập tin.
- *-l* : liệt kê nội dung tập tin.
- *-u username* : sửa lại lịch chạy chương trình của một người dùng khác.

Khi dùng *crontab -e* ta sẽ phải khai báo 6 trường, ngăn cách nhau bằng dấu cách. Ví dụ về một dòng lệnh được ghi vào *crontab* như sau:

```
5 4 * * sun echo "chạy lúc 4 giờ 5 phút các ngày chủ nhật hàng tuần"
```

Nhìn vào câu lệnh nhập vào sau lệnh *crontab -e* hẳn bạn có thể dễ dàng hiểu được nội dung của nó: các kế hoạch được đặt thông qua 6 trường có dạng

```
mm hh dd mm ww command
```

Trong đó, *mm* là phút (0 - 59), *hh* là giờ (0 - 23), *dd* là ngày trong tháng (1 - 31), *mm* là tháng (1 - 12 hoặc dùng tên viết tắt trong tiếng Anh (Jan, Feb, ..., Dec)) và *ww* là ngày trong tuần (0 - 7, tương ứng là từ thứ Hai (Mon) đến chủ nhật (Sun) hoặc ta có thể chỉ định ngày theo chữ viết tắt tiếng Anh). Trường cuối cùng có thể chứa dấu cách nếu cần, vì đây chính là câu lệnh ta sẽ thực thi. Thông thường, đó là một tập tin văn lệnh hệ vỏ chứa các lệnh mà ta muốn chạy. Ngoài ra, ta có thể dùng ký tự "*" để bỏ qua giá trị trong một trong 5 trường thời gian, hoặc dùng danh sách

dạng "0-4,8-12" hay "1,2,5,7", hoặc đặt khoảng cách giữa các giá trị trong danh sách, như "0-15/2" sẽ tương đương với danh sách "0,2,4,6,8,12,14". Do vậy, nếu muốn thực hiện một lệnh nào đó, cứ 2 giờ một lần, ta chỉ việc đưa vào trường giờ giá trị "*/*2":

```
* /*2 * * * echo "Cứ 2 giờ lại chạy cái này một lần"
```

Chương 18. Các ký tự đại diện

Các ký tự đại diện được dùng rất nhiều trên GNU/Linux để thực hiện một tác vụ trên nhiều tập tin hoặc thư mục cùng lúc, hoặc tìm kiếm các chuỗi hay cụm từ nào đó. Có hai dạng ký tự đại diện chính là các ký tự đại diện chuẩn thường dùng trong tương tác với hệ vỏ, và các biểu thức chính quy dùng để tìm và thay thế văn bản.

◆ Mẹo

Nếu bạn không muốn BASH thay thế các ký tự đại diện hoặc biểu thức chính quy, hãy thêm cặp dấu nháy đơn cho nó. Ví dụ, nếu muốn tạo một tập tin tên là "foo*" ta sẽ chạy lệnh:

```
touch 'foo*'
```

18.1. Ký tự đại diện chuẩn

Các ký tự đại diện chuẩn được dùng kết hợp với các lệnh trên GNU/Linux để thi hành các tác vụ liên quan đến nhiều tập tin hoặc thư mục cùng lúc. Xin xem thêm về các ký tự đại diện chuẩn bằng cách gõ:

```
man 7 glob
```

➤ Ứng dụng

Các ký tự đại diện chuẩn có thể chạy với hầu như tất cả các lệnh trên GNU/Linux

?

Dấu hỏi tương ứng với 1 ký tự bất kỳ. Nếu ta gõ *hd?* ở cuối câu lệnh nào đó, GNU/Linux sẽ thực hiện lệnh cho tất cả các tập tin hoặc thư mục bắt đầu bằng *hd* và có 1 ký tự đứng sau, như *hda hdb hdc hdd...*

*

Dấu sao tương ứng với một số bất kỳ các ký tự (kể cả không có ký tự nào). Ví dụ, *cd** sẽ khớp với *cd, cd1, cd21, cda, cdz4, cdabc*, tất cả những gì bắt đầu bằng *cd*. Mẫu *m*l* sẽ khớp với *ml, mil, mail, maxil...* tức là tất cả những gì bắt đầu bằng *m* và kết thúc bằng *l*.

[]

Cặp ngoặc vuông được dùng để chọn ra một chuỗi. Ví dụ, mẫu *m[a,o,u]m* sẽ khớp với ba từ *mam, mom* và *mum*. Lưu ý là không có dấu cách sau dấu phẩy. Mẫu *[a-z]9* sẽ khớp với bất cứ từ nào gồm 2 ký tự bắt đầu bằng chữ thường và kết thúc bằng số 9.

{ }

Cặp ngoặc nhọn tạo ra một danh sách các mẫu hoặc tên tập tin cần xử lý. Lệnh này sẽ chép ra tất cả các nội dung khớp với 1 tên hoặc mẫu bất kỳ có trong ngoặc. Ví dụ, lệnh sau sẽ chép tất cả các tập tin *pdf* và *html* vào thư mục chính:

```
cp {*.pdf,*.html} ~
```

Giống như [], ta không được để khoảng trống ở sau dấu phẩy ngăn cách các mẫu.

[!]

Tương tự như cặp ngoặc vuông, nhưng thay vì khớp thì đây là không khớp. Ví dụ, để xoá tất cả các tập tin tên là myfile1, myfile2... nhưng không xoá myfile9, ta gõ:

```
rm myfile[!9]
```

Đây là đảo của dấu ngoặc vuông.

\

Dấu xoạc ngược được dùng để báo cho BASH hiểu chính xác các ký tự thay thế như \, ?, *, [,], {, } theo nghĩa "hẹp", tức là chúng không phải là ký tự đại diện nữa. Ví dụ, các tham số của một lệnh thường ngăn cách bằng dấu cách, nhưng nếu ta đưa dấu \ vào trước dấu cách, dấu cách sẽ chỉ được hiểu là dấu cách. Do vậy, để xoá một tập tin tên là "My Document.txt", ta phải gõ:

```
rm My\ Document.txt
```

thay vì

```
rm My Document.txt
```

sẽ khiến BASH hiểu là ta muốn xoá 2 tập tin, My và Document.txt.

18.2. Biểu thức chính quy

Các biểu thức chính quy là các mẫu đại diện nhưng dùng trong việc tìm kiếm văn bản. Xem thêm về biểu thức chính quy bằng lệnh

```
man 7 regex
```

> Ứng dụng

Các biểu thức chính quy hay được dùng bởi grep và find

◆ Mẹo

Đôi khi ta phải thêm dấu nháy đơn ' ở đầu và cuối câu lệnh, rồi dùng dấu \ để tách các ký tự đặc biệt ra thì biểu thức chính quy mới hoạt động.

.

Dấu chấm khớp với một ký tự bất kỳ, giống như dấu ? trong bộ ký tự thay thế. Ví dụ, m.p sẽ khớp với map mep mip mup.

\

Dấu xoạc ngược sẽ giữ nguyên một ký tự đặc biệt về đúng nghĩa của nó. \? sẽ ứng với dấu ?, chứ không phải là một ký tự bất kỳ nữa.

.*

Khớp với một số bất kỳ các ký tự bất kỳ, tức là một chuỗi bất kỳ, tương đương với dấu * trong bộ ký tự thay thế chuẩn.

*

Ký tự đứng trước dấu sao được phép lặp lại bao nhiêu lần cũng được (kể cả không lặp lại). Ví dụ, n* sẽ khớp với n, nn, nnn, nnnnnnnnnn nhưng không khớp với no, na, nan.

^

Dấu mũ có nghĩa là "ở đầu dòng". Ví dụ, "^after" sẽ khớp với dòng nào bắt đầu bằng từ after.

\$

Dấu dollar có nghĩa là "ở cuối dòng". Ví dụ, "5\$" sẽ khớp với dòng nào kết thúc bằng số 5. Khi dùng kết hợp \$ và ^, ta có thể tìm ra nhiều dòng bắt đầu và kết thúc xác định. Ví dụ:

```
cat file | grep '^s.*n$'
```

Lệnh trên duyệt nội dung tập tin file và in ra các dòng bắt đầu bằng chữ s, theo sau là 1 chuỗi bất kỳ, rồi kết thúc bằng chữ n.

[]

Tương tự như ở bộ ký tự đại diện chuẩn, dấu ngoặc vuông dùng để đặt một danh sách các từ có thể có.

|

Dấu gạch đứng dùng để thực hiện so khớp hai hay nhiều mẫu khác nhau. Chỉ cần một trong hai biểu thức chính quy ở hai bên dấu này khớp thì đối tượng sẽ được xử lý.

[^]

Tương tự như mẫu [!] ở ký tự đại diện chuẩn. m[^i]m sẽ khớp với mam, mum nhưng không khớp với mim.

18.3. Một số ký tự hay dùng (theo chuẩn POSIX)

Phần này được trích ra từ trang hướng dẫn của lệnh *grep*:

- [:upper:] chữ hoa.
- [:lower:] chữ thường.
- [:alpha:] chữ.
- [:digit:] số từ 0 đến 9
- [:alnum:] cả số và chữ.
- [:space:] khoảng trắng, bao gồm dấu cách, dòng mới và phím TAB.
- [:graph:] các ký tự in được, trừ các khoảng trắng.
- [:print:] các ký tự in được, bao gồm cả các khoảng trắng.
- [:punct:] các dấu !, \$, %, {, }, [,]...
- [:cntrl:] các ký tự điều khiển.
- [:xdigit:] các số hexa.

➤ Cách dùng

Các mẫu ký tự đại diện trên thường dùng so khớp với các công cụ làm việc với văn bản. Ví dụ, câu lệnh sau sẽ quét dữ liệu đầu ra của lệnh *ls* và in ra các dòng có chứa một chữ hoa viết trước một số:

```
ls -l | grep '[:upper:][:digit:]'
```