Bài giảng

PHÁT TRIỂN PHẦN MỀM MÃ NGUỒN MỞ

(Open Source Software Development)

Tác giả: Ngô Bá Hùng - http://ngôbáhùng.vn/

07-2011

MỤC LỤC

Chương 1 - Giới thiệu phần mềm mã nguồn mở

1.1 Phần mềm và mã nguồn phần mềm

Cho một phần mềm thực hiện chức năng cộng 2 số nguyên và in kết quả ra màn hình như sau:

Thực thi: cong.exe 1 2
Kết quả: 1+2=3

Giả sử phần mềm (hay còn gọi là chương trình) cong.exe là do lập trình viên Tèo phát triển bằng ngôn ngữ lập trình C.

Đầu tiên Tèo viết tập tin cong.c có nội dung như sau:

```
main(int argc, char *argv[]) {
   int a= atoi(argv[1]);  int b= atoi(argv[2]);
   printf("%d+%d=%d\n",a,b,a+b);
}
```

Sau đó Tèo sử dụng một trình biên dịch để biên dịch tập tin cong.c thành tập tin cong.exe.

Trong ví dụ này:

- Tập tin cong.exe được gọi là một phần mềm, hay một chương trình phần mềm. Nội dung của cong.exe bao gồm các mã máy, mã thực thi hay mã nhị phân, là các chỉ thị mà máy tính phải thực hiện,.
- Tập tin cong.c được gọi là mã nguồn của phần mềm cong.exe
- Anh Tèo được gọi là chủ sở hữu của phần mềm (cả 2 tập tin, cong.exe và cong.c)

1.2 Chủ sở hữu phần mềm

Khi một phần mềm được tạo ra nó thuộc một chủ sở hữu nào đó. Chủ sở hữu có thể là một cá nhân (lập trình viên viết ra phần mềm) hoặc là một công ty phần mềm (người bỏ tiền ra thuê mướn lập trình viên trực tiếp viết phần mềm cho công ty).Chủ sở hữu phần mềm có toàn quyền trên phần mềm mà họ là chủ sở hữu, và sẽ quyết định mức độ sử dụng và khai thác của những người khác trên phần mềm mà họ là chủ sở hữu. Khi muốn sử dụng một phần mềm đó, người sử dụng phải xin phép chủ sở hữu phần mềm thông qua một giấy phép được cập bởi chủ sở hữu phần mềm.

1.3 Giấy phép sử dụng phần mềm (License)

Được chủ sở hữu phần mềm cấp cho người muốn sử dụng phần mềm. Nó là một bản hợp đồng gồm các điều khoản và điều kiện, mô tả những gì mà chủ sở hữu phần mềm cho phép bạn khai thác phiên bản phần mềm liên quan. Nó qui định về những khả năng mà bạn có thể có được trên phần mềm mà bạn được cấp giấy phép sử dụng.

Xét giấy phép của một số loại phần mềm phổ biến sau

1.3.1 Phần mềm thương mại

Giấy phép sử dụng của phần mềm thương mại chỉ cho phép người sử dụng khai thác phần mềm theo những ràng buộc đã ghi rõ trong giấy phép. Chẳng hạn như không cho phép người sử dụng cài đặt phần mềm trên nhiều máy khác nhau. Bản quyền loại này rất bị hạn chế. Trong trường hợp có những lỗi phần mềm được phát hiện hay một số chức năng hoạt động không tốt thì người sử dụng không còn cách nào khác hơn là phải chờ cho đến khi chủ sở hữu phần mềm sửa đổi chúng. Các nhà sản xuất phần mềm đôi khi không sẵn lòng làm việc đó hoặc thực hiện chúng với thời gian rất lâu hay đôi khi người sử dụng phải trả thêm tiền cho các bản cập nhật. Người sử dụng không có một phương tiện nào để thúc đẩy tiến trình cập nhật và sửa chữa lỗi của các phần mềm thương mại.

1.3.2 Phần mềm miễn phí (freeware) và phần mềm trả một phần (shareware)

Là các phần mềm có chủ sở hữu. Được phân phối một cách tự do. Phần mềm miễn phí không đòi hỏi tiền bản quyền sử dụng phần mềm. Phần mềm trả một phần thì sau một khoản thời gian đã định người sử dụng phải trả tiền nếu như muốn được phép sử dụng tiếp. Cả hai loại phần mềm này đều không cho phép người sử dụng truy cập vào mã nguồn của phần mềm.

1.3.3 Phần mềm mã nguồn mở

Giấy phép phần mềm lại này qui định rằng nó được phân phối đến người sử dụng cùng với mã nguồn của nó mà chúng có thể bị sửa đổi. Nó có thể được phân phối lại mà không bị một ràng buộc nào khác.
Chúng ta có thể phân phối cả những thay đổi mà chúng ta đã thực hiện trên mã nguồn gốc

Các điều khoản, điều kiện mô tả trong các giấy phép sử dùng phần mềm khác nhau là khác nhau. Ở đây ta xem xét các điều khoản liên quan đến 3 khả năng sau đối với người sử dụng :

- Khả năng phân phối lại (Distribution Possibility): Quyền được phép sao chép và phân phối lại phiên bản phần mềm mà bạn đang có trong tay (có giấy phép sử dụng nó) hay không ?
- Khả năng truy cập vào mã nguồn (Accessibility to source code): Chủ sở hữu phần mềm cho phép bạn xem mã nguồn, sử dụng, sửa đổi mã nguồn phần mềm của họ cho mục đích của bạn hay không ?
- Phí sử dụng phần mềm (Free): Khi bạn sử dụng một phần mềm, bạn phải trả tiền hay không cho người chủ sở hữu phần mềm đó ?

Bảng sau cho thấy các khả năng này trên giấy phép của một số loại phần mềm thông dụng:

	Khả năng phân phối lại	Truy cập vào mã nguồn	Miễn phí
Phần mềm thương mại (Commercial Software)	Không	Không	Không
Phần mềm miễn phí (Freeware)	Đa số	Không	Có
Phần mềm trả một phần (Shareware)	Đôi khi	Không	Không
Phần mềm mã nguồn mở (Open Source Software)	Được phép	Được phép	Đa số

1.4 Phòng trào phần mềm tự do

- Nhằm tạo ra những Phần mềm tự do (free software) là những phần mềm mà người dùng có thể tự do chia sẻ, nghiên cứu và sửa đổi chúng.
- Được khởi xướng bởi Richard M. Stallman vào năm 1983 khi ông bắt đầu dự án GNU, viết tắt của "GNU is Not UNIX".
- Nhằm thay thế hệ điều hành Unix với tính năng tự do
- Thành lập quỹ phần mền tự do (FSF - Free Software Foundation) năm 1985

1.4.1 Phần mềm tự do (Free Software)

Phần mềm tự do (PMTD) đề cập đến sự do, không đề cập đến vấn đề chi phí/giá cả:"free" as in "free speech," not as in "free beer."
Một phần mềm được gọi là phần mềm tự do nếu giấy phép sử dụng của nó cho phép người sử dụng phần mềm có 4 khả năng tự do sau:

- Tự do thực thi chương trình cho bất kỳ mục đích gì
- Tự do nghiên cứu cách thực thi của chương trình và sửa đổi chúng cho mục đích của bạn. Truy cập vào mã nguồn chương trình là tiền đề
- Tự do phân phối phần mềm cho người khác
- Tự do cải tiến chương trình và phân phối cải tiến của bạn cho cộng đồng. Truy cập vào mã nguồn chương trình là tiền đề

1.4.2 Giấy phép sử dụng phần mềm GPL (General Public License)

Thông thường, các phần mềm đều được copyright nhằm bảo về quyền tác giả. Richard M. Stallman đưa ra khái niệm Copyleft là một phương pháp tổng quát nhằm làm cho một chương

trình tự do và yêu cầu tất cả những phiên bản sửa đổi hay mở rộng của chương trình cũng phải tự do. Khái niệm Copyleft được cụ thể hóa trong giấy phép «GNU General Public License»,Viết tắt «GNU GPL». Đây là giấy phép cho phần mềm tự do, được phát hành cho phần lớn các sản phẩm của dự án GNU.

Các phiên bản của GPL

Version 1 – General Public License – GPL v1, 1989
Version 2 – General Public License – GPL v2, 1991
Version 2 – Library General Public License – LGPL v2, 1991
Version 2.1 – Lesser General Public License – LGPL v2.1, 1999
Version 3 – GPLv3, 2007

1.4.2.1 GNU GPL V2

Công bố tại địa chỉ http://www.gnu.org/licenses/old-licenses/gpl-2.0.html, tạm dịch tiếng việt tại địa chỉ http://vi.wikisource.org/wiki/GPL

Một số qui định đáng lưu ý trong GPL Version 2:

- Có thể bán mã thực thi tạo ra từ phiên bản sửa đổi Tuy nhiên mã nguồn phải công bố
- Mã nguồn của sản phẩm và tất cả các sửa đổi sau đó phải tồn tại dưới dạng phần mềm tự do
- Tất cả các chương trình có sử dụng mã nguồn GPL phải phát hành dưới giấy phép GPL
- Liên kết động hay tĩnh đến mã nguồn hoặc thư viện GPL
- Sao chép một số dòng của mã nguồn GPL

1.4.2.2 LGPL

Công bố tại địa chỉ http://www.gnu.org/licenses/old-licenses/lgpl-2.1.html, tạm dịch tiếng việt tại địa chỉ http://vi.wikisource.org/wiki/LGPL

- Được tạo ra để cho phép liên kết động mã nguồn không phát hành dưới dạng GPL hoặc LGPL vào mã nguồn LGPL
- Dàn xếp việc sử dụng các thư viện tự do vào mục đích thương mại, ví dụ thư viện GNU C
- Hầu hết các điều khoản và điều kiện tương tự GPL
- Nếu bạn thay đổi và phân phối một thư viện LGPL
  - Thư viện và những thay đổi phải được công bố (mã thực thi và mã nguồn cùng với chú thích về những sửa đổi)
  - Bằng sáng chế được gắn với sự phân phối những sửa đổi

1.4 Hệ điều hành Linux

Linux là hạt nhân (kernel) của hệ điều hành được tạo ra bởi Linus Torvalds năm 1991. Linux được phát hành dưới giấy phép GNU/GPL vào năm 1992. Linux kết hợp với các tiện ích/thư viện tạo ra từ dự án GNU tạo thành hệ điều hành GNU/Linux. Những phiên bản đầu tiên là Debian và Slackware được phát hành vào năm 1993

Vào cuối năm 1990, Linus Torvalds, sinh viên đại học Helsinki, Phần Lan cố gắng phát triển các phần mềm giống như hệ thống UNIX để sử dụng cho máy tính cá nhân 386 với bộ nhớ 4Mbytes, đĩa cứng 40 Mbytes của anh ta. Anh ta tích hợp vào hệ thống mới những kết quả mà anh ta đã thực hiện từ năm 1984 trong dự án của tổ chức phần mềm tự do (Free Software Foundation). Linus cảm nhận được chất lượng của các công việc được thực hiện bởi các lập trình viên trên toàn thế giới trong khuôn khổ của dữ án GNU, vì thế đã quyết định chuyển sản phẩm của mình dưới bản quyền GPL. Anh hi vọng hệ điều hành của mình cũng được phát triển như thế. Dự án được đẩy mạnh nhanh chóng nhờ sự cộng tác của rất nhiều lập trình viên dưới sự điều phối của Linus. Cho đến thời điểm hiện nay, Linux đã tích hợp hầu hết các tính năng của một hệ điều hành hiện đại và được sự tin tưởng của nhiều người sử dụng

1.5 Phần mềm mã nguồn mở (Open Source Software)

Từ free trong tên gọi của phần mềm tự do (Free Software) làm cho người ta liên tưởng đến một loại phần mềm miễn phí hơn là 4 yếu tố tự do của nó và phần mềm tự do thì không thể làm thương mại được. Để tránh sự hiểu lầm này Eric Raymond and Bruce Perens đề xuất một tên gọi khác là Phần mềm mã nguồn mở (Open Source Software) khi thành lập Sáng kiến mã nguồn mở (Open Source Initiative) vào năm 1998. Sáng kiến này đưa ra 10 tiêu chí để đánh giá xem một giấy phép sử dụng phần mềm có đạt chuẩn là một phần mềm mã nguồn mở hay không.

- Tiêu chí (1): Tự do phân phối lại (Free Redistribution)

Bản quyền sẽ không hạn chế bất cứ ai bán hoặc cho phần mềm; và không đòi hỏi tiền bản quyền hay một chi phí nào cho thương vụ này.

- Tiêu chí (2): Mã nguồn (Source Code)

Chương trình phải được phân phối cùng với mã nguồn được công bố bằng những phương tiện công cộng mà người ta có thể lấy được mã nguồn với một chi phí sao chép hợp lý nhất

- Tiêu chí (3): Sản phẩm kế thừa (Derived Works)

Giấy phép phải công nhận những sửa đổi và những sản phẩm kế thừa; và phải cho phép chúng được phân phối với cùng những điều khoản như giấy phép của phần mềm ban đầu

- Tiêu chí (4) Tính toàn vẹn của mã nguồn của tác giả (Integrity of The Author's Source Code)

  Giấy phép có thể ngăn cản việc phân phối mã nguồn dưới dạng bị sửa đổi chỉ khi giấy phép chấp nhận sự phân phối các tập tin vá lỗi (patch file) với mã nguồn vì mục đích sửa đổi chương trình tại thời điểm xây dựng (built time) chương trình. Giấy phép phải cho phép một cách tường minh việc phân phối phần mềm tạo ra từ mã nguồn bị sửa đổi. Giấy phép có thể yêu cầu những sản phẩm kế thừa phải mang một cái tên khác hoặc số phiên bản khác so với phần mềm gốc.
- Tiêu chí (5): Không phân biệt đối xử giữa các cá nhân và các nhóm (No Discrimination Against Persons or Groups)
- Tiêu chí (6): Không phân biệt đối xử với mục đích sử dụng (No Discrimination Against Fields of Endeavor)
- Tiêu chí (7): Phân phối giấy phép (Distribution of License)

  Những quyền được kèm với chương trình phải được áp dụng đối với tất cả những người mà sau đó chương trình được phân phối lại mà không cần thiết phải thực thi thêm những giấy phép phụ của những thành phần này

- Tiêu chí (8): Giấy phép không được dành riêng cho một sản phẩm (License Must Not Be Specific to a Product)
  Những quyền được kèm theo chương trình thì không bị phụ thuộc vào việc chương trình là thành phần của một bản phân phối phần mềm cụ thể. Nếu phần mềm được rút trích từ bản phân phối đó và được sử dụng hoặc phân phối lại với những điều khoản của giấy phép của chương trình thì tất cả các bên mà chương trình được phân phối đến cũng nên có được các quyền lợi ngang bằng như những quyền lợi được đưa ra theo bản phân phối phần mềm gốc.
- Tiêu chí (9): Giấy phép không được cản trở phần mềm khác (License Must Not Restrict Other Software)
  Giấy phép không được đặt những hạn chế lên những phần mềm khác cùng được phân phối với phần mềm của giấy phép này. Ví dụ, giấy phép không được khăng khăng rằng tất cả các phần mềm khác được phân phối trên cùng một phương tiện thì phải là phần mềm mã nguồn mở
- Tiêu chí (10): Giấy phép phải trung lập về mặt công nghệ (License Must Be Technology-Neutral)
  Không có sự dự trù nào của giấy phép dành cho một công nghệ riêng hay một kiểu giao diện nào đó

OSI duy trì danh sách các giấy phép đạt tiêu chuẩn PMMNM: 66 giấy phép (18/07/2009)
Black Duck Software cập nhật thường xuyên 20 giấy phép mã nguồn mở được dùng nhiều nhất

Một định nghĩa khác về Phần mềm mã nguồn mở cũng khá thú vị của Dale Mosby; IBM Linux Technology Center; 15-Feb-07: Phần mềm mã nguồn mở là phần mềm dưới dạng mã nguồn mà nó thường được tạo ra bởi một cộng đồng ảo, cộng tác trên Internet và thường được tải về miễn phí từ Internet hoặc được phân phối dưới dạng các đĩa CD-ROM với một giá không đáng kể. Tác giả giữ bản quyền (copyright) đối với mã nguồn và phân phối mã nguồn dưới một giấy phép định nghĩa những gì bạn được làm (hoặc không được làm) đối với mã nguồn.

1.6 Lợi ích của Phần mềm mã nguồn mở

Phần mêm mã nguồn mở được phát triển bởi một cộng đồng nhiều người nhờ đó có thể tìm ra các lỗi một cách dễ dàng. Đây chính là điểm mạnh nhất của phần mềm mã nguồn mở. Mỗi người, với khả năng có hạn của mình có thể xem xét và cải tiến các công việc được thực hiện bởi những người bạn khác. Mỗi thành viên chỉ tập trung vào phần thuộc lĩnh vực chuyên sâu của mình. Năm trăm lập trình viên làm việc với thời gian khác nhau, mỗi người tập trung vào lĩnh vực chuyên sâu của mình thì sẽ tốt hơn năm mươi lập trình viên làm việc toàn thời gian. Cách phân phối của phần mềm mã nguồn mở giúp nhiều người có điều kiện tiếp cận với chúng hơn. Nhất là đối với các nước đang phát triển, nơi mà giá phần mềm dành cho phần bảo trì, bảo hành luôn là gánh nặng.

1.7 Một số phần mềm mã nguồn mở thông dụng

Tham khảo các tài liệu sau:

- Các phần mềm tương ứng trong Windows và Ubuntu, Dịch từ trang tài liệu công đồng Pháp ngữ Ubuntu-fr, Người dịch : Vũ Dỗ Quỳnh, Hà Nội – tháng 02/2008
- OPEN SOURCE GOD: 480+ Open Source Applications http://mashable.com/2007/09/23/open-source/

Chương 2 – Hạt nhân Linux (Linux Kernel)

2.1 Hệ điều hành Unix

Được phát triển bởi bởi Dennis Ritchie and Ken Thompson, các lập trình viên của Bell Lab vào năm 1969 từ hệ điều hành đa người dùng Multics. Năm 1973 được viết lại hoàn toàn bằng ngôn ngữ C

Version 6 được sử dụng rộng rãi ngoài Bell Lab.

Có nhiều dòng Unix

- Bell Labs: Unix System III năm 1977, hỗ trợ nhiều chủng loại máy tính
- AT&T: System V năm 1983
- University of California at Berkeley:
- 3BSD năm 1979, 4.3 BSD thêm vào Bộ nhớ ảo, quản lý phân trang, TCP/IP
- 4.4BSD năm 1993, thương mại hóa
- Darwin, Dragonfly BSD, FreeBSD, NetBSD, và OpenBSD
- AT&T và BSD dùng nhiều trong thương mại

Điểm mạnh của hệ điều hành Unix là:

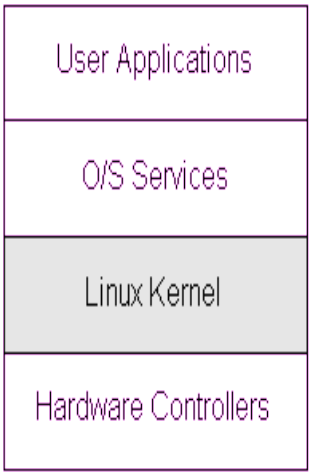
- Thiết kế đơn giản, trong sáng chỉ vài trăm lời gọi hệ thống
- Tất cả đều là tập tin, giúp đơn giản hóa thao tác xử lý dữ liệu và xuất nhập
- Hỗ trợ việc tạo tiến trình nhanh
- Cung cấp cơ chế giao tiếp liên quá trình hiệu quả
- Dễ dàng tạo ra các công cụ nhỏ, đơn giản «Do one thing and do it well»
- Dễ dàng tích hợp nhiều công cụ nhỏ để hoàn thành các tác vụ phức tạp

2.2 Lịch sử của hệ điều hành Linux

Linus Torvalds, sinh viên đại học Helsinki - Phần lan cần một Hệ điều hành có các tính năng như Unix, miễn phí, dễ dàng sửa đổi và phân phối lại mã nguồn để sử dụng cho máy tính cá nhân 386 với bộ nhớ 4Mbytes, đĩa cứng 40 Mbytes. Vì thế anh ta đã tiến hành viết một Terminal emulator nối kết vào hệ thống Unix; Sau đó anh đã tích hợp vào hệ thống mới những kết quả mà anh ta đã thực hiện từ năm 1984 trong dự án của tổ chức phần mềm tự do để tạo thành một hệ điều hành hoành chỉnh và công bố lên Internet năm 1991.

2.3 Hạt nhân Linux (Linux Kernel)

Là phần cốt lõi nhất của một hệ điều hành, được tạo ra bởi Linus Torvald, 1991, phát hành dưới license GPL. Linux chỉ là một thành phần của hệ điều hành, thành phần hạt nhân (Kernel), cốt lõi nhất của một hệ điều hành có nhiệu vụ: Trừu tượng hóa các thiết bị phần cứng, giới thiệu một máy ảo cho các chương trình người dùng; Hỗ trợ đa nhiệm (multi tasking) và hỗ trợ giao tiếp liên quá trình.

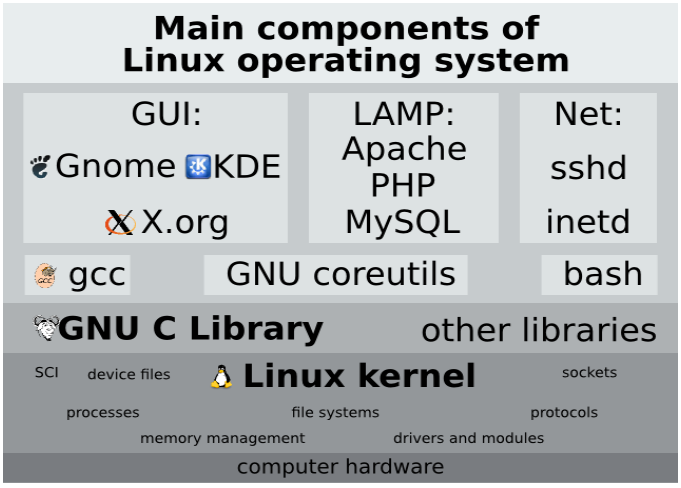


2.4 Hệ điều hành Linux (Linux Operating System)

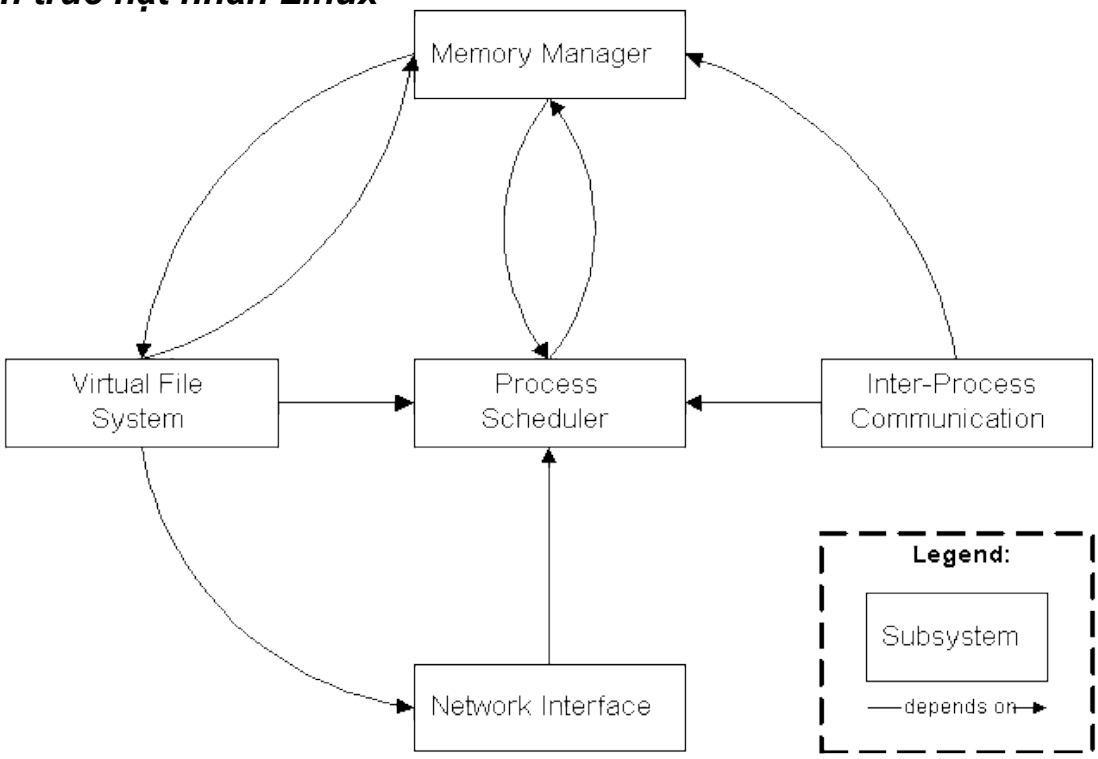
- Là các hệ điều hành sử dụng hạt nhân Linux
- Được gọi với tên Bản phân phối Linux (Linux Distribution), gọi tắt là Linux Distro
- Được phát hành bởi các nhà phân phối hệ điều hành (Linux Distributor)
  Có hơn 500 bản phân phối Linux. 10 bản phân phối phổ biến nhất năm 2010 gồm Ubuntu, Fedora, OpenSuSe, Debian, Mandriva, LinuxMint, PCLinuxOS, Slackware, Gentoo Linux, CentOS.

2.5 Các thành phần của một hệ điều hành Linux

Một hệ điều hành Linux thường bao gồm các thành phần sau: Hạt nhân Linux, trình điều khiển thiết bị, bộ khởi động, cửa sổ lệnh hoặc giao diện người dùng đồ họa, các tiện ích về tập tin và hệ thống...



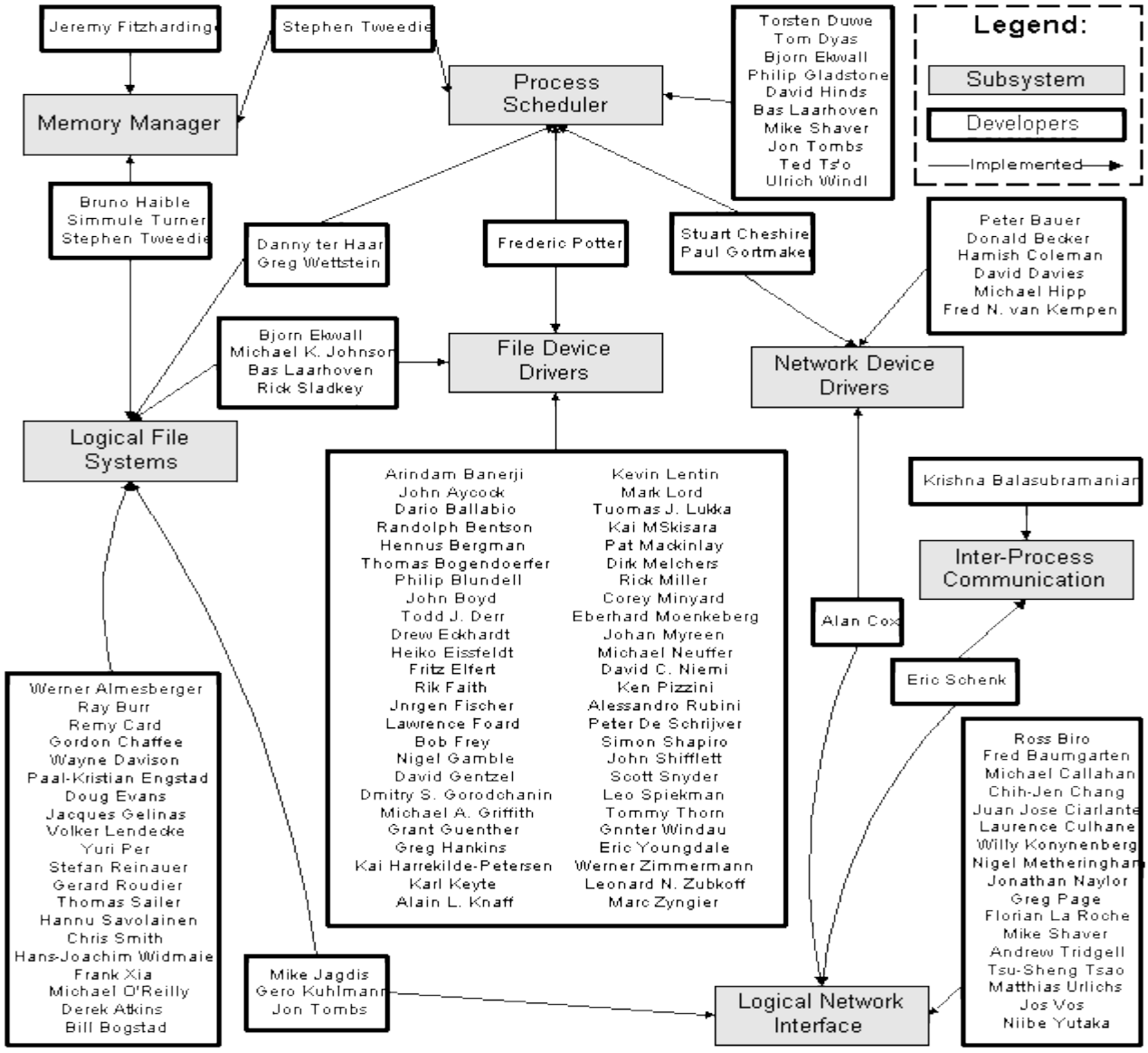
2.6 Kiến trúc hạt nhân Linux



Hạt nhân Linux gồm 5 thành phần cơ bản sau:

- Bộ định thời : Điều khiển việc truy cập đến CPU
- Bộ quản lý bộ nhớ: Đảm bảo nhiều tiến trình cùng sử dụng bộ nhớ máy tính một cách an toàn; cung cấp cơ chế bộ nhớ ảo
- Hệ thống tập tin trừu tượng: Trừu tượng hóa những chi tiết khác biệt của các loại thiết bị bằng cách giới thiệu một giao diện tập tin chung cho tất cả các thiết bị
- Giao diện mạng: Cung cấp truy cập đến nhiều chuẩn mạng và những loại thiết bị mạng khác nhau.
- Giao tiếp liên quá trình: Hộ trợ cơ chế giao tiếp giữa các tiến trình trên cùng một máy tính

2.7 Các nhóm phát triển hạt nhân Linux



2.8 Những khác biệt của Hạt nhân Linux so với Unix

- Hỗ trợ nạp động các mođun của kernel
- Hỗ trợ đa bộ xử lý đồng bộ (Symetrical MultiProcessor)
- Là kernel theo kiểu trưng dụng (Preemptive)
- Hỗ trợ đa luồng
- Hỗ trợ mô hình thiết bị hướng đối tượng, gắn nóng, hệ thống tập tin trên không gian người dùng
- Linux là tự do (Free)

2.9 Phiên bản hạt nhân Linux (Linux Version)

Có hai loại loại phiên bản Linux kernel: Phiên bản ổn định (Stable) và phiên bản phát triển (Development). Stable (ổn định) là phiên bản ở mức sản phẩm phù hợp cho việc triển khai rộng rãi. Development (phát triển) là phiên bản thử nghiệm với nhiều cải tiến được đưa vào.

Tiến trình phát triển các phiên bản diễn ra như sau:

- Đầu tiên các tính năng mới được tạo ra và thêm vào phiên bản Development của Linux kernel.
- Qua thời gian phiên bản development này được trưởng thành, và đến thời điểm tuyên bố đóng băng các tính năng: không cho thêm mới tính năng, chỉ cho chỉnh sửa tính năng đã có.
- Khi phiên bản development được xem là ổn định mã nguồn sẽ được đóng băng: chỉ chấp nhận các hiệu chỉnh lỗi.
- Phiên bản phát triển sẽ được phát hành như phiên bản stable đầu tiên của chuỗi phiên bản stable mới.

2.10 Mã nguồn của hạt nhân Linux

Mã nguồn của Linux Kernel có thể download từ địa chỉ http://www.kernel.org; Sau đó giải nén bằng lệnh tar xvjf linux-x.y.z.tar.bz2 hoặc tar xvzf linux-x.y.z.tar.gz.
Patch là đơn vị mã nguồn dùng để trao đổi trong cộng đồng phát triển, phân phối những thay đổi trên mã nguồn hay nâng cấp version mà không download toàn bộ mã nguồn version mới
2.11 Tình hình phát triển hạt nhân Linux
Để trả lời cho câu hỏi về tình hình phát triển hạt nhân Linux như Nó được phát triển nhanh như thế nào ? Ai đang phát triển nó ? Họ phát triển những gì ? và ai tài trợ cho việc phát triển Hạt nhân Linux? các tác giả Greg Kroah-Hartman, SuSE Labs / Novell Inc.; Jonathan Corbet, LWN.net và Amanda McPherson, The Linux Foundation đã thực hiện một báo vào tháng 8 năm 2009. Một số thông tin đáng chú ý từ báo cáo này như sau:

- Về bức tranh tổng thể: Từ năm 2005, hơn 5000 các nhà phát triển của gần 500 công ty tham gia vào việc xây dựng Linux kernel. Từ năm 2008 đến 2009, số người tham gia phát triển tăng 10% cho mỗi phiên bản, số lượng mã nguồn thêm vào kernel mỗi ngày tăng gần 3 lần. Quan đấy cho thấy một cộng đồng phát triển vững mạnh về cả số lượng và năng suất.

- Về mô hình phát triển: Dựa trên « loose, time-based release model», Nhanh chóng đưa các tính năng mới đến cho người dùng, giảm sự cách biệt giữa các phiên bản. Tần suất phát hành: 81 ngày/phiên bản. Tỷ lệ sửa đổi : 3.83 patch/giờ (từ 2.6.11 đến 2.6.30). Kích thước mã nguồn: thêm 10,923, xóa 5,547 và thay đổi 2,243 dòng / ngày (tính từ version 2.6.24). Version 2.6.30 có 27,911 file gồm 11,560,971 dòng lệnh
- Về mức độ đóng góp của các cá nhân vào việc phát triển mã nguồn cho Linux kernel: Đưa ra một danh sách 30 người tham gia sửa đổi Linux Kernel nhiều nhất. Bảng dưới đây chỉ trích 10 người đầu tiên trong danh sách:

Name	Number of Changes	Percent of Total Changes
David S. Miller	2,239	1.5%
Ingo Molnar	2,125	1.5%
Al Viro	1,981	1.4%
Adrian Bunk	1,883	1.3%
Takashi Iwai	1,801	1.2%
Bartlomiej Zolnierkiewicz	1,651	1.1%
Ralf Baechle	1,471	1.0%
Tejun Heo	1,457	1.0%
Stephen Hemminger	1,408	1.0%
Andrew Morton	1,370	0.9%

- Về mức độ đóng góp của các công ty vào việc phát triển mã nguồn cho Linux kernel: Đưa ra một danh sách 30 công ty đóng góp vào việc phát triển Linux Kernel nhiều nhất. Bảng dưới đây chỉ trích 10 người đầu tiên trong danh sách:

Company	Number of Changes	Percent of Total
None	26,644	18.2%
Red Hat	17,981	12.3%
Unknown	11,164	7.6%
IBM	11,151	7.6%
Novell	11,046	7.6%
Intel	7,782	5.3%
Consultant	3,657	2.5%
Oracle	3,513	2.4%
Linux Foundation	2,345	1.6%
SGI	2,317	1.6%

2.12 Những lý do các công ty hỗ trợ cho việc phát triển Linux Kernel

- Để Linux có thể chạy trên phần cứng của họ và thu hút người dùng Linux: IBM, Intel, SGI, MIPS, Freescale, HP, Fujitsu, etc...
- Để chứng tỏ khả năng của họ để thu hút khách hàng sử dụng bản phân phối của họ: Red Hat, Novell, và MontaVista,...
- Vì Linux là một thành phần trong các sản phẩm (video, tele set, mobilphone) của họ: Sony, Nokia, and Samsung
- Để xây dựng ứng dụng trên nền Linux và họ muốn phiên bản mới tiếp tục hỗ trợ ứng dụng của họ

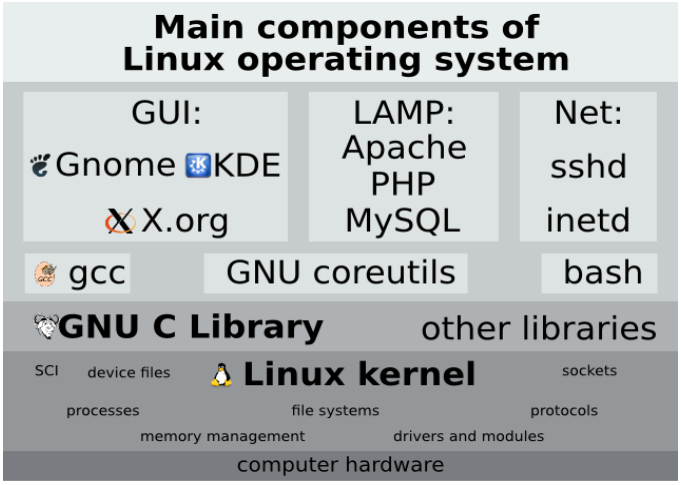
Chương 3 - Hệ điều hành Linux (Linux Operating System)

3.1 Hệ điều hành Linux

- Là các hệ điều hành sử dụng hạt nhân Linux.
- Được gọi với tên Bản phân phối Linux (Linux Distribution), gọi tắt là Linux Distro
- Được phát hành bởi các nhà phân phối hệ điều hành (Linux Distributor)
  Có hơn 500 bản phân phối Linux. 10 bản phân phối phổ biến nhất năm 2010 gồm Ubuntu, Fedora, OpenSuSe, Debian, Mandriva, LinuxMint, PCLinuxOS, Slackware, Gentoo Linux, CentOS.

3.2 Các thành phần của một hệ điều hành Linux

Một hệ điều hành Linux thường bao gồm các thành phần sau: Hạt nhân Linux, trình điều khiển thiết bị, bộ khởi động, cửa sổ lệnh hoặc giao diện người dùng đồ họa, các tiện ích về tập tin và hệ thống...



3.3 Lý do để chọn hệ điều hành Linux

- Ứng dụng: Nhiều ứng dụng sẵn dùng trên Linux (miễn phí lẫn thương mại): văn bản, đồ họa, đa phương tiện, Internet, bảo mật, quản trị, máy chủ …
- Ngoại vi: Hỗ trợ nhiều chủng loại thiết bị ngoại vi, hỗ trợ nhanh chóng các thiết bị ngoại vi mới
- Phần mềm: Tồn tại một lượng lớn các phần mềm dưới dạng mã nguồn hoặc mã thực thi
- Nền: Hỗ trợ nhiều kiến trúc máy tính: Intel, Alpha, MIPS, Motorola, 64bits system, IBM S/390, SMPs
- Bộ giả lập: Cho phép chạy các ứng dụng của các hệ điều hành khác như MS-DOS, Windows, Macintosh
- Máy ảo: Bộ quản lý máy ảo cho phép chạy nhiều máy ảo với những hệ điều hành khác nhau trên cùng một máy tính thật (máy chủ)
- Hệ điều hành chuẩn: Dùng như hệ điều hành cho những nhà sản xuất phần cứng khác nhau.
- Đa người dùng & Đa tác vụ

- Tương thích: Hơn 95% mã nguồn được viết bằng C , độc lập thiết bị, nên có thể dịch để dùng cho nhiều loại máy khác nhau: Máy chủ, máy để bàn, di động,
- POSIX (Portable Operating System Interface for Computer Environments): Cho phép ứng dụng phát triển trên Linux có thể dùng trên các hệ thống khác như UNIX
- Miễn phí, mã nguồn mở & tự do: Tiết kiệm chi phí, không phụ thuộc nhà phát triển ứng dụng

3.4 Làm việc trên một hệ điều hành Linux

- Cần được nhà quản trị máy tính Linux cung cấp một tài khoản biểu hiện bằng một tên đăng nhập (login name/username) và một mật khẩu (password)
- Thực hiện thao tác đăng nhập (login/logon) vào máy tính Linux bằng giao diện đồ họa hoặc dòng lệnh. Người dùng phải khai báo username và password đã cấp trên máy này.

3.5 Các loại tập tin

Tập tin là một khái niệm trừu tượng để chỉ các thiết bị có thể ghi hoặc đọc dữ liệu vào/ra như đĩa cứng, màn hình, con chuột, …
Có 3 loại tập tin dước Linux:

- Tập tin bình thường: là các tập tin chương trình hoặc tập tin chứa dữ liệu, văn bản
- Thư mục
- Các tập tin là các thiết bị ngoại vi

3.6 Chuẩn phân cấp hệ thống tập tin (FHS-Filesystem Hierarchy Standard)

Chuẩn phân cấp hệ thống tập tin là một tài liệu mô tả cách sắp xếp các thư mục trên hệ thống Linux. FHS được phát triển để cấp một khuôn mẫu chung nhằm giúp cho việc phát triển các ứng dụng mà không phụ thuộc vào bản phân phối Linux. FHS mô tả các thư mục sau:
* / : Thư mục gốc
* /boot: Các tập tin tĩnh cần thiết cho tiến trình khởi động
* /dev : Các tập tin thiết bị
* /etc : Các tập tinh cấu hình hệ thống và các ứng dụng
* /lib : Các thư viện chia sẻ và các môdule của hạt nhân
* /mnt : Điểm gắn nối các hệ thống tập tin một cách tạm thời
* /opt : Nơi tích hợp các gói chương trình ứng dụng
* /sbin: Các tập tin thực thi cần thiết cho hệ thống
* /tmp : Nơi chứa các tập tin tạm
* /usr : Hệ phân cấp thứ cấp
* /var : Dữ liệu biến đổi

3.7 Đường dẫn (path)

Đường dẫn là một chuỗi các tên thư mục ngăn cách nhau bởi ký tự '/', kết thúc đường dẫn có thể là tên một tập tin.
Đường dẫn tuyệt đối: là đường dẫn bắt đầu bằng thư mục gốc '/';
Ví dụ: /home/nbhung/Desktop
Thư mục hiện hành: là một vị trí trên cây thư mục
Ví dụ: /home/nbhung
Đường dẫn tương đối: là đường dẫn được tính bắt đầu từ thư mục hiện hành

Ví dụ: Desktop ; Với thư mục hiện hành là /home/nbhung

3.8 Một số thư mục đặc biệt

- Thư mục gốc ký hiệu /
- Thư mục hiện hành ký hiêu là . (một chấm)
- Thư mục cha ký hiệu .. (hai chấm)
- Thư mục cá nhân (home directory) ký hiệu ~: Mỗi người dùng có một thư mục cá nhân nơi mà người dùng có toàn quyền (thêm, sửa, xóa tập tin thư mục).

Lưu ý: Tên thư mục và tập tin có phân biệt chữ hoa và chữ thường.

3.9 Một số lệnh cơ bản trên thư mục

- Xem thư mục hiện hành: pwd
- Xem nội dung thư mục ls [dir]
- Chuyển thư mục: cd newdir
- Tạo thư mục: mkdir newdir
- Sao chép thư mục cp -r old-dir new-dir
- Xóa thư mục rỗng: rmdir a-dir
- Xóa thư mục: rm -rf a-dir

3.10 Một số lệnh thao tác trên tập tin

- Sao chép tập tin cp old-file new-file
- Đổi tên tập tin mv old-name new-name
- Di chuyển tập tin mv file-name dir-name
- Tạo liên kết ln -s file-name link-name
- Tạo/Cập nhật tập tin touch file-name
- Xóa tập tin rm [-f] file-name
- Hiển thị nội dung cat file-name

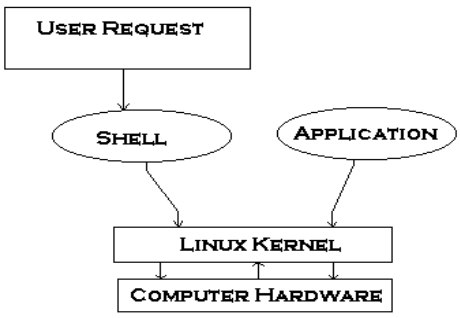
Lưu ý: Chi tiết về các lệnh được trình bày trong 2 tài liệu sau:

- Ngô Bá Hùng - Linux - Các lệnh cơ bản
- Ngô Bá Hùng - Linux - Hệ thống tập tin

Địa chỉ download : https://sites.google.com/site/nbhung/open-source

3.11 Bộ thông dịch lệnh

- Là một chương trình chạy ở mức người dùng
- Thông dịch và thực thi các lệnh nhận từ thiết bị nhập chuẩn (bàn phím) hoặc từ tập tin
- Chuyển các lệnh người dùng đến kernel
- Không thuộc kernel



Một số Shell thông dung dưới Linux:

Tên shell	Người phát triển	Nơi	Ghi chú
BASH (Bourne-Again SHell)	Brian Fox and Chet Ramey	Free Software Foundation	Phổ biến nhất trên Linux
CSH (C SHell)	Bill Joy	University of California (For BSD)	Cú pháp gần ngôn ngữ C
KSH (Korn SHell)	David Korn	AT & T Bell Labs	
TCSH	Ken Greer		Nhiều tính năng hơn CSH

Một số lệnh liên quan đến thông tin về Shell:

- Xem toàn bộ shell của hệ thống: cat /etc/shells
- Xem shell đang dùng: echo $SHELL

Có hai chế độ sử dụng shell:

- Chế độ tương tác: Thông qua một terminal; Người dùng nhập lệnh từ bàn phím; Shell thực hiện từng lệnh một.
- Chế độ kịch bản (shell script): Một chuỗi lệnh được lưu trong một tập tin văn bản, gọi là một shell script; Yêu cầu shell thực thi tập tin shell script

Những tiện lợi của shell script:

- Có thể nhận đầu vào từ người dùng hoặc tập tin và xuất kết quả ra màn hình
- Là phương tiện để tạo ra các lệnh riêng của người dùng
- Tiết kiệm thời gian vì không phải nhập lại lệnh nhiều lần
- Cho phép tự động các thao tác thường nhật
- Cho phép tự động hóa các tác vụ quản trị hệ thống

3.12 Lập trình shell

3.12.1 Tạo một shell script

- Dùng một trình soạn văn bản để biên soạn shell script
- Gán quyền thực thi cho shell script vừa biên soạn
  - chmod +x shell-script-name
  - Hoặc chmod 755 shell-script-name
- Thực thi shell script
  - bash shell-script-name
  - ./shell-script-name

Ví dụ:

Dùng trình soạn thảo văn bản lưu nội dung sau vào tập tin có tên là script-1.sh

```
#!/bin/bash
# File name: script-1.sh
clear
echo "Hello World !"
```

Cấp quyền thực thi:

```
$chmod 755 script-1.sh
```

Thực hiện shell script:

```
$./script-1.sh
Hello World !
```

3.12.2 Biến trong shell script

Có hai loại biến:

- Biến hệ thống:
  - Được tạo và duy trì bởi Linux
  - Tên biến viết hoa
  - Có thể xem toàn bộ biến hệ thống bằng lệnh set
- Biến định nghĩa bởi người dùng:
  - Được tạo và duy trì bởi người dùng
  - Tên biến viết thường

Lệnh hiển thị giá trị biến: echo $VAR_NAME

Một vài biến hệ thốngthường quan tâm:

Tên biến	Ý nghĩa
BASH=/bin/bash	Tên của shell đang dùng
BASH_VERSION= 4.0.33(1)-release	Phiên bản của bash shell
COLUMNS=80	Số cột của màn hình hiển thị
HOME=/home/nbhung	Thư mục cá nhân của người dùng hiện tại
LINES=24	Số dòng của màn hình hiển thị
LOGNAME=nbhung	Tên đăng nhập
OSTYPE=Linux	Kiểu hệ điều hành
PATH=/usr/bin:/sbin:/bin:/usr/sbin	Các thư mục sẽ được tìm đến khi một chương trình được yêu cầu thực thi
PWD=/home/nbhung/Desktop	Thư mục hiện hành
SHELL=/bin/bash	Tên của shell đang dùng
USERNAME=nbhung	Tên của người dùng hiện tại

Biến định nghĩa bởi người dùng:

Cú pháp

```
var_name=value
```

Không có khoảng trắng bên phải hay bên trái dấu =

```
no=10          #OK
no =10         #Error
no= 10         #Error
 no = 10       #Error
```

Biến NULL

```
null_var=              #hoặc null_var=""
```

3.12.3 Lệnh echo

Được dùng để hiển thị văn bản hoặc giá trị biến

Cú pháp: echo [options] [string, $variables]

Option

-n không xuống dòng

-e biên dịch các ký tự sau như ký tự đặc biệt

\a alert (bell) \b backspace
\c suppress trailing new line \n new line
\r carriage return \t horizontal tab
\\ backslash

Ví dụ:
$echo Toi la $USERNAME
Toi la nbhung
$echo -n Khong xuong dong
Khong xuong dongnbhung@nbhung-dell:~$
$echo -e "An apple a day keeps away \a\t\tdoctor\n"
An apple a day keeps away doctor

nbhung@nbhung-dell:~$

3.12.4 Lệnh tính toán biểu thức toán số học

Cú pháp: expr op1 math-operator op2
Ví dụ
$ expr 1 + 3 #4
$ expr 2 – 1 #1
$ expr 10 / 2 #5
$ expr 10 / 3 #3
$ expr 20 % 3 #2
$ expr 10 * 3 #30
$ echo `expr 6 + 3` #9

3.12.5 Các loại dấu nháy

"Bao bọc chuỗi"
$echo "Today is date"
Today is date
'Không thay đổi'
$echo "Today is 'date'"
Today is 'date'
`Biểu thức sẽ được tính trị`
$echo "Today is `date`"
Today is Tue Dec 29 14:48:45 ICT 2009

3.12.6 Lệnh read

Nhập và lưu dữ liệu từ bàn phím vào biến
Cú pháp: read variable1, variable2,...variableN
Ví dụ: Lưu nội dung dưới đây vào tập tin read.sh
echo "Your first name please:"
read fname
echo "Hello $fname, Lets be friend!"
$chmod 755 read.sh

```
$./read.sh
Your first name please:
Ngo Ba Hung
Hello Ngo Ba Hung, Lets be friend!
```

3.12.7 Các ký tự đại diện

- *: Đại diện cho một chuỗi hoặc nhóm các ký tự
- ?: Đại diện một ký tự
- [...]: Đại diện cho một trong số các ký tự được liệt kê
- [.-.]: Đại diện cho một trong số các ký tự trong khoảng
- [!...] hoặc [^...]: Đại diện cho một ký tự KHÔNG là một trong số các ký tự được liệt kê
- [!.-.] hoặc [^.-.]: Đại diện cho một ký tự KHÔNG là một trong số các ký tự trong khoảng

Ví dụ:

```
$ls *.c # Liệt kê tất cả các tập tin có tên kết thúc với .c
$ls fo? # Liệt kê tất cả các tập tin có tên 3 ký tự,     # hai ký tự bắt đầu là fo, ký tự thứ 3 là bất kỳ
$ ls [abc]*     # Liệt kê tất cả các tập tin có tên      # với ký tự bắt đầu là a,b, hoặc c
$ls [a-c]*      # Tương tự như trên
$ ls [!abc]*    #L iệt kê tất cả các tập tin có tên không bắt  đầu bằng ký tự a,b, hoặc c
```

3.12.8 Viết nhiều lệnh trên một dòng

Cú pháp: command1;command2;...;commandn
Ví dụ:

```
$date;who
Tue Dec 29 15:39:12 ICT 2009
nbhung   tty7        2009-12-29 14:04 (:0)
nbhung   pts/0       2009-12-29 14:19 (:0.0)
```

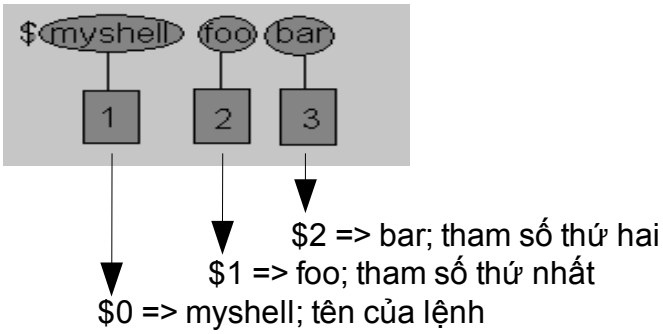
3.12.9 Các thành phần của lệnh

Cú pháp tổng quát của một lệnh là command_name arg1 arg2 arg3 … agrn

- Biến $#: giữ số lượng đối số
- Biến $* hoặc $@ tham khảo đến tất cả các đối số của lệnh

Ví dụ: $cp y y.bak

- command_name: cp
- arg1: y
- arg2= y.bak
- $# = 2
- $* = y, y.bak



Ví dụ: In các thành phần của một lệnh
Lưu nội dung sau vào tập tin có tên là print_arg.sh
```
#!/bin/bash
echo "Total number of command line argument are $#"
echo "$0 is script name"
echo "$1 is first argument"
echo "$2 is second argument"
echo "All of them are :- $* or $@"
```
Thực thi shell script print_arg.sh ta có kết quả sau:
```
$./print_arg.sh Hello World
./print_arg.sh is script name
Hello is first argument
World is second argument
All of them are :- Hello World or Hello World
```

3.12.10 Lệnh if

Cú pháp:
```
if                condition
then
```
Các lệnh sẽ được thực hiện nếu condition có giá trị là true (tức 0), hoặc kết quả trả về của lệnh condition là 0
```
fi
```
Ví dụ:
```
#!/bin/bash
# lưu trong tập tin if-exam1.sh
if cat $1
then
echo -e "\n\nFile $1, found and successfully echoed"
fi
chmod 755  if-exam1.sh
./if-exam1.sh  if-exam1.sh
.....
File if-exam1.sh, found and successfully echoed
./if-exam1.sh  not-a-file.txt
cat: not-a-file.txt: No such file or directory
```

Các lệnh kiểm tra điều kiện
Cú pháp:
```
        test expression hoặc [ expression ]
```
Ví dụ:
```
#!/bin/bash
if test $1 -gt 0
then
echo "$1 number is positive"
fi
$ chmod 755 ispostive
$ ispostive 5
```

5 number is positive
Các phép toán so sánh số:

Phép toán dùng trong shell script	Ý nghĩa	Ý nghĩa số học
-eq	Bằng nhau	5==6
-ne	Không bằng nhau	5!=6
-lt	Nhỏ hơn ?	5<6
-le	Nhỏ hơn hoặc bằng	5<=6
-gt	Lớn hơn ?	5>6
-ge	Lớn hơn hoặc bằng ?	5>=6

Các phép toán so sánh chuỗi:

Phép toán dùng trong shell script	Ý nghĩa
String1 = string2	Chuỗi string1 bằng với chuỗi string2 ?
String1 != string2	Chuỗi string1 khác chuỗi string2 ?
string1	Chuỗi string1 thì khác NULL hoặc chưa được định nghĩa
-n string1	Chuỗi string1 thì khác NULL và đã tồn tại
-z string1	Chuỗi string1 là NULL và đã tồn tại

Các phép toán kiểm tra tập tin:

Phép toán dùng trong shell script	Ý nghĩa
-s file	Tập tin file không rỗng ?
-f file	Tập tin file tồn tại hoặc là một tập tin bình thường, không là một thư mục ?
-d dir	Thư mục dir tồn tại hoặc không là một tập tin
-w file	Có quyền ghi vào file ?
-r file	Có quyền đọc nội dung file ?
-x file	Có quyền thực thi file ?

Các phép toán luận lý:

Phép toán dùng trong shell script	Ý nghĩa
! expression	NOT
expression1 -a expression2	AND
expression1 -o expression2	OR

3.12.11 Cấu trúc lệnh if-else đơn cấp

```
osch=0
echo "1. Unix (Sun Os)";       echo "2. Linux (Ubuntu)";
echo -n "Select your os choice [1 or 2]? ";    read osch;
if [ $osch -eq 1 ] ; then
    echo "You Pick up Unix (Sun Os)"
else
     if [ $osch -eq 2 ] ; then
```

```
        echo "You Pick up Linux (Ubuntu)"
    else
        echo "What you don't like Unix/Linux OS."
    fi
fi
```

3.12.12 Cấu trúc lệnh if-else đa cấp

Cú pháp:

```
if condition
then
        execute all commands up to elif statement
elif condition1
then
        execute all commands up to elif statement
else
        None of the above condtion,condtion1  are true (i.e.
        all of the above nonzero or false);   execute all commands up to fi
fi
```

Ví dụ:

```
#!/bin/bash
if [ $1 -gt 0 ]; then
  echo "$1 is positive"
elif [ $1 -lt 0 ]
then
  echo "$1 is negative"
elif [ $1 -eq 0 ]
then
  echo "$1 is zero"
else
  echo "Opps! $1 is not number, give number"
fi
```

3.12.13 Vòng lặp for

Cú pháp 1:

```
for { variable name } in { list }
do
Execute one for each item in the list until the list is not finished (And repeat all statement between do and done)
done
```

Ví dụ:

```
#!/bin/bash
if [ $# -eq 0 ]
then
echo "Error - Number missing form command line argument"
echo "Syntax : $0 number"
```

```
echo "Use to print multiplication table for given number"
exit 1
fi
n=$1
for i in 1 2 3 4 5 6 7 8 9 10
do
echo "$n * $i = `expr $i \* $n`"
done
```

Cú pháp 2:

```
for (( expr1; expr2; expr3 ))
do
Repeat all statements between do and done until expr2 is TRUE
Done
```

Ví dụ:

```
for ((  i = 0 ;  i <= 5;  i++  ))
do
  echo "Welcome $i times"
done
```

3.12.14 Vòng lặp while

Cú pháp

```
while [ condition ]
do
Command1; Command2
 ....
done
```

Ví dụ

```
n=$1;       i=1
while [ $i -le 10 ]
do
     echo "$n * $i = `expr $i \* $n`"
     i=`expr $i + 1`
done
```

3.12.15 Lệnh case

Cú pháp:

```
case  $variable-name  in
pattern1)   command
            …;;
pattern2)   command
            …;;
patternN)   command
            …;;
*)            command
              …;;
```

esac

Ví dụ:

```
case $rental in
  "car") echo "For $rental Rs.20 per k/m";;
  "van") echo "For $rental Rs.10 per k/m";;
  "jeep") echo "For $rental Rs.5 per k/m";;
  "bicycle") echo "For $rental 20 paisa per k/m";;
  *) echo "Sorry, I can not gat a $rental for you";;
esac
```

Chương 4 - Mô hình phát triển phần mềm mã nguồn mở

4.1 Giới thiệu

Một PMMNM là một phần mềm, vì thế nó được phát triển trong một dự án phát triển phần mềm, với một ngoại lệ:Là một dự án nhóm mà các thành viên của nhóm có thể chưa bao giờ gặp nhau.
Câu hỏi đặt ra: Các qui trình công nghệ phần mềm (CNPM) hay qui trình phát triển phần mềm truyền thống có ứng dụng được vào cho phát triển PMMNM hay không ?

4.2 Mô hình phát triển phần mềm truyền thống

Là mô hình xây dựng nhà thờ thời trung cổ.
Đòi hỏi tính chặc chẽ trong các công đoạn quản lý, thiết kế và xây dựng
Có sự quản lý chặc chẽ như:

- Quản lý ai là người viết các phần mã lệnh, phương pháp mà họ tích hợp các gói mã lệnh
- Định nghĩa rõ ràng một cấu trúc quản lý
- Xây dựng một kế hoạch chính xác về lịch phát hành mã lệnh

4.3 Mô hình phát triển PMMNM

Là mô hình xây dựng chợ:Không có một thiết kế ban đầu rõ ràng, không có một qui trình quản lý chính thức.
Sử dụng một chính sách lỏng lẽo trong việc:

- Phát hành mã nguồn
- Quản lý ai là người viết mã nguồn cho việc sửa lỗi và cho các chức năng mới

Nguyên tắc căn bản: «Viết mã lệnh thường xuyên, phát hành thường xuyên»
Đây là mô hình tăng trưởng: Tự phát triển khi phần mềm đạt đến một số chức năng cơ bản nào đó. Mô hình phát triển gồm 2 giai đoạn:

- Giai đoạn khởi đầu:
  - Phần mềm chưa đủ các chức năng để có thể hấp dẫn các lập trình viên khác
  - Cần một số tài trợ về tài chánh để có thể đạt đến điểm có thể sử dụng được, sẽ chuyển sang giai đoạn tăng trưởng
- Giai đoạn tăng trưởng
  - Nhận được thêm nhiều chức năng mới và các gói sửa lỗi từ cộng đồng

4.4 Sự khác biệt giữa mô hình phát triển phần mềm truyền thống và PMMNM

Có sự khác biệt về tài nguyên cho việc phát triển phần mềm trong 2 mô hình: Lập trình viên, Máy tính, Kênh phân phối, Kỹ thuật viên

Đối với CNPM truyền thống	Đối với PMMNM
Khang hiếm và tốn kém, vì thế cần quản lý	Lập trình viên là tình nguyện

chặc chẽ	Sử dụng hạ tầng cơ sở (ví dụ máy tính) sẵn có
Cần xây dựng môi trường để bảo vệ tài nguyên này	Phân phối qua Internet

4.5 Động cơ của người phát triển PMMNM

- Vì phần thưởng tiền bạc: Không nhiều
- Phần lớn PMMNM là kết quả của
  - Niềm đam mê lập trình,
  - Kết quả của một số bài tập trong các chương trình đại học,
  - Vì Lợi ích cộng đồng
- Một số công ty dùng như mô hình kinh tế để:
  - Thâm nhập thị trường đã bị thống trị bởi công ty khác
  - Phát hành sản phẩm nhanh hơn nhờ sử dụng lại PMMNM

4.6 Môi trường phát triển PMMNM

Môi trường phát triển PMMNM cần cung cấp các chức năng sau:

- Các kênh truyền thông (communication channel)
- Các cơ sở dữ liệu về lỗi (Bug database)
- Hệ thống quản lý mã nguồn (Version control)

4.6.1 Các kênh truyền thông

Gồm các thành phần như: Website, Mailing list, Bug Tracker, IRC, Wiki, Newsletters, Files bundled with code

Cung cấp các thông tin như:

- Mô tả và mục tiêu dự án
- Tin tức và bản phân phối mới nhất
- Tài liệu người dùng
- Tài liệu thiết kế
- Vật phẩm quảng cáo
- Kế hoạch và lịch trình tương lai
- Chuẩn lập trình
- Quyền sở hữu tập tin/môđun
- Danh sách lỗi đang mở (và đóng)
- Cách thức để lấy mã; đóng góp vào mã nguồn
- Liên kết tới những kênh giao tiếp khác

4.6.2 Các cơ sở dữ liệu về lỗi

Lỗi (bugs) là không tránh khỏi, cần có phương tiện để người dùng thông báo lỗi

Sử dụng mailing list có hạn chế:

- dễ bị mất,
- Lập trình viên mới không biết các lỗi trước đây

Lưu lỗi vào cơ sở dư liệu có những lợi thế:

- Dễ dàng trong tìm kiếm lỗi
- Dùng cho các mục đích khác nữa: yêu cầu tính năng, cải tiến, bản vá lỗi
- Ví dụ: Bugzilla, Mantis, Trac, Google Code

4.6.3 Hệ thống quản lý mã nguồn (Version control)

- Lưu trữ mã nguồn trực tuyến
- Theo dõi vết thay đổi trên mã nguồn
- Trộn những đụng độ trên một tập tin

4.7 Xưởng phát triển phần mềm mã nguồn mở

Được biết đến với tên Forge, là các website cung cấp môi trường phát triển PMMNM
Ví dụ:

Codendi	NovaForge,
FusionForge (ex-GForge)	QualiPSo,
InriaForge,	Picoforge,
Savane,	SourceForge
Google Code	Coclico

Chương 5 - Lập trình C trên Linux

5.1 Các công cụ cần thiết

Trình soạn thảo văn bản (text): vi, gedit, emacs, geany,...
Trình biên dịch: gcc/GNU, cc/Sun, bcc/Borland, g++/GNU, CC/Sun
Thư viện chuẩn của ngôn ngữ C: glibc

5.2 Biên dịch chương trình đơn giản

Biên soạn chương trình sau và lưu vào tập tin có tên là hello.c

```
/*hello.c*/
#include <stdio.h>
main()
{
    printf("Hello, world!\n");
    return 0;
}
```

Mở một terminal và dịch chương trình hello.c trên bằng trình biên dịch gcc của GNU

- gcc hello.c
- Tạo ra tập tin thực thi a.out
- gcc -o hello hello.c
- Tạo ra tập tin thực thi hello
- gcc -c hello.c
- Tạo ra tập tin mã đối tượng hello.o
- Thực thi
- ./a.out
- ./hello

Một số tùy chọn khác của gcc:
-Wall: hiển thị toàn bộ các warning
-ansi: Sử dụng C chuẩn ANSI
-o: Đặt tên cho tập tin kết quả biên dịch
-c: Tạo các tập tin đối tượng, không liên kết
-lm: Liên kết với thư viện toán, nếu trong chương trình có #include math.h

Ví dụ:

- gcc -o hello hello.c
- Tạo ra tập tin thực thi hello
- gcc -c hello.c bonjour.c chao.c
- Tạo ra các tập tin hello.o bonjour.o chao.o
- gcc hello.o bonjour.o chao.o -o helloworld
- Liên kết 3 tập tin mã đối tượng để tạo thành một tập tin thực thi helloworld

- Tập tin mã đối tượng giúp chỉnh sửa một tập tin không cần biên dịch lại các tập tin khác

5.3 Tập tin tiêu đề (header file)

- Chứa các định nghĩa hằng, các khai báo về các hàm hệ thống hoặc hàm thư viện mà một chương trình C có thể gọi sử dụng.
- Lưu trữ mặc nhiên ở thư mục chuẩn /usr/include và các thư mục con của thư mục này
- Sử dụng tùy chọn -I khi biên dịch để tham khảo đến các tập tin tiêu đề ở một thư mục bất kỳ
    - gcc -I/usr/openwin/include myprog.c

5.4 Tập tin thư viện hàm

- Chứa các hàm đã được biên dịch trước để có thể được sử dụng lại bởi các chương trình C khác mà không cần phải viết lại
- Các tập tin thư viện hàm chuẩn của hệ thống Linux được lưu trong thư mục /lib hoặc /usr/lib
- Qui tắc đặt tên:
    - Thư viện tĩnh (static librabry): libIndicat.a
    - Thư viện chia sẻ (shared librabry): libIndicat.so

Ví dụ: libc.a - Thư viện hàm C; libm.a - Thư viện về toán

Cách sử dụng các tập tin thư viện hàm:

- Mô tả đường dẫn đến tập tin thư viện hàm:
    - gcc -o myprog myprog.c /usr/lib/libm.a
- Dùng tùy chọn -l và indicat của thư viện hàm
    - gcc -o myprog myprog.c -lm
        - Tìm trong thư mục thư viện hàm chuẩn hệ thống;
        - Sử dụng thư viện chia sẻ libm.so trước nếu tồn tại, nếu không sẽ dùng thư viện tĩnh libm.a
- Dùng tùy chọn -L để bổ sung thư mục chứa thư viện hàm:
    - gcc -o myprog -L/usr/openwin/lib myprog.c -lX11

Cách thức xây dựng thư viện hàm:
Giả sử tạo một thư viện gồm 2 hàm là hello và bonjour.
1. Tạo tập tin header mylib.h khai bào 2 hàm này:

```
//File name: mylib.h
void hello(char * name);
void bonjour(char *name);
```

2. Tạo các tập tin cài đặt cho 2 hàm được khai báo trong mylib.h

```
// File name: hello.c
#include <stdio.h>
void hello(char * name)
{
  printf("Hello %s\n",name);
}
```

```
// File name: bonjour.c
#include <stdio.h>
void bonjour(char *name)
{
  printf("Bonjour %s\n",name);
}
```

3. Các chương trình khác include mylib.h để có thể sử dụng các hàm hello() và bonjour

```
// File name: helloworld.c
#include "mylib.h"
int main()
{
  hello("Hung");
  bonjour("Hung");
  return 0;
}
```

Cách biên dịch chương trình helloworld.c khi chưa tạo tập tin thư viện hàm:

```
gcc -c *.c
ls *.o
bonjour.o hello.o helloworld.o
gcc -o helloworld hello.o bonjour.o helloworld.o
./helloworld
Hello Hung
Bonjour Hung
```

Tạo tập tin thư viện hàm chứa mã đối tượng của các hàm hello() và bonjour()

```
ar  crv libmylib.a hello.o bonjour.o
```

Cách biên dịch chương trình helloworld.c sử dụng tập tin thư viện hàm libmylib.a:

```
gcc -o helloworld helloworld.o libmylib.a
```

Hoặc

```
gcc -o helloworld helloworld.o -L. -lmylib
```

Tiện ích nm: xem các hàm sử dụng trong một chương trình, thư viện:

```
nm helloworld
nm libmylib.a
```

5.5 Tiện ích make

5.5.1 Giới thiệu

make là một tiện ích lập trình giúp người lập trình:

- Không phải đánh lại các câu lệnh biên dịch nhiều lần
- Tránh sai sót khi nhập các tùy chọn biên dịch từ bàn phím
- Tiết kiệm thời gian biên dịch chương trình vì không biên dịch lại các tập tin nguồn không có sửa đổi
- Dễ dàng phân phối phần mềm dưới dạng mã nguồn để người cài đặt biên dịch lại khi cài đặt hệ thống

5.5.2 Tập tin mô tả

Make sử dụng tập tin mô tả có tên mặc định là makefile hoặc Makefile để chỉ dẫn make cách thức biên dịch/biên dịch lại một cách tự động một chương trình;
Một tập tin mô tả bao gồm 3 thành phần sau:

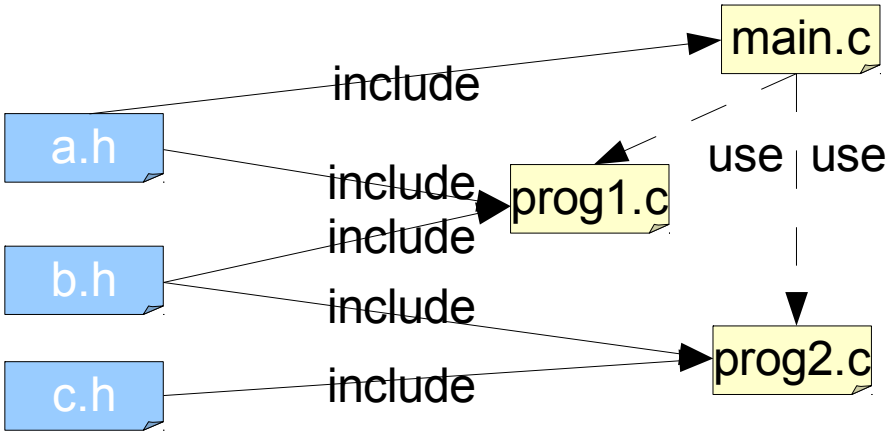
- Các mục tiêu (targets): thường là các tập tin thực thi hoặc các tập tin mã đối tượng cần tạo ra
- Những sự phụ thuộc (dependencies) để chỉ ra sự phụ thuộc của một mục tiêu vào các tập tin khác
- Các luật (rules) để chỉ ra cách thức tạo ra các mục tiêu

5.5.3 Cách thức hoạt động của make

make bắt đầu từ một mục tiêu được yêu cầu trong tập tin mô tả Makefile. Make kiểm tra xem mục tiêu hiện tại có phụ thuộc vào các mục tiêu khác không ? Nếu có đi xuống một các đệ qui các mục tiêu con. Make ịch các tập tin nguồn thành các tập tin đối tượng, sau đó liên kết chúng lại thành tập tin thực thi. Make chỉ dịch lại tập tin nguồn thành tập tin đối tượng khi tập tin nguồn này bị sửa đổi.

5.5.4 Xây dựng tập tin mô tả

Cho một dự án phần mềm phát triển bằng C có các tập tin phụ thuộc nhau như hình vẽ dưới đây:



Tập tin Makefile để biên dịch dự án phần mềm này được viết như sau:

```
myapp: main.o prog1.o prog2.o
	gcc -o myapp main.o prog1.o prog2.o
main.o: main.c a.h
	gcc -c main.c
prog1.o: prog1.c a.h b.h
	gcc -c prog1.c
prog2.o: prog2.c b.h c.h
	gcc -c prog2.c
```

Trong tập tin trên,

- Các mục tiêu là: myapp, main.o, prog1.o:, prog2.o. Đây là những cụm từ bắt đầu dòng và có ký tự hai chấm kết thúc

- Một mục tiêu sẽ phụ thuộc vào các tập tin hoặc các mục tiêu phụ khác, được liệt kê phía sau dấu 2 chấm của một mục tiêu. Ví dụ. mục tiêu myapp phụ thuộc vào các mục tiêu con main.o, prog1.o và prog2.o
- Các luật là các dòng phía dươic mục tiêu và thụt vào đầu dòng 1 tab, ví dụ gcc -o myapp main.o prog1.o prog2.o

5.5.5 Cú pháp sử dụng lệnh make

- make
  - Sử dụng tập tin makefile hoặc Makefile trong thư mục hiện hành như tập tin mô tả. Tạo mục tiêu đầu tiên trong tập tin mô tả
  - make -f MyMakeFile
    - Sử dụng tập tin MyMakeFile như tập tin mô tả
- make target-name
  - Tạo mục tiêu target-name trong tập tin mô tả

Mục tiêu all thường được định nghĩa để bao gồm tất cả các mục tiêu

5.5.6 Sử dụng macro trong tập tin mô tả

Macro cho phép viết makefile một cách tổng quát và mềm dẽo hơn, tương tự như việc sử dụng biến và hằng trong lập trình. Macro cho phép có nhiều tùy chọn cho việc biên dịch chương trình: phiên bản debug, phiên bản phát hành cũng như thay đổi trình biên dịch tùy thuộc vào hệ thống.

Đinh nghĩa macro: MACRONAME=Value
Truy cập giá trị: $(MACRONAME), ${MACRONAME} hoặc $MACRONAME

Ví dụ một tập tin Makefile có sử dụng macro

```
all: myapp
# Which compiler
CC = gcc
# Where are include files kept
INCLUDE = .
# Options for development
CFLAGS = -g -Wall -ansi
# Options for release
# CFLAGS = -O -Wall -ansi
myapp: main.o prog1.o prog2.o
  $(CC) -o myapp main.o prog1.o prog2.o
main.o: main.c a.h
  $(CC) -I$(INCLUDE) $(CFLAGS) -c main.c
prog1.o: prog1.c a.h b.h
  $(CC) -I$(INCLUDE) $(CFLAGS) -c prog1.c
prog2.o: prog2.c b.h c.h
  $(CC) -I$(INCLUDE) $(CFLAGS) -c prog2.c
```

Một số macro sẵn dùng như:

- $?: Danh sách các tập tin phụ thuộc có sửa đổi gần đây hơn so với mục tiêu hiện hành
- $@: Tên của mục tiêu hiện hành
- $<: Tên của tập tin phụ thuộc hiện hành
- $*: Tên của tập tin phụ thuộc hiền hành không có phần mở rộng
- -cmd: Bỏ qua lỗi khi thực thi cmd
- @cmd: yêu cầu make không in cmd ra màn hình trước khi thực thi nó

Chương 6 - Hệ thống quản lý phiên bản Subversion

6.1 Hệ thống quản lý phiên bản (Version Control System)

- Cho phép lưu trữ trực tuyến mã nguồn các dự án.
- Theo dõi những thay đổi trên mã nguồn.
- Trộn (merge) các đụng độ trên cùng một tập tin.
- Xây dựng theo mô hình tập trung: CVS, Subversion, Perforce
- Xây dựng theo mô hình phân tán: Git, Mercurial, Darcs

6.2 Giới thiệu Subversion

Subversion (SVN) là một hệ thống quản lý phiên bản mã nguồn mở.
Subversion cho phép:

- Quản lý tập tin, thư mục và những thay đổi trên tập tin/thư mục
- Phục hồi lại phiên bản cũ
- Phân tích lịch sử thay đổi của tập tin/thư mục
- Hoạt động trên mạng, người dùng phân tán

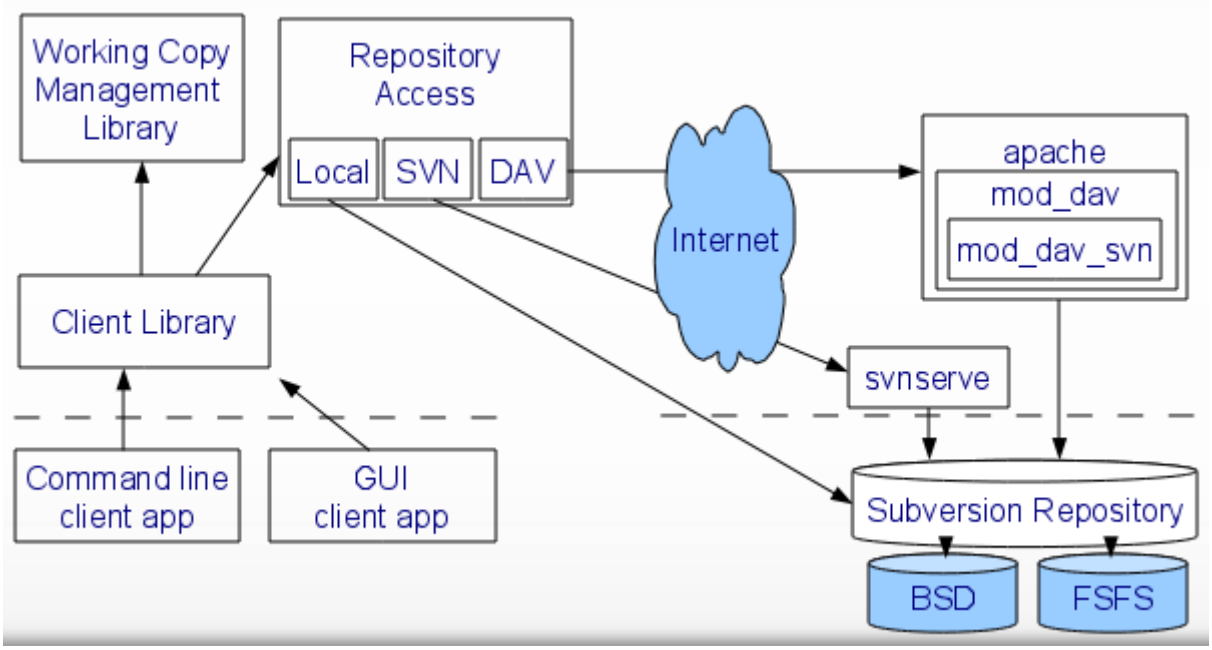
Subversion được dùng cho các mục đích sau:

- Lưu trữ tập tin/thư mục; phục hồi các phiên bản cũ; phân tích lịch sử thay đổi của tập tin/thư mục theo thời gian
- Làm việc cộng tác với đồng nghiệp trên cùng một tài liệu; theo dõi ai thay đổi gì trên tài liệu
- Phát triển phần mềm

6.3 Lịch sử phát triển của Subversion

- 2000: CollabNet (http://www.collabnet.com) thử thay thế mođun CVS trong ứng dụng CollabNet Enterprise Edition (CEE) của họ
- 5/2000: Thiết kế chi tiết với sự tham gia của Karl Fogel, tác giả của Open Source Development with CVS (1999)
- 8/2001: Subversion chính thức được công bố dưới bản quyền của CollabNet: Mã nguồn mở, tự do tải về, sửa đổi, phân phối lại, không cần xin phép

6.4 Kiến trúc của Subversion



6.5 Các thành phần của gói phần mềm subversion

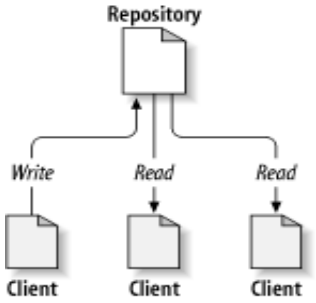
- svn: command-line client
- svnversion: chương trình báo tình trạng của một phiên bản làm việc (working copy)
- svnlook: tiện ích để kiểm tra trực tiếp một kho dữ liệu subversion (Subversion repository)
- svnadmin: tiện ích tạo, thay đổi, sửa chửa một kho dữ liệu subversion
- mod_dav_svn: module để truy cập vào kho dữ liệu svn qua web/Apache web server
- svnserve: một svn standalone server

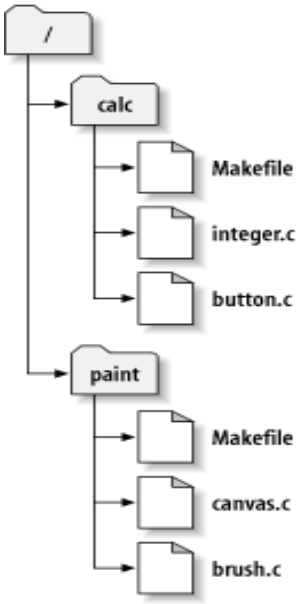
6.6 Kho chứa (Repository)

Subversion dùng repository để:

- Lưu thông tin dưới dạng một cây của hệ thống tập tin (filesytem tree)
- Chia sẻ thông tin cho các client
  - Ghi: chia sẻ thông tin cho client khác
  - Đọc: nhận thông tin từ các client khác
- Ghi nhận tất cả các thay đổi trên tập tin và thư mục
- Khi đọc bình thường: nhận được phiên bản mới nhất
- Có thể xem lại trạng thái của cây thư mục trước đó

Mỗi repository được tổ chức như một cây hệ thống tập tin (filesystem tree), lưu tập tin/thư mục của nhiều dự án (project). Mỗi dự án là một thư mục con của cây hệ thống tập tin.
Ví dụ: calc và paint là hai thư mục tương ứng cho hai dự án





6.7 Các mô hình quản lý phiên bản

Mô hình quản lý phiên bản là các chiến lược giúp một hệ thống quản lý phiên bản thực hiện được nhiệm vụ cơ bản của mình là chia sẻ thông tin giữa nhiều người dùng và tránh tình trạng ghi chồng dữ liệu lẫn nhau giữa những người dùng trên cùng một tập tin.
Có một số mô hình quản lý phiên bản như:
Giải pháp Lock-Modify-Unlock:

- Người dùng Khóa một tập tin trước khi sửa đổi nó.
- Sau khi sửa đổi hoàn thành sẽ mở khóa tập tin.
- Nhược điểm của giải pháp này là Quên mở khóa, Không nhất thiết phải tuần tự,Không an toàn khi các tập tin phụ thuộc nhau.

Giải pháp Copy-Modify-Merge:

- Mỗi người tạo một phiên bản làm việc từ Repository
- Sửa đổi trên phiên bản làm việc
- Các phiên bản làm việc được trộn lại để tạo thành phiên bản mới
- Người dùng sử lý đụng độ
- Đây là giải pháp được d ùng bởi Subversion

6.8 Định vị tập tin thư mục

Subversion sử dụng URL để định vị các tập tin/thư mục lưu trên một Repository

- Truy cập trực tiếp (trên đĩa cục bộ)
  - file:///var/svn/repos
- Truy cập thông qua giao thức WebDAV
  - http://svn.example.com:9834/repos
  - https://svn.example.com:9834/repos
- Truy cập đến một svnserve
  - svn:// hoặc svn+ssh://

6.9 Phiên bản làm việc (Working copy)

Là một thư mục bình thường trên hệ thống cục bộ. Một lập trình viên có thể sửa đổi, biên dịch mà không ảnh hưởng đến người khác, phổ biến các thay đổi của mình bằng cách dùng lệnh «publish», trộn với các phiên bản sửa đổi bởi các người khác. Thư mục con .svn theo dõi các thay đổi trên phiên bản làm việc: chưa được công bố, đã bị thay đổi bởi người khác (out of date),...

Dùng lệnh checkout để tạo một phiên bản làm việc trên máy tính cục bộ

```
$ svn checkout http://svn.example.com/repos/calc
A    calc/Makefile
A    calc/integer.c
A    calc/button.c
Checked out revision 56.

$ ls -A calc
Makefile  button.c integer.c .svn/
```

Công bố sự thay đổi:
Bạn đã thay đổi nội dung tập tin button.c. Bạn muốn công bố sự thay đổi này. Hãy dùng lệnh commit

```
$ svn commit button.c -m "Fixed a typo in button.c."
Sending        button.c
Transmitting file data .
Committed revision 57.
```

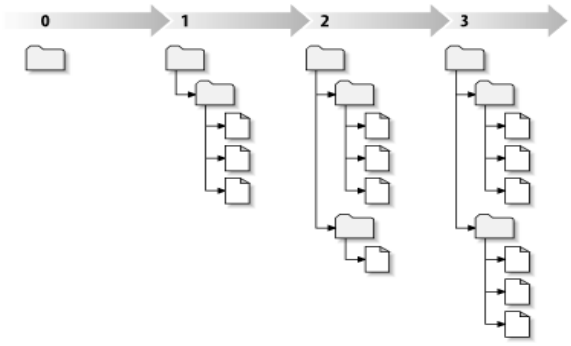
Cập nhật một phiên bản làm việc:
Một người dùng khác muốn có phiên bản mới nhất của button.c do bạn sửa đổi, họ phải cập nhật phiên bản làm việc của họ.
Dùng lệnh update để cập nhật một phiên bản làm việc

```
$ ls -A
Makefile button.c integer.c .svn/
$ svn update
U    button.c
Updated to revision 57.
```

6.10 Quản lý sự sửa đổi trển repository

Mỗi commit thực hiện bởi một lập trình viên sẽ truyền lên repository sự thay đổi trên nhiều tập tin và thư mục.
Tập các thay đổi này sẽ được cập nhật vào repository dưới dạng một giao dịch nguyên tử (automic transaction):

- Hoặc toàn bộ các thay đổi được cập nhật vào repository;
- Hoặc không có một sự thay đổi nào được cập nhật

Mỗi lần repository chấp nhận một commit, một trạng thái mới của cây hệ thống tập tin được tạo ra Tương ứng với một con số gọi là lần sửa đổi (revision)	

Một phiên bản làm việc có thể chứa tập tin và thư mục có số revision khác nhau.

6.11 Đồng bộ phiên bản làm việc với repository

Với mỗi file trong một phiên bản làm việc, Subversion ghi nhận 2 thông tin sau trong thư mục .svn:

- Số revision mà phiên bản làm việc đang dựa trên đó
- Thời điểm sau cùng file trong phiên bản làm việc được cập nhật từ repository

Dựa vào 2 thông tin, khi giao tiếp với repository, Subversion sẽ xác định 4 trạng thái của một tập tin trong phiên bản làm việc:

- Không thay đổi và hiện hành (Unchanged, and current)
  - File không thay đổi bởi người dùng và không thay đổi trên repository
  - Lệnh commit hay update sẽ không làm gì trên file này
- Thay đổi cục bộ và hiện hành (Locally changed, and current)
  - File bị thay đổi bởi người dùng nhưng không thay đổi trên repository
  - Lệnh commit sẽ cập nhật thay đổi lên repository
  - Lệnh update sẽ không làm gì trên file này
- Không thay đổi nhưng quá hạn (Unchanged, and out of date)
  - File không thay đổi bởi người dùng nhưng đã bị thay đổi trên repository,
  - Lệnh commit không làm gì trên file này
  - Lệnh update sẽ cập nhật file từ repository
- Thay đổi cục bộ và quá hạn (Locally changed, and out of date)
  - File bị thay đổi bởi người dùng và bị thay đổi trên repository
  - Lệnh commit thất bại với lỗi out-of-date
  - Phải thực hiện lệnh update trước
    - Subversion sẽ trộn những thay đổi bởi người dùng hiện tại với những thay đổi của người dùng khác một cách tự động (nếu có thể)
    - Nếu Subversion không thể trộn được, nó sẽ để cho người dùng giải quyết đụng độ

6.12 Các lệnh cơ bản trên subversion

6.12.1 Lệnh trợ giúp - help

svn help
Liệt kê các lệnh con của svn (subcommand)
svn help subcommand
Xem trợ giúp liên quan đến lệnh con subcommand
Ví dụ: svn help import

6.12.2 Đưa dữ liệu vào repository - import

Lệnh con import được dùng để sao chép một cây thự mục các tập tin vào một repository để theo dõi
tạo các thư mục và tập tin tương ứng trên Repository
Cú pháp
svn import src-dir rep-url/path -m message
src-dir: Thư mực chứa dữ liệu cần đưa vào repository
rep-url/path: Địa chỉ của repository và thư mục nơi đặt dữ liệu trong repository
message: ghi chú cho việc cập nhật dữ liệu

Ví dụ:

```
$ svn import mytree  file:///var/svn/newrepos/some/project \
          -m "Initial import"
Adding        mytree/foo.c
Adding        mytree/bar.c
Adding        mytree/subdir
Committed revision 1.

$ svn list file:///var/svn/newrepos/some/project
bar.c
foo.c
subdir/
```

6.12.3 Tạo phiên bản làm việc - checkout

Cú pháp: svn checkout rep-url/path/dir
Sao chép toàn bộ nội dung thư mục dir trên repository về thư mục hiện hành
Ví dụ

```
$ svn checkout http://svn.collab.net/repos/svn/trunk
A    trunk/Makefile.in
A    trunk/ac-helpers
    …
Checked out revision 8810.
```

6.12.4 Sửa đổi phiên bản làm việc

Sửa đổi tập tin đã tồn tại:

Dùng các trình soạn thảo một cách bình thường

Sửa đổi cấu trúc thư mục:

- Thêm tập tin/thư mục: svn add obj
- Xóa tập tin/thư mục: svn delete obj
- Sao chép tập tin/thư mục: svn copy src-obj des-obj
- Đổi tên tập tin/thư mục: svn remove old-obj new-obj
- Tạo thư mục mới: svn mkdir new-dir

6.12.5 Xem lại những sửa đổi status

Trong thư mục gốc của phiên bản làm việc đánh lệnh
svn status
Sẽ hiển thị toàn bộ các tập tin bị sửa đổi.
Ví dụ

```
?      scratch.c              # file không được quản lý bởi svn
A      stuff/loot/bloo.h      # file được thêm vào
C      stuff/loot/lump.c      # file đụng độ do cập nhật
D      stuff/fish.c           # file sẽ xóa
M      bar.c                  # file đã bị thay đổi
```

Xem trạng thái một tập tin cụ thể:

```
svn status stuff/fish.c
D      stuff/fish.c
```

Xem chi tiết về trạng thái tất cả tập tin:

```
$ svn status -v
M             44       23    sally     README
              44       30    sally     INSTALL
M             44       20    harry     bar.c
```

Xem các tập tin đã quá hạn:

```
$ svn status -u -v
M     *       44       23    sally     README
M             44       20    harry     bar.c
      *       44       35    harry     stuff/trout.c
```

Dấu * đánh dấu các tập tin đã quá hạn, cần cập nhật trước khi commit

6.12.6 Phục hồi lại các sửa chửa -revert

```
$ svn revert README
Reverted 'README'
$ svn status foo
?      foo
$ svn add foo
A        foo
$ svn revert foo
```

Reverted 'foo'
$ svn status foo
? foo
$ svn delete README
D README
$ svn revert README
Reverted 'README'
6.12.7 Cập nhật phiên bản làm việc
Cú pháp: svn update
Ví dụ:
$ svn update
U foo.c
U bar.c
Updated to revision 2.
Mã trạng thái
A: Added D: Deleted U: Updated
C: Conflict G: Merged E: Existed

6.12.7 Xử lý đụng độ khi cập nhật hoặc công bố

Giả sử rằng:

- Repository chứa sandwich.txt với revision là 1
- Sally và Harry cùng sửa đổi nội dung sandwich.txt
- Hary commit trước và tạo ra sandwich.txt với revision là 2 trên repository
- Khi commit, Sally cập nhật phiên bản hiện hành
  - SVN phát hiện sự khác biệt giữa sandwich.txt trong phiên bản hiện hành của Sally và của repository
    - SVN tạo một tập tin trộn (merged file) nếu có thể
    - Hoặc để Sally xử lý đụng độ giữa hai phiên bản

Ví dụ về đụng độ khi cập nhật phiên bản làm việc:
$ svn update
Conflict discovered in 'sandwich.txt'.
Select: (p) postpone, (df) diff-full, (e) edit, (h)elp for more options : p
C sandwich.txt
Updated to revision 2.

Tùy chọn xử lý khi đụng độ xảy ra:

- (p) postpone: Đánh dấu đụng độ để xử lý sau
- (df) diff-full: Hiển thị tất cả các sửa đổi đã thực hiện để tạo thành một tập tin trộn (merged file)
- (e) edit: Sửa đổi nội dung tập tin trộn
- (r) resolved: Xác nhận nội dung tập tin sau khi xử lý đụng độ
- (mf) mine-full: Chỉ chấp nhận những thay đổi của mình
- (tf) theirs-full: Chỉ chấp nhận những thay đổi của người khác
- (l) launch: Kích hoạt công cụ xử lý đụng độ
- (h) help: Hiện thị các tùy chọn xử lý đụng độ

Đánh dấu đụng độ để xử lý sau:
$ svn update
Conflict discovered in 'sandwich.txt'.
Select: (p) postpone, (df) diff-full, (e) edit, (h)elp for more options : p
C sandwich.txt
Updated to revision 2.
$ ls -1
sandwich.txt
sandwich.txt.mine Phiên bản chứa thay đổi của người dùng hiện tại
sandwich.txt.r1 Phiên bản trước khi sửa đổi
sandwich.txt.r2 Phiên bản tải từ repository
Hiển thị tập tin trộn:
Select: (p) postpone, (df) diff-full, (e) edit,
(h)elp for more options : df
--- .svn/text-base/sandwich.txt.svn-base Tue Dec 11 21:33:57 2007
+++ .svn/tmp/tempfile.32.tmp Tue Dec 11 21:34:33 2007
@@ -1 +1,5 @@
-Just buy a sandwich.
+<<<<<<< .mine
+Go pick up a cheesesteak.
+=======
+Bring me a taco!
+>>>>>>> .r32

Sửa đổi tập tin trộn:
Dùng trình soạn thảo văn bản thông thường để sửa đổi tập tin trộn
Xóa các đánh dấu đụng độ (conflict marker)
<<<<<<< .mine #conflict marker
Các sửa đổi của người dùng hiện tại
....
======= #conflict marker
Các sửa đổi nhận được từ repository
....
>>>>>>> .r2 #conflict marker

Xác định nội dung tập tin đụng độ:
svn resolve --accept option filename
option
base: chọn phiên bản trước khi sửa đổi
mine-full: chỉ chấp nhận những sửa đổi của bạn
theirs-full: chỉ chấp nhận những sửa đổi lấy về từ repository
working: dùng tập tin trộn đã được sửa đổi

6.12.8 Xác định sự sửa đổi - commit

Cú pháp:svn commit -m «note for your changes»

- Cập nhật phiên bản làm việc của bạn lên repository
- SVN kiểm tra xem có tập tin nào bị sửa đổi bởi người dùng khác không ?
- Nếu có, cần thực thi lệnh update để cập nhật; xử lý đụng độ nếu có và thực hiện lại lệnh commit

6.12.9 Xem lại nhật ký của repository

Thông tin tổng quát về những sửa đổi:

```
 $ svn log
------------------------------------------------------------------------
r3 | sally | 2008-05-15 23:09:28 -0500 (Thu, 15 May 2008) | 1 line
Added include lines and corrected # of cheese slices.
------------------------------------------------------------------------
r2 | harry | 2008-05-14 18:43:15 -0500 (Wed, 14 May 2008) | 1 line
Added main() methods.
------------------------------------------------------------------------
```

Chi tiết về một lần sửa đổi:

```
 $ svn log -r 8 -v
------------------------------------------------------------------------
r8 | sally | 2008-05-21 13:19:25 -0500 (Wed, 21 May 2008) | 1 line
Changed paths:
   M /trunk/code/foo.c
   M /trunk/code/bar.h
   A /trunk/code/doc/README
Frozzled the sub-space winch.
------------------------------------------------------------------------
```

Xem lại những thay đổi cục bộ:

```
 $ svn diff
Index: rules.txt
===================================================================
--- rules.txt     (revision 3)
+++ rules.txt   (working copy)
@@ -1,4 +1,5 @@
 Be kind to others
 Freedom = Responsibility
 Everything in moderation
-Chew with your mouth open
+Chew with your mouth closed
+Listen when others are speaking
```

So sánh phiên bản làm việc với một revision của repository:

```
$ svn diff -r 3 rules.txt
Index: rules.txt
===================================================================
--- rules.txt     (revision 3)
+++ rules.txt   (working copy)
@@ -1,4 +1,5 @@
```

```
 Be kind to others
 Freedom = Responsibility
 Everything in moderation
-Chew with your mouth open
+Chew with your mouth closed
+Listen when others are speaking
```

So sánh 2 revision trên repository:

```
$ svn diff -r 2:3 rules.txt
Index: rules.txt
===================================================================
--- rules.txt      (revision 2)
+++ rules.txt   (revision 3)
@@ -1,4 +1,4 @@
 Be kind to others
-Freedom = Chocolate Ice Cream
+Freedom = Responsibility
 Everything in moderation
 Chew with your mouth open
```

Xem lại một phiên bản:

```
$ svn cat -r 2 rules.txt
Be kind to others
Freedom = Chocolate Ice Cream
Everything in moderation
Chew with your mouth open
```

6.12.10 Liệt kê nội dung một thư mục trên repository – list

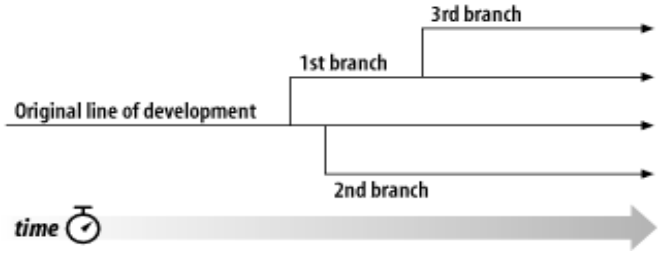
```
$ svn list http://svn.collab.net/repos/svn
README
branches/
clients/
tags/
trunk/
$ svn list -v http://svn.collab.net/repos/svn
20620  harry   1084    Jul   13  2006  README
23339  harry                    Feb 04 01:40   branches/
....
```

Ghi nhớ khi sử dụng subversion

- Luôn luôn nhập các ghi chú
- Luôn luôn nhóm các thay đổi lại với nhau
- Đừng bao giờ commit các tập tin đối tượng (object) hoặc các tập tin thực thi (executable)
- Đừng bao giờ commit một phiên bản làm việc biên dịch không thành công (broken build)

6.13 Giới thiệu về nhánh (Branch)

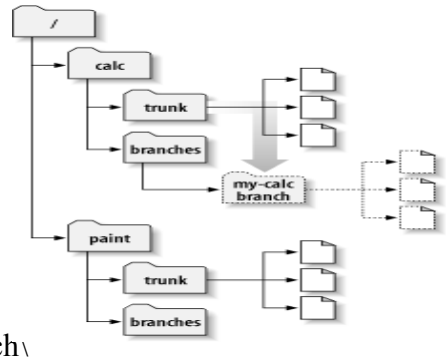
Một nhánh được hình thành như là một bản sao của một dòng phát triển chính -DPTC -(original line of development), từ đó nó tiến triển và có lịch sử riêng. Một nhánh phát triển độc lập với các nhánh khác, tuy nhiên nó vẫn chia sẻ một lịch sử chung



SVN cho phép:

- Tạo nhánh bằng cách sao chép dữ liệu từ DPTC
- Ghi nhớ những bản sao có liên quan nhau
- Nhân bản sự sửa đổi từ nhánh này sang nhánh khác
- Tạo những phần trong phiên bản làm việc phản ánh các nhánh khác nhau

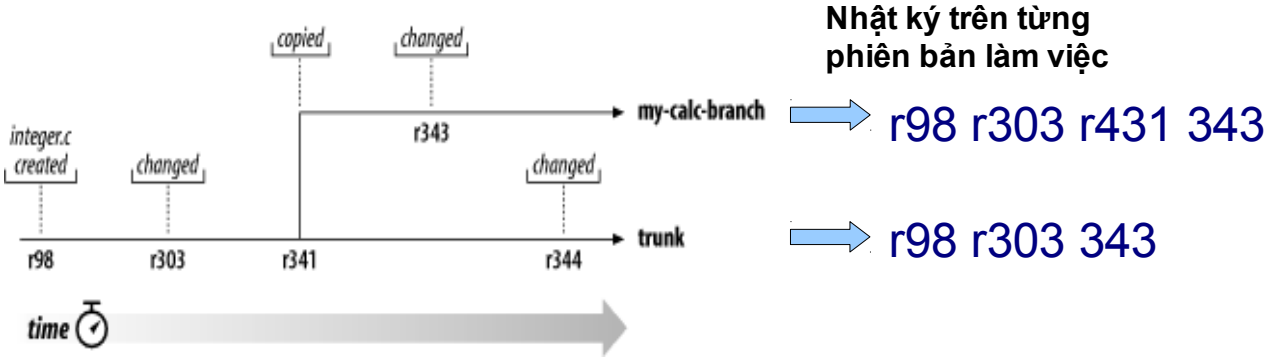
Tạo nhánh:
calc và paint là thư mục của các dự án
trunk: dòng phát triển chính
branches: chứa các nhánh của dự án
Dùng lệnh svn copy để tạo một nhánh




```
$ svn copy http://svn.example.com/repos/calc/trunk\
        http://svn.example.com/repos/calc/branches/my-calc-branch\
          -m "Creating a private branch of /calc/trunk."
Committed revision 341.
```

Sử dụng nhánh:

- Sử dụng lệnh checkout để tạo một phiên bản làm việc từ thư mục của nhánh trên repository
- Sửa đổi, cập nhật, công bố sửa đổi trên phiên bản làm việc của nhánh một cách bình thường
  - $ svn checkout http://svn.example.com/repos/calc/branches/my-calc-branch



Trong phiên bản làm việc của nhánh, thực hiện lệnh

svn merge original-line-development-url

Ví dụ:

```
$ pwd
/home/user/my-calc-branch
$ svn merge url-project/trunk
--- Merging r345 through r356 into '.':
U    button.c
```

Xử lý đụng độ nếu có, kế tiếp thực hiện lệnh commit

Cập nhật những sửa đổi trên một nhánh vào DPTC:

Đồng bộ nhánh với dòng phát triển chính

Cập nhật phiên bản làm việc của DPTC

svn update hoặc svn checkout project-url/trunk

Cập nhật phiên bản làm việc của DPTC với những sửa đổi của nhánh

svn merge --reintegrate project-url/branches/my-calc-branch

Công bố những thay đổi trên phiên bản làm việc của DPTC

svn commit -m "Merge my-calc-branch back into trunk!"

6.14 Nhãn

Nhãn là ảnh chụp (snapshort) của một dự án tại một thời điểm nào đó.

Ví dụ: nhãn «release 1.0» được gán cho trạng thái của dự án sau một lần commit tương ứng với revision 901

Nhãn được tạo ra tương tự như cách tạo một nhánh với một qui ước là nội dung của thư mục nhãn sẽ không giờ bị thay đổi, tức không bao giờ sửa đổi và commit một phiên bản làm việc của thư mục nhãn

Tạo nhãn:

Nhãn được tạo ra bằng cách sử dụng lệnhsvn copy để sao chép DPTC trong thư mục truck của dự án

Nhãn thường được chứa trong thư mục tags

Ví dụ

```
$ svn copy http://svn.example.com/repos/calc/trunk \
    http://svn.example.com/repos/calc/tags/release-1.0 \
    -m "Tagging the 1.0 release of the 'calc' project."Committed revision 902.
```

Tài liệu tham khảo

1. http://sites.google.com/site/ptpmmnm
2. http://sites.google.com/site/nbhung/open-source
3. Http://www.ctu.edu.vn/pmnm/
4. Giáo trình Unix của http:Galaxy CD
5. Diomidis Spinesllis, The Open Source Perspective. 2003
6. Karl Fogel, Producing Open Source Software. 2005
7. Andrew M. St. Laurent, Open Source and Free Software Licensing, 2004
8. James Guérin (traduction), Le Logiciel libre. 2001
9. Ron Goldman, Richard P. Gabriel, Innovation Happens Elsewhere – Open Source as Business Strategy. 2005
10. TeachingOpenSource.org
11. Course: Open Source Development and Distribution of Digital Information: Economic, Legal, and Social Perspectives
12. http://rosetta.sims.berkeley.edu:8085/sylvia/f05/view/?course=296A-2&view=complete
13. Special Topics Course in Open Source Development
14. http://classes.engr.oregonstate.edu/eecs/winter2007/cs419-002/pmwiki/pmwiki.php/Main/HomePage
15. 100 Free Open Courseware Classes About Open Source Everything
16. http://www.bschool.com/blog/2008/100-free-open-courseware-classes-about-open-source-everything/
17. HOWTO: Pick an open source license (part 1)
18. http://blogs.zdnet.com/Burnette/?p=130
19. OPEN SOURCE GOD: 480+ Open Source Applications
20. http://mashable.com/2007/09/23/open-source/
21. Diomidis Spinesllis. The Open Source Perspective. 2003
22. Karl Fogel. Producing Open Source Software. 2005
23. Andrew M. St. Laurent. Open Source and Free Software Licensing, 2004
24. James Guérin (traduction). Le Logiciel libre. 2001
25. http://www.anyware-tech.com . Programme de formation open source. 2006
26. Ron Goldman, Richard P. Gabriel, Innovation Happens Elsewhere – Open Source as Business Strategy. 2005
27. Greg Kroah-Hartman, SuSE Labs / Novell Inc., Jonathan Corbet, LWN.net, Amanda McPherson. The Linux Foundation, Linux Kernel Development - How Fast it is Going, Who is Doing It, What They are Doing, and Who is Sponsoring It: An August 2009 Update
28. Robert Love. Linux Kernel Development Second Edition, 2005
29. Daniel P. Bovet, Marco Cesati. Understanding the Linux Kernel, 2nd Edition, 2002
30. Ivan Bowman. Conceptual Architecture of the Linux Kernel, 1998
31. Linux Knowledge Base and Tutorial, http://www.linux-tutorial.info/index.php
32. Vivek G. Gite. Linux Shell Scripting Tutorial v1.05r3. A Beginner's handbook, 1999-2002
33. Mark G. Sobell. A practical guid to Ubuntu Linux, 2007