

Công nghệ mã nguồn mở

GV. Nguyễn Thị Mai Trang

1

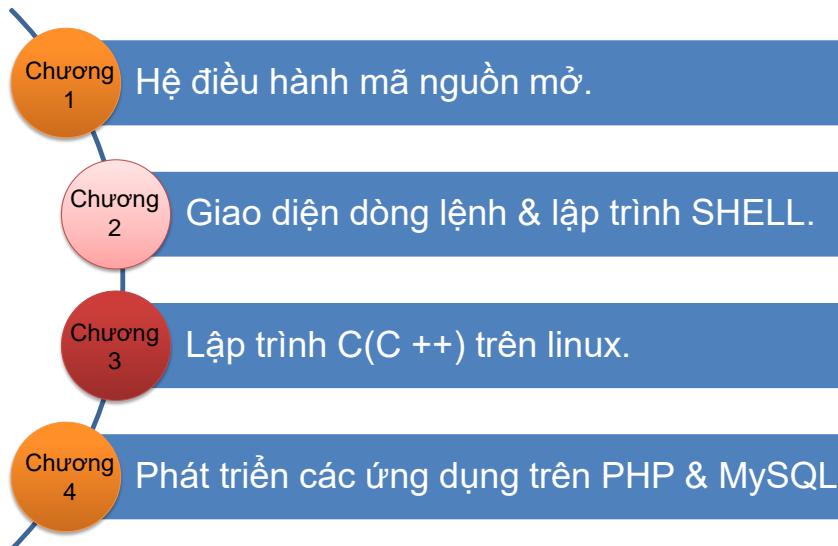


Mục tiêu

-
- **Kiến thức:**
 - Các kiến thức cơ bản về thiết lập và quản trị mạng cục bộ trên HĐH Linux.
 - Có khả năng sử dụng ngôn ngữ shell để lập trình.
 - Có khả năng sử dụng ngôn ngữ gcc để lập trình
 - Có khả năng sử dụng ngôn ngữ PHP và MySQL để lập trình các trang web đơn giản
 - **Kỹ năng:**
 - Lập trình trên môi trường Ubuntu
 - **Thái độ:**
 - Có đạo đức tốt, tác phong tốt.
 - Có tính tự học, tự trao đổi kiến thức.
 - Tự tin và yêu thích đón với những bài toán về lập trình.

2

Nội dung



3

Chương 1

HỆ ĐIỀU HÀNH MÃ NGUỒN MỞ

Nội dung

1. Tổng quan.
2. Cài đặt
3. Thao tác cơ bản với các phần mềm trên HĐH Linux
4. Cài đặt phần mềm.
5. Quản trị người dùng và nhóm.
6. Hệ thống tập tin và thư mục.

5

1.1 Tổng quan

- Phần mềm và vấn đề bản quyền phần mềm
- Các loại phần mềm
- Phong trào phần mềm tự do
- Phần mềm mã nguồn mở
- Giới thiệu Hệ điều hành Linux

6

Phần mềm và vấn đề bản quyền phần mềm

- Một phần mềm thuộc về một chủ sở hữu
- Chủ sở hữu có toàn quyền đối với phần mềm và quyết định mức độ sử dụng và khai thác trên phần mềm thuộc quyền sở hữu
- Khi muốn sử dụng một phần mềm, người sử dụng phải có một giấy phép sử dụng (license) của phần mềm đó

7

Phần mềm và vấn đề bản quyền phần mềm

- Giấy phép sử dụng phần mềm:
 - Là một bản hợp đồng cho phép người sử dụng khai thác phiên bản phần mềm, qui định về những khả năng có thể sử dụng đối với phần mềm
- Các tiêu chí phân loại phần mềm dựa trên giấy phép
 - Khả năng phân phối lại (Distribution Possibility)
 - Truy cập vào mã nguồn (Accessibility to source code)
 - Phí sử dụng (Free)

8

Phần mềm và vấn đề bản quyền phần mềm

- Phân loại theo Khả năng phân phối lại
 - Quyền được phép sao chép và phân phối lại phiên bản phần mềm mà bạn đang có trong tay (đã có giấy phép sử dụng) hay không?
- Phân loại theo Truy cập vào mã nguồn
 - Cho phép xem mã nguồn, sử dụng, sửa mã nguồn phần mềm
- Phân loại theo Phí sử dụng
 - Có phải trả tiền cho chủ sở hữu phần mềm hay không?

9

Các loại phần mềm

-
- Phần mềm thương mại
 - Phần mềm miễn phí - trả một phần
 - Phần mềm mã nguồn mở

10

Phần mềm thương mại

- Bản quyền của phần mềm thương mại chỉ cho phép người sử dụng phần mềm theo những ràng buộc ghi trên giấy phép
- Bản quyền loại này rất bị hạn chế
- Trong trường hợp phần mềm có lỗi hay các chức năng hoạt động không tốt
 - Chờ chủ sở hữu sửa lỗi
 - Các nhà sản xuất phần mềm không nhiệt tình sửa lỗi hoặc thực hiện trong thời gian rất lâu
 - Người sử dụng có thể phải trả thêm phí cập nhật
 - Người sử dụng không có một phương tiện gì để thúc đẩy tiến trình sửa đổi và cập nhật các phần mềm thương mại

11

Phần mềm miễn phí - trả một phần

- Phần mềm miễn phí (Freeware) và phần mềm trả một phần (Shareware) không là phần mềm mã nguồn mở
 - Vẫn có chủ sở hữu
 - Được phân phối một cách tự do
- Phần mềm trả một phần: sau một thời gian đã định, người sử dụng phải trả thêm tiền để sử dụng tiếp

12

Phần mềm mã nguồn mở

- Phần mềm mã nguồn mở phải hội đủ các yếu tố sau:
 - Được phân phối đến người sử dụng cùng với mã nguồn có thể được sửa đổi
 - Nó có thể được phân phối lại mà không bị một ràng buộc nào khác
 - Có thể phân phối cả những thay đổi trên mã nguồn gốc

13

Các loại phần mềm

	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)	Đôi khi	Không	Có
Phần mềm trả một phần (Shareware)	Đôi khi	Không	Không
Phần mềm mã nguồn mở (Open Source Software)	Được phép	Có	Đôi khi

14

Phong trào phần mềm tự do

- Nhằm tạo ra những phần mềm tự do (Free Software): tự do chia sẻ, nghiên cứu và sửa đổi.
- 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 mềm tự do (FSF-Free Software Foundation) năm 1985



Richard M. Stallman

15

Phong trào phần mềm tự do

- Phần mềm tự do đề cập đến sự tự do, không đề cập đến vấn đề chi phí/giá cả.
- Sự tự do bao gồm 4 yếu tố
 - Tự do chạy chương trình, cho bất cứ mục đích nào.
 - Tự do tìm hiểu cách hoạt động của chương trình, và tự do sửa đổi nó. (Quyền truy cập mã nguồn là điều kiện tiên quyết cho quyền tự do này.)
 - Tự do tái phân phối bản sao.
 - Tự do cải tiến chương trình, và phát hành những gì cải tiến ra công cộng. (Quyền truy cập mã nguồn là điều kiện tiên quyết cho quyền tự do này.)

16

Phong trào phần mềm tự do

- **Copyright:** bảo vệ quyền tác giả
- **Copyleft:**
 - Làm cho một chương trình là phần mềm tự do
 - Yêu cầu tất cả những phiên bản sửa đổi hay mở rộng của chương trình cũng phải tự do.
- Giấy phép “**GNU General Public License**”
 - Viết tắt “**GNU GPL**”
 - Cụ thể hóa khái niệm Copyleft
 - Dùng cho phần lớn các sản phẩm của dự án GNU

17

Phong trào phần mềm tự do

- Giấy phép mã nguồn mở: là các giấy phép bản quyền dành cho các phần mềm mã nguồn mở.
- GNU (GNU General Public License /GNU GPL/GPL)
- Nội dung các giấy phép GNU thay đổi tùy theo từng phiên bản
 - Version 1- GPL v1- 1989
 - Version 2- GPL v2- 1991
 - Version 2- LGPL v2 (Library General Public License) - 1991
 - Version 2.1- LGPL v2.1 (Lesser General Public License) - 1999
 - Version 3- GPL v3- 2007



18

Phong trào phần mềm tự do

- Nội dung các giấy phép GNU

- Xử lí vi phạm:

- Người vi phạm bị tước quyền sử dụng giấy phép GNU.

- Quyền lợi:

- Quyền được sao chép và phân phối chương trình
- Quyền được yêu cầu trả phí cho việc phân phối đó.
- Quyền được thay đổi chương trình để sử dụng cho mục đích cá nhân.
- Quyền được phân phối bản đã được thay đổi đó.

19

Phong trào phần mềm tự do

- Nội dung các giấy phép GNU

- Nghĩa vụ:

- Khi sao chép và phân phối chương trình, phải đính kèm các thông báo về bản quyền gốc và không nhận bảo hành (trừ trường hợp có văn bản thêm về quy định bảo hành.)
- Khi phân phối bản đã được thay đổi bởi chính mình, phải chú thích rõ đó là bản đã được thay đổi, các thành phần được thay đổi, và áp dụng giấy phép GNU cho bản đã được thay đổi đó.
- Khi phát hành chương trình phải công khai mã nguồn của chương trình, đồng thời phải công bố mã nguồn của chương trình trong tối thiểu 3 năm mà không được đòi một khoản phí nào từ những người yêu cầu mã nguồn trừ chi phí vận chuyển hay tương đương.

20

Phần mềm mã nguồn mở

- Sáng kiến mã nguồn mở OSI (Open Source Initiative—www.opensource.org)
 - Là tổ chức phi lợi nhuận được thành lập năm 1998 bởi Eric Raymond and Bruce Perens
 - Thay thế khái niệm **Phần mềm tự do** (Free Software) bằng khái niệm **Phần mềm mã nguồn mở** (Open Source Software) để tránh sự hiểu nhầm:
 - Ý nghĩa **tự do** với **miễn phí** của từ **Free** tiếng Anh
 - Phần mềm tự do là không thương mại

21

Phần mềm mã nguồn mở

- OSI (Open Source Initiative—www.opensource.org)



Eric Raymond and Bruce Perens

22

Phần mềm mã nguồn mở

- Là phần mềm dưới dạng mã nguồn, 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 được (hoặc không được) làm đối với mã nguồn

23

Phần mềm mã nguồn mở

- **Ưu điểm:**
 - PMMNM được phát triển bởi cộng đồng → dễ phát hiện và sửa lỗi
 - Mỗi người có thể xem xét và cải tiến các công việc được thực hiện bởi những người khác.
 - Mỗi người chỉ tập trung vào lĩnh vực chuyên sâu của mình
 - 500 lập trình viên làm việc với thời gian khác nhau, tập trung vào lĩnh vực chuyên sâu của mình → tốt hơn 50 lập trình viên làm việc toàn thời gian.
 - Cách phân phối của PMMNM 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

24

Phần mềm mã nguồn mở

- Mười tiêu chí của OSI (Open Source License):

1. **Tự do phân phối lại** (free Redistribution)

- Bản quyền sẽ không hạn chế bất cứ ai bán hoặc cho phần mềm (không yêu cầu tiền bản quyền hay một chi phí nào).

2. **Mã nguồn** (Source code)

- Chương trình phải được phân phối cùng với mã nguồn, được công bố bằng những phương tiện công cộng với không có hoặc với một chi phí hợp lý nhất.

25

Định nghĩa PMMNM của OSI (tt)

3. **Sản phẩm kế thừa** (Derived Works)

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

26

Định nghĩa PMMNM của OSI (tt)

Mười tiêu chí của OSI

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, ngoại trừ việc 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).
- 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

27

Định nghĩa PMMNM của OSI (tt)

Mười tiêu chí của OSI

5. Không phân biệt đối xử giữa các cá nhân và các nhóm

(No Discrimination Against Persons or Groups)

6. Không phân biệt đối xử với mục đích sử dụng

(No Discrimination Against Fields of Endeavor)

7. Phân phối giấy phép (Distribution of license)

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

28

Định nghĩa PMMNM của OSI (tt)

Mười tiêu chí của OSI

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 không bị phụ thuộc vào việc chương trình là thành phần của một bản phân phối phần mềm cụ thể.
- Nếu phần mềm được trích ra từ bản phân phối, được sử dụng hoặc phân phối lại với những điều khoản của giấy phép của chương trình thì người sử dụng bản phân phối cũng có được các quyền lợi giống như những quyền lợi được đưa ra theo bản phân phối phần mềm gốc.

29

Định nghĩa PMMNM của OSI (tt)

Mười tiêu chí của OSI

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 yêu cầu tất cả các phần mềm khác được phân phối trên cùng một phương tiện thì phải là phần mềm mã nguồn mở.

10. Giấy phép phải trung lập về mặt công nghệ (License Must Be Technology-Neutral)

- Giấy phép không được có quy định dành cho một công nghệ riêng hay một kiểu giao diện nào đó.

30

Hệ điều hành Linux

- Là một hệ điều hành được phát triển dựa trên hệ điều hành Minix bởi Linus Torvalds năm 1991
- Là hệ điều hành tương tự Unix, tự do :
 - Miễn phí, hoặc phí khiêm tốn.
 - Sử dụng tự do.
- Là hệ điều hành thông dụng có khả năng chạy được trên hầu hết các thiết bị phần cứng chính.

31

Hệ điều hành Linux

- **Đặc điểm của linux**
 - Là hệ điều hành mã nguồn mở, miễn phí.
 - Đa người dùng (multiuser)
 - Đa nhiệm (multitasking)
 - Hỗ trợ các định dạng hệ thống tập tin khác nhau
 - Khả năng hỗ trợ mạng
 - Độc lập kiến trúc
 - Bảo mật
 - ...

32

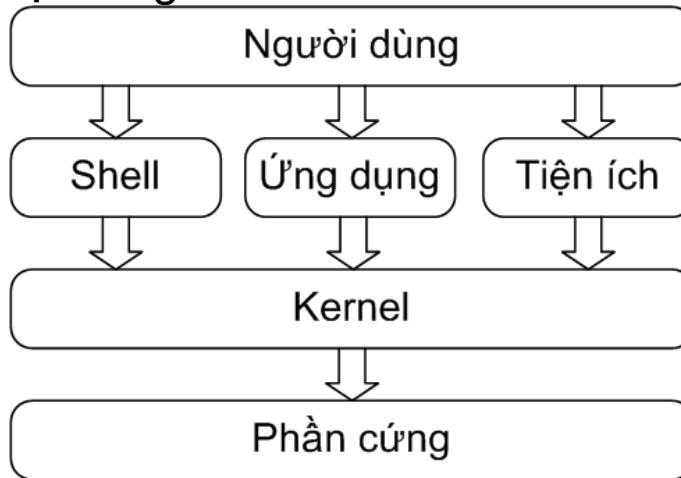
Hệ điều hành Linux

- Một số hạn chế của linux
 - Chưa thân thiện với người dùng
 - Cài đặt còn phức tạp
 - Phần mềm ứng dụng còn khó thao tác
 - Thiếu trợ giúp kỹ thuật
 - Còn dựa nhiều vào giao tiếp dòng lệnh
 - Thiếu hỗ trợ phần cứng

33

Hệ điều hành Linux

- Kiến trúc hệ thống Linux

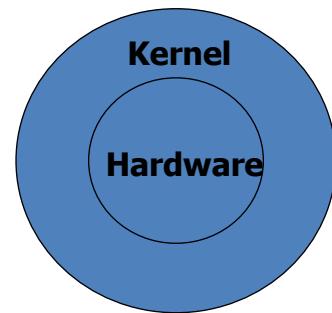


34

Hệ điều hành Linux

- **Linux Kernel:**

- Là trung tâm điều khiển của hệ điều hành Linux, chứa các mã nguồn điều khiển hoạt động của toàn bộ hệ thống.
- Là cầu nối giữa chương trình ứng dụng và phần cứng.
- Lập lịch, phân chia tài nguyên cho các tiến trình.
- Sử dụng không gian đĩa hoán đổi (swap space) để lưu trữ dữ liệu xử lý của chương trình.

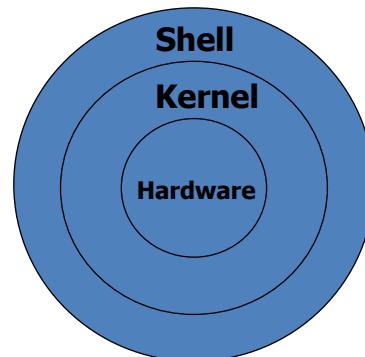


35

Hệ điều hành Linux

- **Shell:**

- Cung cấp tập lệnh cho người dùng thao tác với kernel để thực hiện công việc.
- Có nhiều loại shell trong Linux :
 - C Shell (%)
 - Bourne Shell (\$)
 - Korn Shell (\$)
 - ...



36

Hệ điều hành Linux

- Bản phân phối Linux:
 - Cấu trúc hệ thống tập tin
 - Chương trình cài đặt
 - Các tiện ích và chương trình ứng dụng
 - Trình quản lý và cập nhật gói phần mềm
 - Các sửa đổi của riêng nhà sản xuất
 - Tài liệu hướng dẫn, hỗ trợ người dùng

37

Hệ điều hành Linux

- Một số phiên bản linux

- Red Hat
- Mandrake
- SuSe
- Debian
- Slackware
- Gentoo
- Knoppix
- Lycoris
- Xandros
- Lindows



Lycoris



xandros

Xandros



Lindows

<http://www.distrowatch.com>



Knoppix

Knoppix



Knoppix

Một số phần mềm mã nguồn mở

- Internet
 - Apache, Sendmail, BIND, Squid, Wu-ftp, Inn
- Database
 - Postgresql, mySQL
- Desktop
 - KDE, GNOME
- Office
 - OpenOffice, Koffice, Abiword
- Graphics
 - GIMP

39

1.2. Cài đặt Linux

- Khái niệm phân vùng
 - Đĩa cứng được phân ra nhiều vùng khác nhau gọi là partition.
 - Ví dụ : Tên phân vùng trên MS-DOS/Windows: C:, D:, E:
 - Mỗi đĩa chỉ chia được tối đa 4 partition chính (Primary)
 - Master Boot Record – MBR
 - Phân loại:
 - Primary
 - Extended
 - Logical

40

Cài đặt Linux

- Yêu cầu phân vùng Linux

- Unix lưu trữ file trên các hệ thống file (filesystem)
 - /usr, /var, /home
- Hệ thống file chính: root filesystem “/”
- Mỗi hệ thống file có thể nằm trên một phân vùng riêng biệt. Ít nhất cần phải có hệ thống file “/”
- Nên sử dụng nhiều phân vùng khác nhau cho các hệ thống file.

41

Cài đặt Linux

- Ký hiệu đĩa

- Mỗi ổ đĩa được khai báo trong thư mục : /dev/

- Ký hiệu ổ đĩa :

- Đĩa mềm : fd được khai báo /dev/fd0
- Đĩa cứng : hd được khai báo /dev/hda
- Đĩa SCSI : sd được khai báo /dev/sda

- Ký tự a, b, c để xác định các ổ đĩa cùng loại khác nhau

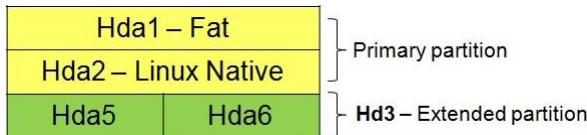
Ký hiệu	Mô tả
Hda	Primary Master
Hdb	Primary Slave
Hdc	Secondary Master
Hdd	Secondary Slave
Sda	First SCSI disk

42

Cài đặt Linux

- Ký hiệu partition: dùng các số đi kèm để xác định partition.
 - Primary partition và extented partition đánh số từ 1 → 4
 - Các logical partition được đánh số từ 5 trở lên

- Ví dụ :



- Cấu trúc đĩa thứ nhất gồm có hai partition chính và một partition mở rộng.
 - Partition chính gồm : hda1 và hda2
 - Partition mở rộng hda3 có 2 partition logic gồm : hda5 và hda6

43

Cài đặt Linux

- Các bước cài đặt
 - Phần cứng: đáp ứng yêu cầu tối thiểu
 - Chuẩn bị: CD/DVD tùy theo phiên bản cài đặt, download từ website
 - Tiến hành cài đặt

44

Cài đặt Linux

- Ví dụ cài Ubuntu 14.04

- Download DVD, iso tại <http://www.ubuntu.com/download>.
- Yêu cầu hệ thống:
 - 1 GHz x86 processor (Pentium 4 or better)
 - 1 gb of system memory (ram)
 - 5 gb of disk space (at least 15 gb is recommended)
 - Video support capable of 1024×768 resolution
 - Audio support
 - An Internet connection (highly recommended, but not required)

45

Cài đặt Linux

- Khởi động hệ thống

- Bước 1 : PC khởi động.
- Bước 2 : BIOS tìm đĩa chứa trình khởi động.
- Bước 3 : Và chuyển quyền điều khiển cho MBR.
- Bước 4 : MBR nạp trình quản lý khởi động và chuyển quyền điều khiển cho trình quản lý.
- Bước 5 : Hiển thị Operating Systems Kernel.
- Bước 6 : Xác định mức hoạt động.
- Bước 7 : Thực thi các tập tin script được chỉ định cho từng mức hoạt động.
- Bước 8 : Hệ thống sẽ chạy chương trình login để yêu cầu đăng nhập cho từng người dùng

46

1.3. Các phần mềm tiện ích

- Terminal
- Trình soạn thảo Vi
- Trình tiện ích mail
- Trình tiện ích mc (Midnight Commander)
- ...

47

Terminal

- Cửa sổ dòng lệnh cho phép nhập các lệnh thao tác với hệ thống
- Khởi động: Ctrl - Alt – T
- Một số lệnh thông dụng:
 - clear: xóa màn hình
 - apt-get install tenphanmem: cài đặt phần mềm
 - man tenlenth: xem thông tin trợ giúp của lệnh



48

Terminal

- Nhóm lệnh hiển thị thông tin hệ thống
 - arch/ uname -m: hiển thị cấu trúc của máy (VD i686)
 - uname -r: hiển thị phiên bản kernel đang sử dụng
 - dmidecode -q: Hiển thị hệ thống phần cứng (SMBIOS / DMI)
 - hdparm -i /dev/hda hoặc hdparm -i /dev/sda: hiển thị thông tin ổ cứng ATA/SATA
 - lspci -tv: hiển thị thiết bị PCI
 - lsusb -tv: hiển thị thiết bị USB
 - date: hiển thị ngày hệ thống
 - cal y hiển thị lịch năm y
 - date chuoingaygio (MonthDayhoursMinutesYear.Seconds): thiết lập ngày và giờ
Ví dụ: date 101916002017.00 → 19/10/2017 - 16h30'00s
 - clock -w lưu thay đổi ngày trên BIOS

49

Terminal

- Nhóm lệnh Shutdown/Restart/Logout:
 - shutdown -h now: tắt máy
 - init 0: tắt máy
 - telinit 0: tắt máy
 - shutdown -h hours:minutes: tắt máy sau thời gian đợi
 - shutdown -c :hủy lệnh tắt máy
 - shutdown -r now: khởi động lại
 - reboot: khởi động lại
 - logout: rời khỏi phiên làm việc
 - su [username]: Chuyển sang tài khoản username

50



Terminal

- Các lệnh khác:
 - history: Xem danh sách các lệnh đã gõ
 - ifconfig: Xem địa chỉ IP
 - startx: Khởi động chế độ đồ họa (Start graphic mode)
 - exit: thoát
 - env: xem các biến môi trường
 - hostname: xem tên máy
 - hostname tenmay: đổi tên máy tính tạm thời sang tenmay
 - whoami hoặc who: Xem user hiện hành
 - finger: Xem thông tin user hiện hành

51



Trình soạn thảo Vi

52

Trình soạn thảo Vi (tt)

- Có 3 chế độ (mode) làm việc:
 - Lệnh (command mode) – phím nhập vào là lệnh
 - Soạn thảo (edit mode)
 - Dòng lệnh (“:” mode) – thực hiện dòng lệnh sau “:”
- Nhấn <ESC> để thoát khỏi chế độ hiện tại
- Hầu hết các lệnh là phân biệt hoa thường

53

Trình soạn thảo Vi (tt)

- Chế độ soạn thảo:
 - a chèn ngay sau vị trí con trỏ
 - A chèn vào cuối dòng
 - i chèn ngay trước vị trí con trỏ
 - I chèn vào đầu dòng
 - o chèn một hàng mới dưới vị trí con trỏ
 - O chèn một hàng mới trên vị trí con trỏ
 - r thay thế ký tự tại vị trí con trỏ
 - R thay thế bắt đầu từ vị trí con trỏ
 - S thay thế dòng hiện tại
 - C thay thế từ vị trí con trỏ đến cuối dòng

54



Trình soạn thảo Vi (tt)

- Di chuyển theo ký tự
 - Sử dụng phím mũi tên để di chuyển con trỏ từng ký tự (tùy hỗ trợ của terminal)
 - h, j, k, l thay thế cho các phím mũi tên
 - [n]h dịch trái [n] ký tự
 - [n]j dịch xuống [n] ký tự
 - [n]k dịch lên [n] ký tự
 - [n]l dịch phải [n] ký tự
 - Lưu ý: lệnh có thể thêm chữ số đứng trước để chỉ số lần lặp lại lệnh đó

55



Trình soạn thảo Vi (tt)

- Di chuyển theo màn hình
 - Sử dụng các phím PgUP, PgDown để cuộn 1 khung màn hình (tuỳ hỗ trợ của terminal)
 - ctrl + F cuộn xuống 1 khung màn hình
 - ctrl + B cuộn lên 1 khung màn hình
 - ctrl + D cuộn xuống 1/2 khung màn hình
 - ctrl + U cuộn lên 1/2 khung màn hình
 - (không phân biệt phím hoa thường)

56

Trình soạn thảo Vi (tt)

- Di chuyển theo từ, dòng
 - G đến dòng cuối file
 - [n]G đến cuối file hoặc dòng thứ [n]
 - :n đến dòng thứ n
 - gg đến dòng đầu file
 - \$ về cuối dòng (End)
 - ^ về đầu dòng (Home)
 - [n]w tới [n] từ (word)
 - [n]b lùi [n] từ
 - e về cuối từ

57

Trình soạn thảo Vi (tt)

- Nhóm lệnh xóa:
 - [n]x xoá [n] ký tự tại vị trí con trỏ (Del)
 - X xoá ký tự trước vị trí con trỏ (BkSpc)
 - [n]dw xoá [n] từ
 - D xoá từ vị trí con trỏ đến cuối dòng
 - [n]dd xoá [n] dòng từ vị trí con trỏ
 - d\$ xoá đến cuối dòng
 - dG xoá đến cuối file
 - Văn bản bị xoá luôn được lưu tạm trong một bộ đệm (ý nghĩa giống như “cut”)

58

Trình soạn thảo Vi (tt)

- Copy, cut, paste:

- [n]yw copy [n] từ vào bộ đệm (yank)
- [n]yy copy (yank) [n] dòng vào bộ đệm
- [n]dw cắt [n] từ vào bộ đệm
- [n]dd cắt [n] dòng vào bộ đệm
- p dán từ bộ đệm vào sau con trỏ
- P dán từ bộ đệm vào trước con trỏ

59

Trình soạn thảo Vi (tt)

- Một số lệnh đặc biệt

- J nối dòng hiện tại và dòng kế
- u undo thay đổi cuối cùng
- U khôi phục dòng như trước khi bị sửa đổi
- ^R redo thay đổi sau đó
- . lặp lại thay đổi cuối cùng
- /[pattern] tìm kiếm theo hướng tới
- ?[pattern] tìm kiếm theo hướng lùi
- n lặp lại tìm kiếm theo cùng chiều
- N lặp lại tìm kiếm theo ngược chiều

60

Trình soạn thảo Vi (tt)

- Lưu và thoát tập tin

- ZZ ghi nội dung bộ đệm ra file và thoát
- x ghi nội dung bộ đệm ra file và thoát
- :w ghi nội dung bộ đệm ra file
- :q! huỷ phiên làm việc hiện tại và thoát
- :wq ghi nội dung bộ đệm ra file và thoát
- ! buộc thi hành lệnh (force operation)

61

Trình tiện ích mail

- Cài đặt: \$apt-get install mailutils
- Lệnh **\$mail** hiển thị nội dung các email, sau mỗi email hiện lên dấu ?, người dùng nhập các thao tác sau:
 - +: hiển thị mail kế tiếp
 - p: in nội dung mail
 - s [filename]: lưu mail vào filename
 - d: xóa mail
 - q: thoát
 - ![lệnh]

62

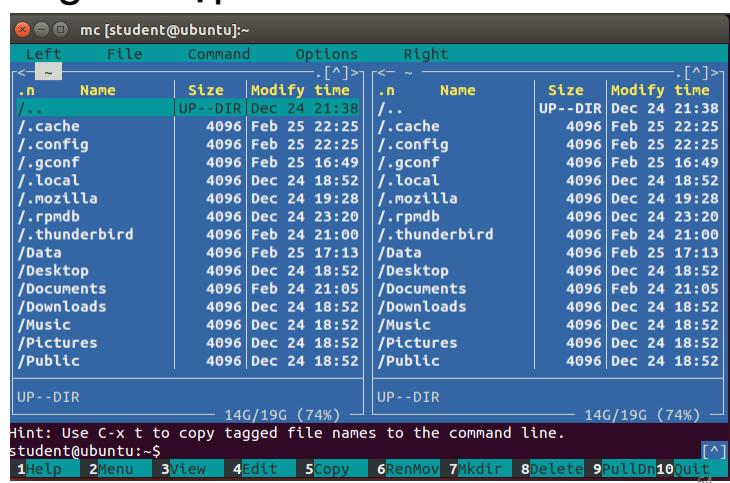
Trình tiện ích mail (tt)

- Gửi mail:
 - **\$mail** nguoinhan
 - Nội dung
 - Ctrl-D

63

Trình tiện ích mc (Midnight Commander)

- Cho phép thao tác dễ dàng với tập tin
- Cài: **apt-get install mc**
- **\$mc**



The screenshot shows the mc (Midnight Commander) interface. It has two main panes: the left pane and the right pane. Both panes display a list of files and directories with columns for Name, Size, and Modify time. The left pane shows the current directory structure, while the right pane shows a mirror image. At the bottom of each pane, there is a status bar indicating the total size and percentage of the current directory.

Left	File	Command	Options	Right
.[^]>	File	Command	Options	.[^]>
..	Left	File	Command	..
./	File	Command	Options	./
/.cache	Name	Size	Modify time	/.cache
/.config		4096	Feb 25 22:25	/.config
/.gconf		4096	Feb 25 16:49	/.gconf
/.local		4096	Dec 24 18:52	/.local
/.mozilla		4096	Dec 24 19:28	/.mozilla
/.rpmbdb		4096	Dec 24 23:20	/.rpmbdb
/.thunderbird		4096	Feb 24 21:00	/.thunderbird
/Data		4096	Feb 25 17:13	/Data
/Desktop		4096	Dec 24 18:52	/Desktop
/Documents		4096	Feb 24 21:05	/Documents
/Downloads		4096	Dec 24 18:52	/Downloads
/Music		4096	Dec 24 18:52	/Music
/Pictures		4096	Dec 24 18:52	/Pictures
/Public		4096	Dec 24 18:52	/Public
UP--DIR				UP--DIR

Hint: Use C-x t to copy tagged file names to the command line.
 student@ubuntu:~\$ [^]
 1Help 2Menu 3View 4Edit 5Copy 6RenMov 7Mkdir 8Delete 9PullDn 10Quit
 04

1.4. Cài đặt phần mềm

- Các gói phần mềm được sử dụng trong hệ thống:
 - **RPM** - Redhat Package Manager (Fedora/Red Hat): lệnh rpm.
 - Chứa các thông tin về những phụ thuộc mà phần mềm có đối với các phần mềm khác.
 - Ở mức cao, thông qua Yum (hoặc up2date trong một số bản phân phối có dẫn xuất từ Red Hat).
 - **DEB**: Các gói Debian thường được điều khiển với một tập hợp các công cụ làm việc trong các mức khác nhau với các gói hoặc nhóm riêng rẽ (dselect, taskset, dpkg, apt-get)
 - **tar** hoặc **tgz** (hoặc **tar.gz**): các gói đơn giản được nén bằng các lệnh như tar, và gzip

65

Cài đặt phần mềm (tt)

- Các công cụ
 - RPM: Kpackage. Cần cài các gói phụ thuộc trước
 - Yum: Các gói phụ thuộc được cài tự động
 - DEB: Synaptic, Gnomeapt
 - Tgz: Kpackage
 - Giao diện đồ họa: trong Gnome, KDE.

66

Cài đặt phần mềm (tt)

- Các cách cài đặt:
 - Từ các CD/DVD.
 - Cập nhật hoặc các dịch vụ tìm kiếm phần mềm
 - Tự do: apt-get (Debian), yum (Fedora)
 - Trả tiền: thông qua các dịch vụ như Red Hat Network của các phiên bản Red Hat.
 - Thông qua các kho phần mềm với những gói phần mềm được xây dựng sẵn trước cho một bản phân phối được xác định
 - Từ người tạo ra hoặc nhà phân phối phần mềm cung cấp các gói cài đặt phần mềm.
 - Các phần mềm không được đóng gói hoặc chỉ nén, không có bất kỳ dạng phụ thuộc nào cả.
 - Chỉ có mã nguồn, ở dạng của một gói hoặc tập tin được nén.

67

1.5. Quản trị người dùng và nhóm

- Một số khái niệm.
- Quản trị người dùng.
- Quản trị nhóm người dùng.
- Các tập tin liên quan.

68

1.5.1 Một số khái niệm

- Tài khoản (user):

- Những người sử dụng của một hệ thống GNU/Linux thường có một tài khoản có liên quan:
 - dữ liệu, độ ưu tiên
 - Không gian đĩa để chứa thư mục và tập tin: chỉ có thể được sử dụng bởi người sử dụng (trừ phi có các quyền chỉ định khác).
- Mỗi user có duy nhất một tên và id (UID).
- Mỗi user thuộc về ít nhất một nhóm (primary group).

69

Một số khái niệm

- Các loại tài khoản:

- Tài khoản của người quản trị (root):
 - chỉ sử dụng được cho các hoạt động quản trị
 - có tất cả các quyền truy cập và cấu hình.
- Tài khoản của người sử dụng:
 - Tài khoản thông thường cho bất kỳ người sử dụng máy tính
 - Các quyền được hạn chế.
- Các tài khoản dịch vụ đặc biệt:
 - Các tài khoản lp, news, wheel, www-data... được các dịch vụ nội bộ của hệ thống, sử dụng
 - Một số các dịch vụ cũng được sử dụng theo tài khoản của root.

70

Một số khái niệm

- Nhóm người dùng :
 - Mỗi nhóm có duy nhất một tên và id (GID).
 - Mỗi nhóm có thể chứa một hay nhiều thành viên.
- Lưu ý :
 - Tên tài khoản và tên nhóm người dùng là duy nhất.
 - User ID (UID) và Group ID (GID) có thể trùng nhau.

71

Một số khái niệm (tt)

- Thư mục chủ :
 - Mỗi người dùng có một thư mục chủ trùng với tên tài khoản và được đặt trong thư mục **/home/**
 - Thư mục chủ của người dùng cho phép user chứa thông tin riêng của mình trên đó.
- Thông tin môi trường làm việc người dùng - **/etc/skel/**
 - Thư mục **/etc/skel/** chứa các tập tin và thư mục cấu hình màn hình của user.
 - Nội dung có trong thư mục **/etc/skel/** cũng sẽ được chép vào thư mục chủ khi thư mục chủ được tạo.

72

Một số khái niệm

- root – tài khoản Superuser: tài khoản có quyền cao nhất trên hệ thống
 - Không bị giới hạn
 - Đảm nhiệm việc quản trị và bảo trì hệ thống
 - Sử dụng: không login trực tiếp
 - Đặt password cho root: **passwd root**
 - Chuyển sang tài khoản root: **\$su**
 - Chuyển sang tài khoản khác: **\$su username**

```
student@ubuntu:~$ sudo passwd root
[sudo] password for student:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
student@ubuntu:~$ su
Password:
root@ubuntu:/home/student#
```

73

1.5.2 Quản trị người dùng

- Tạo tài khoản mới
 - Cú pháp: useradd [options] ... username
 - Một số tùy chọn :
 - -c Mô tả thông tin tài khoản người dùng.
 - -m Tạo thư mục chủ nếu nó chưa tồn tại.
 - -u uid User ID.
 - -G group [...] Danh sách nhóm
 - -d home_dir Tạo thư mục chủ home_dir.
 - -g initial_group Tên nhóm hoặc GID.
 - Ví dụ :
 - useradd -g studs -c "Student 01" stud01
 - useradd -c "Cathy Mary" -g admin -d /home/user1 -s /bin/bash cathy

74

Quản trị người dùng

- Thay đổi mật khẩu:

- Cú pháp: **passwd [options] [username]**

- Một số tùy chọn :

- **-1** Khóa tài khoản người dùng.

- **-u [-f]** Mở khóa tài khoản người dùng. Tùy chọn **-f** cho phép mở khóa tài khoản không sử dụng mật khẩu.

- **-d** Xóa bỏ mật khẩu của tài khoản người dùng.

- Ví dụ :

```
# passwd stud01
passwd:
```

75

Quản trị người dùng

- Xóa tài khoản:

- Cú pháp: **userdel [-r] login**

- Trong đó :

- **login** Tên tài khoản người dùng muốn khóa.

- **-r** Xóa toàn bộ thông tin liên quan tới user

- Ví dụ :

```
# userdel -r sv001
```

76

Quản trị người dùng

- Thay đổi thông tin người dùng:
 - Cú pháp: `usermod [option] ... login`
 - Một số tùy chọn :
 - `-L` Khóa tài khoản
 - `-U` Mở khóa tài khoản
 - `-l login_name` Thay đổi tên tài khoản
 - `-G group[...]` Danh sách nhóm
 - `-g initial_group` Thay đổi nhóm hay mã nhóm
 - `-d home_dir` Thay đổi thư mục chủ.
 - Ví dụ :


```
#usermod -c "CNPM" -g studs sv001
```

77

1.5.3 Quản trị nhóm người dùng

- Tạo nhóm:
 - Cú pháp: `groupadd [options] group_name`
 - Một số tùy chọn :
 - `-g gid` Mã nhóm, mặc định giá trị này lớn hơn 500
 - `-r` Tạo tài khoản nhóm hệ thống, có `gid` từ 0 đến 499
 - Ví dụ :
 - `# groupadd students`
 - `# groupadd -g 10 -o sales`

78

Quản trị nhóm người dùng

- Xóa nhóm:

- Cú pháp: **groupdel group_name**
- Trong đó **group_name** là tên tài khoản nhóm.
- Ví dụ : **#groupdel sinhvien**
- **Lưu ý :**
 - Không thể xóa các nhóm còn chứa các tài khoản.
 - Phải thực hiện loại bỏ các thành viên ra khỏi nhóm sau đó mới thực hiện xóa nhóm.

79

Quản trị nhóm người dùng

- Thêm user vào nhóm:

- Cú pháp: **addgroup username groupname**
- Ví dụ: Thêm user sinh viên vào nhóm student:
 - **sudo addgroup sinhvien student**

80

Quản trị nhóm người dùng

- Thay đổi thông tin nhóm:
 - Cú pháp: **groupmod [options] group_name**
 - Một số tùy chọn :
 - **-g gid** Thay đổi mã nhóm.
 - **-n name** Thay đổi tên nhóm thành *name*.
 - Ví dụ :
 - # **groupmod -n sales marketing**

81

Quản trị nhóm người dùng

- Xem thông tin nhận diện tài khoản
 - Cú pháp: **id [option] ... [username]**
 - Một số tùy chọn :
 - **-g** Chỉ hiện thị chỉ số GID của tài khoản
 - **-u** Chỉ hiện thị chỉ số UID của tài khoản
 - **-G** Chỉ hiển thị danh sách tất cả các GID của các nhóm mà tài khoản là thành viên
 - Ví dụ :
 - # **id sv 001**
 - **uid=500(sv01) gid=500(sv01) groups=500(sv01)**

82

Quản trị nhóm người dùng

- Chuyển user sang nhóm mới
 - Cú pháp: newgrp **group_name**
 - Một số tùy chọn :
 - -g Chỉ hiện thị chỉ số GID của tài khoản
 - -u Chỉ hiện thị chỉ số UID của tài khoản
 - -G Chỉ hiển thị danh sách tất cả các GID của các nhóm mà tài khoản là thành viên
 - Ví dụ :
 - # id sv 001
 - uid=500(sv01) gid=500(sv01) groups=500(sv01)

83

Quản trị nhóm người dùng

- Các tập tin liên quan
 - **/etc/passwd**: cơ sở dữ liệu / danh sách các tài khoản người dùng trong hệ thống dưới dạng tập tin văn bản thông thường
 - **/etc/shadow**: lưu trữ toàn bộ mật khẩu của người dùng (đã được mã hóa) dùng trong hệ thống
 - **/etc/group**: Thông tin về group tương tự như file /etc/passwd
 - **/etc/gshadow**: Chứa thông tin mật khẩu của groups
 - **/etc/login.defs**: file chứa thông số mặc định khi tạo user hoặc tạo group
 - **/etc/default/useradd**: chứa các thông tin mặc định của user
 - **/etc/skel/**: Nội dung của thư mục này sẽ được copy sang HOME của user ngay khi mới vừa được tạo (mặc định là rỗng)

84

1.5.4 Các tập tin liên quan

- /etc/passwd
- /etc/shadow
- /etc/group
- /etc/login.defs
- /etc/default/useradd

85

Tập tin /etc/passwd

- Cú pháp mỗi dòng trong tập tin:
– **username:password:uid:gid:gecos:homedir:shell**
- Trong đó:
 - username Chuỗi ký tự bất kỳ, tên dùng để login.
 - password: Mật khẩu đã được mã hóa.
 - uid User ID.
 - gid Group ID.
 - gecos Thông tin thêm về user (ghi chú).
 - homedir Thư mục home của user.
 - shell Chỉ ra shell đăng nhập của người dùng.
- Lưu ý: nếu :: rỗng

86

Tập tin /etc/passwd

- Ví dụ :
 - juan:x:1000:1000:Juan Garcia, , , : /home/juan:/bin/bash
 - root:x:0:0:root:/root:/bin/bash
 - juan: username của người sử dụng hệ thống
 - x: mật khẩu được mã hóa của user; nếu có một “x” thì nó được lưu trong /etc/shadow.
 - 1000: ID của user, hệ thống sử dụng như là mã xác thực của user.
 - 1000: ID của nhóm chính của user, thông tin của nhóm này nằm trong /etc/group.
 - Juan Garcia: ghi chú, thường là tên đầy đủ của user.
 - /home/juan: thư mục cá nhân có liên quan tới tài khoản của user
 - /bin/bash: trình biên dịch lệnh tương tác mà user dùng khi tương tác với hệ thống

87

Tập tin /etc/shadow

- Cú pháp mỗi dòng trong tập tin
 - **username:passwd:d1:d2:d3:d4:d5:d6:reserved**
- Trong đó
 - username Tương ứng username trong /etc/passwd
 - passwd Mật khẩu đã được mã hóa
 - d1 Số ngày kể từ lần cuối thay đổi mật khẩu
 - d2 số ngày tối thiểu yêu cầu giữa các lần thay đổi mật khẩu
 - d3 số ngày tối đa xác định tính hợp lệ của mật khẩu sau khi thay đổi mật khẩu
 - d4 quy định số ngày trước khi mật khẩu hết hạn sẽ cảnh báo người dùng
 - d5 số ngày đã bị khóa tài khoản
 - d6 quy định ngày cụ thể hết hạn
 - reserved: chưa sử dụng
- Lưu ý : các giá trị số ngày tính theo mốc từ 1/1/1970

88

Tập tin /etc/shadow

- Tài khoản bị khóa nếu có ký tự ! đứng trước **passwd**.
- Tài khoản không có mật khẩu và không để đăng nhập hệ thống nếu có giá trị !! ở trường **passwd**.
- Tài khoản không được phép đăng nhập hệ thống nếu có giá trị * ở trường **passwd**.
- Ví dụ :
 - root:\$1\$dxtC0Unf\$2SCgulhTlrcnkSH5tjw0s/:12148:0:99999:7:::
 - daemon:*:12148:0:99999:7:::adm:*:12148:0:99999:7:::
 - nobody:*:12148:0:99999:7:::
 - xfs:!!:12148:0:99999:7:::

89

Tập tin /etc/group

- Cú pháp mỗi dòng trong tập tin:
 - **groupname:password:gid:members**
- Trong đó :
 - groupname chuỗi ký tự bất kỳ, xác định tên group
 - password mật khẩu (tùy chọn)
 - gid group id
 - members danh sách thành viên, cách nhau bằng “,” (các thành viên có groupname là secondary group)
- Ví dụ :
 - root:x:0:
 - bin:x:1:bin,daemon
 - student:x:500:

90

Tập tin /etc/login.defs

- Lưu trữ các thông tin mặc định khi tạo mới một user:
 - MAIL_DIR: Thư mục mail của user.
 - PASS_MAX_DAYS : Số ngày nhiều nhất mà mật khẩu còn hiệu lực.
 - PASS_MIN_DAYS : Số ngày ít nhất để thay đổi mật khẩu.
 - PASS_MIN_LEN : Độ dài nhỏ nhất của mật khẩu.
 - PASS_WARN_AGE : Số ngày cảnh báo trước khi mật khẩu hết hiệu lực.
 - UID_MIN : Số id user thấp nhất tự gán khi tạo user mặc định.
 - UID_MAX : Số id user cao nhất tự gán khi tạo user mặc định.
 - GID_MIN : Số id group thấp nhất tự gán khi tạo group mặc định.
 - GID_MAX : Số id group cao nhất tự gán khi tạo group mặc định.
 - CREATE_HOME : Có/ không tạo thư mục home cho user.
 - UMASK : umask cho user.
 - USERGROUPS_ENAB : Yes/no Xóa group nếu user không tồn tại
 - MD5_CRYPT_ENAB : Sử dụng mã hóa md5 hoặc DES để mã hóa mật khẩu.

91

Tập tin /etc/login.defs

- Cú pháp: **field value**
- Ví dụ :
 - MAIL_DIR /var/spool/mail
 - PASS_MAX_DAYS 99999
 - PASS_MIN_DAYS 0
 - PASS_MIN_LEN 5
 - PASS_WARN_AGE 7
 - UID_MIN 500
 - UID_MAX 60000
 - GID_MIN 500
 - GID_MAX 60000
 - CREATE_HOME yes

92

Tập tin /etc/default/useradd

- Cú pháp: **field=value**
- Ví dụ :

<code>GROUP=100</code>	Nhóm mặc định
<code>HOME=/home</code>	Thư mục chứa thư mục chủ
<code>INACTIVE=-1</code>	Số ngày tối đa được thay đổi mật khẩu sau khi mật khẩu hết hạn sử dụng.
<code>EXPIRE=</code>	Ngày hết hạn sử dụng tài khoản
<code>SHELL=/bin/bash</code>	Shell mặc định của tài khoản
<code>SKEL=/etc/skel</code>	Thư mục chứa thông tin môi trường làm việc

93

1.6. Hệ thống tập tin

- Tổ chức tập tin trên Linux
- Các lệnh thao tác trên tập tin
- Bảo mật hệ thống tập tin

94

1.6.1 Tổ chức tập tin trên Linux

- Tập tin trong Unix được để trên thiết bị khồi (đĩa, băng từ).
- Máy tính có thể có nhiều đĩa, mỗi đĩa có thể có một hay vài phân hoạch, mỗi phân hoạch tạo thành một hệ thống tập tin (**File system - FS**).
- Phân hoạch một đĩa thành nhiều phần tạo điều kiện dễ dàng cho việc kiểm soát dữ liệu, vì kernel xử lí ở mức logic với các FS chứ không phải với bản thân thiết bị.
- Mỗi một phân hoạch là một thiết bị logic

95

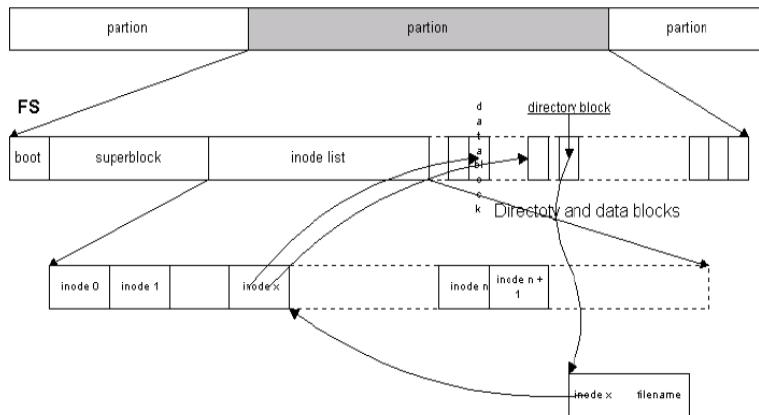
Tổ chức tập tin trên Linux (tt)

- Trong Linux file được xem như là một inode, thư mục là một file chứa những entry.
- Các thành phần của hệ thống tập tin :
 - Boot block
 - Superblock
 - Inode
 - Storageblock

96

Tổ chức tập tin trên Linux (tt)

- Tổ chức hệ thống tập tin trên đĩa



97

Tổ chức tập tin trên Linux (tt)

- **Boot block:** phần đầu tiên của FS đĩa, là sector đầu tiên chứa mã bootstrap được đọc vào máy và chạy để nạp HĐH
- **Super block:** là cấu trúc được tạo tại vị trí bắt đầu file system, mô tả tình trạng của FS:
 - Độ lớn (block size), chứa được bao nhiêu tập tin (inode)
 - Không gian còn trống (free block) của đĩa để chứa nội dung tập tin tìm thấy ở đâu.
 - Thời gian gắn kết (mount) cuối cùng của tập tin.
 - Thông tin trạng thái tập tin

98

Tổ chức tập tin trên Linux (tt)

- Super block có các trường sau đây:
 - kích thước của FS
 - tổng số các block còn chưa cấp phát cho tập tin (free block) trong FS.
 - danh sách các free block sẵn có trên FS (xem thêm phần cấp phát block đĩa),
 - chỉ số của free block tiếp theo trong danh sách free block.
 - Kích thước của danh sách inode
 - tổng số các inode chưa cấp phát (free inode) trong FS
 - danh sách free inode trong FS
 - chỉ số của free inode tiếp theo trong danh sách free inode
 - các trường khoá (lock) cho free block và các danh sách free inode
 - cờ (flag) cho biết super block đã có thay đổi.

99

Tổ chức tập tin trên Linux (tt)

- **Inode:**
 - Lưu những thông tin về tập tin và thư mục được tạo trong hệ thống tập tin. Nhưng không lưu tên tập tin và thư mục.
 - Mỗi tập tin tạo ra sẽ được phân bổ một inode lưu thông tin sau:
 - Loại tập tin và quyền hạn truy cập.
 - Người sở hữu tập tin.
 - Kích thước và số hard link đến tập tin.
 - Ngày và giờ chỉnh sửa tập tin lần cuối cùng.
 - Vị trí lưu nội dung tập tin trong filesystem.

100

Tổ chức tập tin trên Linux (tt)

- **Storage block:**

- Là vùng lưu trữ dữ liệu thực sự của tập tin và thư mục.
- Chia thành những datablock, trong đó mỗi block chứa 1024 byte.
 - Datablock của tập tin thường lưu inode của tập tin và nội dung của tập tin.
 - Datablock của thư mục lưu danh sách những entry gồm inode number, tên tập tin và những thư mục con.

101

Tổ chức tập tin trên Linux (tt)

- Một số hệ thống tập tin

- VFS
- Ext2
- Ext3
- Jfs
- Vfat
- Iso9660
- Swap

102

Tổ chức tập tin trên Linux (tt)

- Các kiểu tập tin trong Linux:

Kiểu tập tin	Ký hiệu
Regular	- hoặc f
Directory	d
Charater device	c
Block device	b
Domain socket	s
Name pipes	p
Hard link	
Symbolic link	l

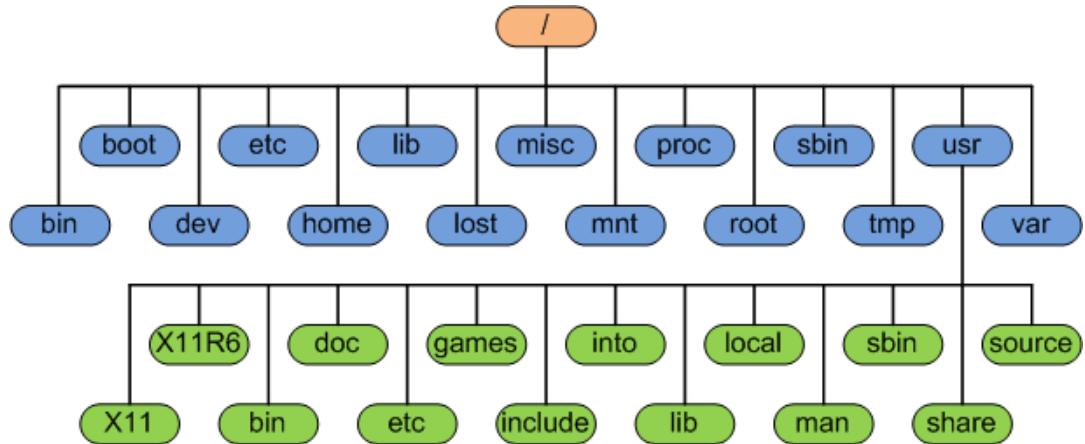
103

Quy ước đặt tên file

- Tối đa 255 ký tự, có phân biệt chữ thường, chữ hoa
- Có thể sử dụng bất kỳ ký tự nào (kể cả ký tự đặc biệt)
 - Ví dụ: "very ? long - file + name.test"
- File / thư mục ẩn được bắt đầu bằng một dấu “.”
 - Ví dụ:
 - .bash_history .bash_profile .bashrc
 - .desktop/ .kde/ .mozilla/

104

Tổ chức cây thư mục



105

Các thư mục cơ bản

Thư mục	Ý nghĩa
/bin, /sbin	Chứa tập tin nhị phân hỗ trợ cho việc boot và thực thi các lệnh.
/boot	Chứa Linux Kernel, file ảnh hỗ trợ load hệ điều hành.
/lib	Chứa các thư viện cần thiết để thi hành các tập tin nhị phân trong thư mục /bin, /sbin
/usr/local	Chứa các thư viện, phần mềm để chia sẻ cho các máy khác trong mạng.
/tmp	Chứa các file tạm.
/dev	Chứa các tập tin đại diện cho các thiết bị (CD-ROM, Floppy) được gắn với hệ thống.

106

Các thư mục cơ bản (tt)

Thư mục	Ý nghĩa
/etc	Chứa các tập tin cấu hình của các dịch vụ trên máy tính.
/home	Chứa các thư mục home directory của người dùng.
/root	Lưu trữ home directory cho user root.
/usr	Chứa các tập tin có thể dùng chung trên toàn hệ thống, đây cũng là nơi lưu trữ tập tin các chương trình đã được cài đặt.
/var	Lưu trữ các log file, các file quản trị, và các file tạm của hệ thống.
/mnt	Chứa các tham chiếu đến các hệ thống tập tin được gắn kết (mount) vào hệ thống.
/proc	Chứa những tập tin đại diện cho trạng thái hiện tại của kernel.

107

Đường dẫn

- Đường dẫn tuyệt đối: bắt đầu bằng “/”
 - Ví dụ: / /bin /usr /usr/bin
- Đường dẫn tương đối: không bắt đầu bằng “/”
 - Ví dụ: bin usr/local/bin ..sbin ./
- Đường dẫn đặc biệt
 - .. thư mục cha
 - . thư mục làm việc hiện tại

108

1.6.2 Các lệnh thao tác trên tập tin

- Lệnh gắn kết hệ thống tập tin
- Lệnh tạo tập tin liên kết
- Các lệnh xem nội dung.
- Nhóm lệnh sao chép/xóa/di chuyển tập tin.
- Nhóm lệnh tìm kiếm và so sánh.
- Lưu trữ tập tin, thư mục.
- Bảo mật hệ thống tập tin.

109

Gắn kết hệ thống tập tin

- Lệnh **mount** để gắn kết hệ thống tập tin vào hệ thống.
 - Cú pháp : **mount [-t type] <device> <directory>**
 - Trong đó :
 - -t type : chỉ rõ kiểu hệ thống tập tin type của thiết bị.
 - device : là thiết bị vật lý như CD-ROM, đĩa mềm, usb,...
 - directory : là thư mục muốn mount vào.
- Lệnh **umount** để gỡ bỏ gắn kết hệ thống tập tin đã được mount ra khỏi hệ thống.
 - Cú pháp: **umount <device hoặc directory>**

110

Gắn kết hệ thống tập tin

- Ví dụ:

 - mount usb

- Cắm usb vào, gõ lệnh **fdisk -l** để xem các partition
 - Tạo thư mục để mount usb vào, ví dụ **mkdir data**
 - Thực hiện mount:
- **mount /dev/sdb1 data**

 - Umount:

 - Umount data

```

Device      Boot Start      End  Sectors  Size Id Type
/dev/sdb1          256 15734783 15734528  7.5G b W95 FAT32

student@ubuntu:~$ sudo mount data /dev/sdb1
mount: /home/student/data is not a block device
student@ubuntu:~$ sudo mount /dev/sdb1 data
  
```

111

Tạo tập tin liên kết

- Link là tạo ra một tập tin thứ hai cho một tập tin đã tồn tại.
- Có 2 loại tập tin liên kết :

 - **Hard link** : là một tập tin liên kết tới một tập tin khác.

 - Nội dung của hard link và tập tin nó liên kết tới luôn giống nhau.
 - Khi thay đổi nội dung của hard link thì nội dung của tập tin mà nó liên kết tới cũng thay đổi, và ngược lại.
 - Không thể tạo hard-link trên hai partition khác nhau
 - Cú pháp: **ln [option] filename linkname**
 - option: -f: ghi đè
 - Ví dụ: **cat > abc.txt Hello Ctrl-D**
ln abc.txt hard_abc.txt

112

Tạo tập tin liên kết (tt)

– **Symbolic link** : là một tập tin chỉ chứa tên của tập tin khác. Khi nhân của hệ điều hành duyệt qua symbolic link thì nó sẽ được dẫn tới tập tin mà symbolic link chỉ đến.

- Symbolic link có thể liên kết đến một thư mục và liên kết đến tập tin trên partition khác
- Cú pháp: **In –s oldfile hardfile**

113

Nhóm lệnh xem nội dung

- **pwd**: Hiển thị đường dẫn đầy đủ tới thư mục hiện hành.
- **cd**: Thay đổi thư mục hiện hành
 - \$ cd /usr ([/usr])
 - \$ cd bin ([/usr/bin])
 - \$ cd ../../etc ([/etc])
 - \$ cd ~ ([/home/student])
 - \$ cd ([/home/student])

114

Nhóm lệnh xem nội dung

- **ls:** Liệt kê nội dung thư mục

– Cú pháp: **ls [option] [directory]**

– option:

- ls -x: hiển thị trên nhiều cột.
- ls -l: hiển thị chi tiết các thông tin của tập tin.
- ls -a: hiển thị tất cả các tập tin kể cả tập tin ẩn.

115

Nhóm lệnh xem nội dung

- **ls:**

Phân loại file các quyền truy cập cho chủ						
Các quyền truy cập cho nhóm chủ						
Các quyền truy cập cho người dùng bên ngoài						
1	/home/user # ls -l	Kích thước của file	Thời điểm file được tạo ra			
2	total 36			.		
3	drwxr-xr-x	3 root root 4096 2006-06-29 04:21 .				
4	drwxr-xr-x	7 root root 4096 2006-06-23 02:13 ..				
5	-rwxr-xr-x	1 root root 6096 2006-06-22 09:26 functions				
6	-rw-r--r-	2 anon users 651 2006-06-23 05:23 hardlink				
7	-rw-r--r--	2 anon users 651 2006-06-23 05:23 mark.txt				
8	drwxr-xr-x	2 root root 4096 2006-06-22 09:27 mydir				
9	brw-rw----	1 root disk 8, 192 2005-05-24 08:09 sdm				
10	-rwsr-sr-x	1 root root 6096 2006-06-22 09:29 share				
11	lrwxrwxrwx	1 root root 9 2006-06-22 09:28 softlink -> functions				
12	-rw-r--r--	1 root root 0 2006-06-29 04:21 zerobyte.txt				
Tổng số liên kết đến cùng 1 file						
Tên nhóm chủ của tập tin						
Tên tập tin						
Tài khoản cá nhân chủ tập tin						

116

Nhóm lệnh xem nội dung

- **wc**: cho biết thông tin về số dòng, số từ, kích thước (byte) của tập tin.
 - Cú pháp: **wc [option] [file1] ... [file n]**
 - option:
 - -c kích thước tập tin (byte) gồm cả ký tự CR và EOF
 - -m số lượng ký tự có trong tập tin
 - -w số lượng từ có trong tập tin
 - -l số dòng trong tập tin
 - -L chiều dài của dòng dài nhất

117

Nhóm lệnh xem nội dung

- **touch**: tạo tập tin rỗng
- **cat**: tạo tập tin, có thể cho phép xem và ghi nội dung tập tin
 - Cú pháp: **cat [option] [file1] ... [file n]**
 - **cat**: xem nội dung tập tin
 - **cat >**: tạo tập tin (ghi đè tập tin đã tồn tại)
 - **cat >>**:
 - tạo tập tin (nếu chưa tồn tại)
 - Ghi nối vào tập tin (nếu đã tồn tại)
 - Một số tùy chọn :
 - -s xóa các dòng trống chỉ để lại 1 dòng duy nhất.
 - -n đánh số thứ tự các dòng, kể cả dòng trống.
 - -b đánh số thứ tự các dòng, ngoại trừ dòng trống.

118

Nhóm lệnh xem nội dung

- **more**: Xem nội dung của tập tin theo từng trang màn hình.
 - Cú pháp: **more [option] [file 1] ... [file n]**
 - option:
 - -num xác định kích thước của màn hình num dòng.
 - +num dòng bắt đầu hiển thị.
 - -s xóa bớt các dòng trắng, chỉ để lại một dòng trắng giữa mỗi đoạn

119

Nhóm lệnh xem nội dung

- **head**: Xem nội dung đầu tập tin.
 - Cú pháp: **head [option] [file 1] ... [file n]**
 - option:
 - -n in ra màn hình n dòng đầu tiên (mặc định lệnh head sẽ hiển thị 10 dòng đầu).
 - -q không hiển thị ra màn hình phần đầu đề chứa tên tập tin trong trường hợp mở nhiều tập tin cùng lúc.

120

Nhóm lệnh xem nội dung

- **tail:** xem nội dung cuối tập tin.
 - Cú pháp: tail [option] [file 1] ... [file n]
 - option:
 - -n in ra màn hình n dòng cuối cùng (mặc định lệnh tail sẽ hiển thị 10 dòng cuối).
 - -q không hiển thị ra màn hình phần đầu đè chứa tên tập tin trong trường hợp mở nhiều tập tin cùng lúc.

121

Nhóm lệnh sao chép di chuyển

- **cp:** sao chép tập tin / thư mục.
 - Cú pháp: cp [option] source dest
 - option:
 - -f ghi đè không cần hỏi (force)
 - -i hỏi trước khi ghi đè (interactive)
 - -R,-r copy toàn bộ thư mục kể cả con
 - Ví dụ :
 - \$cp -r dir1 dir5
 - \$cp file1 file5

122

Nhóm lệnh sao chép di chuyển

- **rm**: Xóa tập tin và thư mục
 - Cú pháp: **rm [option] file**
 - option:
 - -f xoá không cần hỏi
 - -i hỏi trước khi xoá
 - -R,-r xoá toàn bộ thư mục kể cả con
 - Lưu ý : không dùng lệnh: #rm -rf

123

Nhóm lệnh sao chép di chuyển

- **mv**: đổi tên/di chuyển tập tin.
 - Cú pháp: **mv [option] source dest**
 - option:
 - -f ghi đè không cần hỏi (force)
 - -i hỏi trước khi ghi đè (interactive)
 - Ví dụ :
 - \$mv file5 file6
 - \$mv file1 dir5

124

Nhóm lệnh sao chép di chuyển

- **mkdir:** tạo thư mục.
 - Cú pháp: **mkdir [option] directory**
 - option:
 - -p tạo thư mục cha nếu chưa tồn tại
 - Ví dụ :
 - \$mkdir dir1
 - \$mkdir dir1 dir2
 - \$mkdir -p dir3/dir4

125

Nhóm lệnh sao chép di chuyển

- **rmdir:** xóa thư mục rỗng.
 - Cú pháp: **rmdir [option] directory**
 - option:
 - -p xoá tất cả các thư mục tạo nên đường dẫn
 - Ví dụ :
 - \$rmdir dir1
 - \$rmdir dir1 dir2
 - \$rmdir -p dir3/dir4
 - <=> \$rmdir dir3/dir4 dir3
- **rm -r:** xóa thư mục không rỗng

126

Nhóm lệnh tìm kiếm và so sánh

- **find:** tìm kiếm tập tin.

– Cú pháp: **find [path ...] [expression]**

– Một số tùy chọn :

- -name pattern tìm các tập tin có tên chứa chuỗi pattern
- -group name tìm các tập tin thuộc nhóm name
- -user name tìm các tập tin tạo bởi user có tên name
- -size [+/-]n[bck] tìm các tập tin kích thước lớn hơn/nhỏ hơn n block (512 bytes/block). Kích thước là block nếu ký tự sau là b, c là byte, k là kilobytes.
- -type filetype tìm các tập tin có kiểu là filetype

127

Nhóm lệnh tìm kiếm và so sánh

- **grep:** tìm kiếm các dòng có chứa một chuỗi hoặc từ khóa trong tập tin.

– Cú pháp: **grep [option] pattern [file] ...**

– option:

- -i không phân biệt hoa thường
- -n kèm theo số thứ tự dòng khi xuất kết quả
- -r tìm lặp lại trong thư mục con
- -v tìm nghịch đảo
- -a xử lý tập tin nhị phân như là một tập tin văn bản.
- -w tìm chính xác
- -c đếm số dòng trong kết quả tìm thấy

128

Nhóm lệnh tìm kiếm và so sánh

- **grep:** (tt)
 - Một số regular expression :
 - ^ begin of line
 - . ký tự bất kỳ
 - \$ end of line
 - Ví dụ :
 - Liệt kê tất cả các file trong /etc bắt đầu bằng b, k, n: `ls /etc | grep "^[bkn]"`
 - Liệt kê tất cả các file trong /etc có ký tự kế cuối là a: `ls /etc | grep "a.$"`
 - Tìm user tên thanh trong file /etc/passwd: `$ grep thanh /etc/passwd`
 - Tìm hai từ: `$ egrep -w 'word1|word2' /path/to/file`
 - Đếm số dòng tìm thấy từ word: `$ grep -c 'word' /path/to/file`

129

Nhóm lệnh tìm kiếm và so sánh

- **cmp:** so sánh hai tập tin có kiểu bất kỳ.
 - Cú pháp: `cmp [-l] file1 file2`
 - Trong đó `-l` cho phép xuất ra danh sách các vị trí khác nhau giữa hai tập tin.
 - Ví dụ :
 - `$cmp myfile m1`
 - `myfile m1 differ: byte 2, line 5`

130

Nhóm lệnh tìm kiếm và so sánh

- **diff**: tìm sự khác nhau giữa hai tập tin.
 - Cú pháp: **diff [option] file1 file2**
 - option:
 - **-i** so sánh không phân biệt hoa thường
 - **-s** hiển thị thông báo nếu hai tập tin giống nhau
 - **-w** bỏ qua khoảng trắng giữa các từ
 - **-r** so sánh tất cả các tập tin trong các thư mục con

131

Nhóm lệnh nén - lưu trữ

- Nén / giải nén:

Nén	Giải nén	Cú pháp	Mở rộng
bzip2	bunzip2	bzip2 [options] filebz2
gzip	gunzip	gzip [options] filegz
zip	unzip	zip [options] zipfile filezip

- Ví dụ:
 - **gzip /etc/passwd**
 - **gzip /etc/passwd.gz**
 - **zip -u myzip myfile**

132

Nhóm lệnh nén - lưu trữ (tt)

- Lưu trữ: sao lưu hoặc kết hợp nhiều tập tin thành một tập tin.
 - Cú pháp: **tar [OPTIONS] [DIRECTORY/FILE]**
 - Một số tùy chọn:
 - **-c** tạo một tập tin lưu trữ mới
 - **-x** lấy các tập tin ra từ một tập tin lưu trữ
 - **-z** nén/giải nén các tập tin lưu trữ bằng gzip
 - **-j** nén/giải nén các tập tin lưu trữ bằng bzip2
 - **-f** lưu trữ tới tập tin hay thiết bị, phải **luôn có** tùy chọn này.
 - **-v** hiển thị danh sách các tập tin trong quá trình bung.
 - **-vv** cung cấp thêm nhiều thông tin hơn so với **-v**

133

Nhóm lệnh nén - lưu trữ (tt)

- Lưu trữ (tt) - ví dụ :
 - \$tar -cvf bak.tar dir1/
 - \$tar -xvf bak.tgr
 - \$tar -zcvf dir1.tar.gz dir1/
 - \$tar -zcvf alldir.tgz dir1 dir2 dir3
 - \$tar -zxvf source.tar.gz
 - \$tar -jxvf kernel.tar.bz2
- Lưu ý :
 - Sử dụng nhóm tùy chọn cvf để gom tập tin / thư mục.
 - Sử dụng nhóm tùy chọn xvf để bung tập tin / thư mục.

134

1.6.3 Bảo mật hệ thống tập tin

- Quyền sở hữu và quyền truy cập

- Tất cả file và thư mục thuộc sở hữu user tạo ra chúng
- Quyền truy cập file được chia làm 3 nhóm
 - User chủ sở hữu file (owner)
 - Group nhóm có user là thành viên
 - Others các user khác còn lại trên hệ thống
- Xem quyền truy cập với lệnh ls -l

135

Bảo mật hệ thống tập tin (tt)

- Biểu diễn quyền truy cập

user	group	others
r w x	r w x	r w x

Dạng ký hiệu	Dạng số	Ý nghĩa
r	4	Cho phép đọc
w	2	Cho phép ghi
x	1	Cho phép thực thi
-	0	Loại bỏ quyền

- Xác định quyền truy cập bằng cách tính tổng các giá trị.
- Ví dụ :
 - Quyền đọc và thực thi là : $4 + 1 = 5$
 - Quyền đọc, ghi và thực thi là : $4 + 2 + 1 = 7$

136

Bảo mật hệ thống tập tin (tt)

- Định danh quyền truy cập
 - u user, chủ sở hữu file
 - g group, nhóm có user là thành viên
 - o others, các user khác trên hệ thống
 - a all, tất cả user (u, g và o)
- Tác vụ trên quyền truy cập
 - + thêm quyền
 - - loại bỏ quyền
 - = gán quyền

137

Bảo mật hệ thống tập tin (tt)

- Lệnh **chmod**: cấp quyền sử dụng tập tin/thư mục.
 - Cú pháp : **chmod [option] mode file**
 - option: -R : thay đổi cả trong thư mục con
 - Ví dụ
 - g+w thêm quyền ghi cho group
 - o-rwx loại bỏ tất cả các quyền của others
 - +x thêm quyền thực thi cho tất cả
 - a+rwx thêm quyền ghi cho tất cả
 - ug+r thêm quyền đọc cho user và group
 - o=x chỉ cho phép thực thi với others

138

Bảo mật hệ thống tập tin (tt)

- Lệnh chmod (tt): ví dụ
 - \$ chmod -x *.php
 - \$ chmod -R ug+rw lecture
 - \$ chmod u=rwx,ug=r desktop.jpg
 - \$ chmod 644 homelist.txt
 - \$ chmod 755 myprogram
 - \$ chmod 777 /tmp/tmp

139

Bảo mật hệ thống tập tin (tt)

- Lệnh **chown**: thay đổi người sở hữu tập tin
 - Cú pháp: **chown [option] username file ...**
 - -R : thay đổi cả trong thư mục con
- Lệnh **chgrp** cho phép thay đổi nhóm sở hữu tập tin.
 - Cú pháp: **chgrp [option] group file**

140

Bảo mật hệ thống tập tin (tt)

- Lệnh **umask** (*user file-creation mode mask*): thiết lập quyền truy cập mặc định khi tạo tập tin, thư mục
 - Quyền truy cập file và thư mục phụ thuộc vào giá trị umask
 - Quyền truy cập file = 666 AND (NOT **umask**)
 - Quyền truy cập thư mục = 777 AND (NOT **umask**)

141

Bảo mật hệ thống tập tin

- Lệnh **umask** (tt)
 - Ví dụ: umask = 022
 - quyền truy cập mặc định khi tạo file sẽ là:
 $666 \text{ AND } (\text{NOT } 022) = 666 \text{ AND } 755 = 644$
 - quyền truy cập mặc định khi tạo thư mục sẽ là:
 $777 \text{ AND } (\text{NOT } 022) = 777 \text{ AND } 755 = 755$
 - Các phép toán AND, NOT được thực hiện trên các số nhị phân, ví dụ:
 - $\text{NOT } (022) = \text{NOT } (000\ 010\ 010) = 111\ 101\ 101 = 755$
 - $666 \text{ AND } 755 = (110\ 110\ 110) \text{ AND } (111\ 101\ 101) = 110\ 100\ 100 = 644$

142

Bảo mật hệ thống tập tin

- **Lệnh umask (tt)**

- **Tập tin**

0 = 1 = rw

2 = 3 = r

4 = 5 = w

6 = 7 = -

VD: umask 123 = 644 = rw-r--r--

- **Thư mục**

0 = chmod 7 = rwx

1 = chmod 6 = rw-

2 = chmod 5 = r-x

3 = chmod 4 = r--

4 = chmod 3 = -wx

5 = chmod 2 = -w-

6 = chmod 1 = --x

7 = chmod 0 = ---

VD: umask 123 = 654 = rw-r-xr--

143

Chương 2

LẬP TRÌNH SHELL

144

NỘI DUNG

1. Giới thiệu
2. Soạn thảo, cấp quyền, thực thi
3. Cú pháp ngôn ngữ Shell
4. Làm việc với chuỗi và văn bản
5. Mảng
6. Hàm
7. Các lệnh nội tại của Shell

145

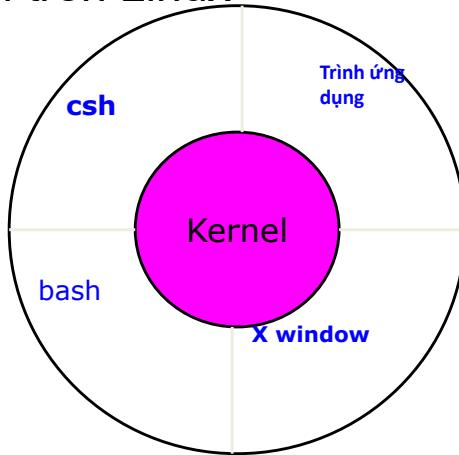
2.1. Giới thiệu

- HĐH cung cấp khả năng giao tiếp với kernel thông qua trình diễn dịch trung gian → Shell
- Chức năng giống “command.com” (DOS)
- Các lệnh trong và lệnh ngoài
- Kết hợp nhiều tiến trình :
- \$ls -al | more

146

Giới thiệu (tt)

- Các loại Shell trên Linux



147

Giới thiệu (tt)

- Các loại Shell trên Linux
 - Bourne shell: còn gọi là sh, do Steve Bourne tạo ra
 - C-shell: còn gọi là csh, đi kèm với BSD UNIX
 - Korn shell: còn gọi là ksh, do David Korn tạo ra
 - Bash (Bourne Again Shell): là shell mặc định của Linux
 - Người dùng có thể chuyển đổi giữa các shell
 - Ví dụ: chuyển từ bash sang csh, gõ lệnh:
 - \$ csh

148

Giới thiệu (tt)

- Sử dụng Shell như ngôn ngữ lập trình: có hai cách để viết chương trình điều khiển shell
 - Cách 1: Gõ chương trình trực tiếp ngay dòng lệnh

• Ví dụ:

```
$for file in *
>do
>if grep -l 'main()' $file
>then
>more $file
>fi
>done
```

• Có thể viết liên tục các lệnh phân cách bởi dấu “;”

```
$for file in *;do;if grep -l 'main()' $file; then;
more $file;fi;done
```

149

Giới thiệu (tt)

- Cách 2: viết các câu lệnh vào một tập tin và yêu cầu shell thực thi tập tin này như là một file chương trình (cần cấp quyền thực thi cho tập tin này mới có thể thực thi được)

```
# script tìm trong thư mục hiện hành #
các chuỗi chứa chuỗi "main()"
for file in *
do
if grep -l 'main()' $file
then
more $file
fi
Done
exit 0
```

150

2.2. Soạn thảo, cấp quyền, thực thi

- Tạo tập tin shell script bằng một trình soạn thảo văn bản (thông thường, tập tin shell nên có phần mở rộng là .sh)
- Cấp quyền thực thi cho tập tin
- Thực thi tập tin

Cấp quyền và thực thi chương trình hello :

Ví dụ

```
$cd /home/hv/baitap
$chmod +x hello
$./ hello
```

151

2.3. Cú pháp ngôn ngữ Shell

- Biến
- Chuyển hướng vào/ra
- Lệnh kiểm tra
- Biểu thức tính toán expr
- Cấu trúc điều khiển
- Danh sách
- Lấy về kết quả một lệnh

152

Sử dụng biến

- Biến không cần phải khai báo trước khi sử dụng
- Có phân biệt chữ hoa, chữ thường
- Biến trong shell luôn có dạng chuỗi
- Gán trị cho biến:
 - Tenbien=giatri (không có khoảng trắng hai bên dấu =)
 - name="hung"
 - number=100

153

Sử dụng biến

- Lấy giá trị của biến: sử dụng \$ trước tên biến
 - Ví dụ:
- tp='HaNoi'
- echo \$tp
- HaNoi
- Hiển thị giá trị của biến ra màn hình: sử dụng lệnh echo
 - echo 'noi dung'
 - echo "noi dung"

154

Sử dụng biến

- Phân biệt giữa cặp nháy đơn và nháy đôi

- Nội dung nằm giữa cặp nháy đơn được xem là hằng chuỗi echo ‘ Gia tri cua bien la \$bien ’

– Ví dụ:

Kết quả hiện thị : Gia tri cua bien la \$bien

- Nội dung nằm giữa cặp nháy đôi có thể là hằng chuỗi và cả giá trị biến. Mu

\

```
echo ten=Dung
echo "Su dung dau nhay kep"
echo "Gia tri bien la $ten"
echo "Ky hieu tien la \$"
```

c hiết nhoi thêm vào trước đó ký tự
Su dung dau nhay kep
Gia tri bien la Dung
Kết quả
hiển thị
Ky hieu tien la \$

155

Sử dụng biến

- Nhập giá trị cho biến từ bàn phím

– Cú pháp: **read tenbien**

Ví dụ

```
echo "Nhap vao ten cua ban "
read ten
echo "Ten vua nhap la $ten"
```

- Biến môi trường:

- Là biến được định nghĩa trước và mang giá trị mặc định khi shell khởi động
- Tên biến môi trường thường là chữ hoa

156

Sử dụng biến

- Biến môi trường:

HOME	Chứa thư mục home của người dùng
PATH	Danh sách các thư mục tìm kiếm khi thực hiện các lệnh
PS1	Dấu nhắc lệnh (root: #, người dùng bình thường: \$)
PS2	Dấu nhắc thứ cấp, thường là >
IFS	Dấu phân cách các trường trong danh sách chuỗi, thường là dấu khoảng trắng, tab và xuống hàng
PPID	Số ID của tiến trình cha trong SHELL
RANDOM	Số ngẫu nhiên
SECONDS	Thời gian làm việc tính theo giây

157

Sử dụng biến

- Biến tham số:

- Dùng để tiếp nhận tham số trên dòng lệnh cho việc xử lý
- Sau các lệnh được gọi thường có các tham số, đó là giá trị các biến tham số của chương trình

\$1, \$2, \$3	Giá trị các biến tham số thứ nhất, thứ 2.. tương ứng với các tham số từ trái sang phải trong dòng tham số.
\$0	Tên tập tin lệnh gọi
\$*	Danh sách tham số đầy đủ
\$#	Tổng số tham số.
\$\$	Số tiến trình mà chương trình đang hoạt động

158

Chuyển hướng vào ra

- Tiến trình linux luôn gắn liền với các đầu xử lý dòng (stream) dữ liệu như:
 - Đầu vào chuẩn: stdin (0)
 - Đầu ra chuẩn: stdout (1)
 - Cơ sở dữ liệu lỗi: stderr (2)
- Các hướng vào ra có thể thay đổi bởi các thông báo đặc biệt như:

>	Đầu ra hướng tới ...
>>	Nối vào nội dung của ...
<	Lấy đầu vào từ < ...
<< word	đầu vào là ở đây ...
2>	đầu ra báo lỗi sẽ hướng vào ...
2>>	đầu ra báo lỗi hướng và ghi thêm vào ...

159

Chuyển hướng vào ra (tt)

- Ví dụ:
 - date > file1.txt: lấy kết quả lệnh date ghi vào file1.txt
 - ls
 - cat < file1 > file2: lệnh cat nhận nội dung file1 ghi vào file2

160

Lệnh kiểm tra

- Sử dụng lệnh test hoặc [], cho phép kiểm tra 3 kiểu sau:
 - Kiểm tra chuỗi:

Phép so sánh	Kết quả
Chuoi1 = chuoi2	Đúng (true) nếu 2 chuỗi bằng nhau
Chuoi1 != chuoi2	Đúng nếu 2 chuỗi khác nhau
-n chuoi	Đúng nếu chuỗi “chuoi” không rỗng
-z chuoi	Đúng nếu chuỗi “chuoi” rỗng

161

Lệnh kiểm tra

- So sánh toán học:

Phép so sánh	Kết quả
bieuthuc1 -eq biethuc2	Đúng nếu bieuthuc1 bằng biethuc2
bieuthuc1 -ne biethuc2	bieuthuc1 không bằng biethuc2
bieuthuc1 -gt biethuc2	bieuthuc1 lớn hơn biethuc2
bieuthuc1 -ge biethuc2	bieuthuc1 lớn hơn hoặc bằng biethuc2
bieuthuc1 -lt biethuc2	bieuthuc1 nhỏ hơn biethuc2
bieuthuc1 -le biethuc2	bieuthuc1 nhỏ hơn hoặc bằng biethuc2

162

Lệnh kiểm tra

– Kiểm tra
tập tin:

Phép kiểm tra	Kết quả
-d file	Đúng nếu tập tin là thư mục
-e file	tồn tại trên đĩa
-f file	là tập tin thông thường
-g file	có xác lập set-group-id trên file
-s file	có kích thước >0
-u file	có xác lập set-user-id
-r file	cho phép đọc
-w file	có phép ghi
-x file	cho phép thực thi

163

Lệnh kiểm tra

– Ví dụ:

```
if test -f hello.c
then
    ...
fi
```

```
if [-f hello.c ]
then
    ...
fi
```

164

Lấy kết quả của một lệnh

- Cú pháp \$(command)
 - Ví dụ:

```
echo Current directory is $PWD
echo It contents $(ls -a) files
exit 0
```

- Lệnh let: sử dụng không cần tiền tố \$:
 - no1=4;
 - no2=5
 - let result=no1+no2
 - echo \$result
 - let no1++
 - let no1--
 - let no1+=6 (tương đương với let no1=no1+6)
 - let no2-=6 (tương đương với let no2=no2-6)

165

Lấy trị biểu thức với [], ()

- Cú pháp \${[bieuthuc]}, \${((bieuthuc))}

- Ví dụ:

- result=\${[no1 + no2]}
 - result=\${((no1 + 50))}

- Có thể sử dụng tiền tố \$ bên trong toán tử []:

- result=\${[\$no1 + 5]}

166

Biểu thức tính toán expr

- Dùng để tính trị các biểu thức số nguyên
- expr được đặt giữa hai dấu `
- Trong biểu thức, các toán tử và toán hạng phải cách nhau bằng khoảng trắng
- Cho phép sử dụng các toán tử sau:
 - +, -, *, /, %
 - |, &
 - >, >=, <, <=, !=
- Ví dụ:
 - result=`expr 3 + 4`
 - result=\$(expr \$no1 + 5)

167

Sử dụng bc

- Được dùng trong tính trị các biểu thức, kể cả số thực
- Hỗ trợ định dạng xuất
- Ví dụ:
 - echo "4 * 0.56" | bc → 2.24
 - no=54
 result=`echo "\$no * 1.5" | bc`
 echo \$result
 →81.0

168

Sử dụng bc (tt)

- Định dạng số thập phân với bc:
 - echo "scale=2;3/8" | bc → 0.37
- Chuyển đổi cơ số với bc
 - no=100
echo "obase=2;\$no" | bc
→ 1100100
 - no=1100100
echo "obase=10;ibase=2;\$no" | bc
→ 100

169

Kết nối lệnh, khối lệnh

- Kết hợp **và** (and)
 - Cú pháp: **lenh1 && lenh2 && lenh3**
 - Trả về true nếu cả ba lệnh 1,2,3 đều đúng
 - Trả về false nếu một trong các lệnh sai
- Kết hợp **hoặc** (or)
 - Cú pháp: **lenh1 || lenh2 || lenh3**
 - Trả về true nếu một trong các lệnh đúng
 - Trả về false nếu cả ba lệnh 1,2,3 đều sai
- Khối lệnh: các lệnh nằm giữa cặp **{ }**

170

Cấu trúc điều khiển

- Cấu trúc lựa chọn
 - if
 - if..elif
 - case
- Cấu trúc lặp
 - for
 - while
 - until
- Lệnh break, continue, exit

171

Lệnh if

- Cú pháp:

```
if condition
then
statements
else
statements
fi
```

Ví dụ:

```
echo "Is it morning? Please answer yes or no"
read timeofday
if [ $timeofday = "yes" ]; then
    echo "Good morning"
else
    echo "Good afternoon"
fi
```

172

Lệnh if..elif

- if..elif:

```

echo "Is it morning? Please answer yes or no"
read timeofday
if [ $timeofday = "yes" ]
then
echo "Good morning"
elif [ $timeofday = "no" ]; then
echo "Good afternoon"
else
echo "Sorry, $timeofday not recognized. Enter yes or no"
exit 1
fi
exit 0

```

173

Lệnh case

- Cú pháp:

```

case variable in
pattern [ | pattern] ...) statements;;
pattern [ | pattern] ...) statements;;
...
esac

```

- Ví dụ:

```

echo "Is it morning? Please answer yes or no"
read timeofday
case "$timeofday" in
yes) echo "Good Morning";;
no ) echo "Good Afternoon";;
y ) echo "Good Morning";;
n ) echo "Good Afternoon";;
* ) echo "Sorry, answer not recognized";;
esac

```

174

Lệnh case (tt)

- Có thể gộp các điều kiện:

```
echo "Is it morning? Please answer yes or no"
read timeofday
case "$timeofday" in
yes | y | Yes | YES ) echo "Good Morning";;
n* | N* ) echo "Good Afternoon";;
* ) echo "Sorry, answer not recognized";;
esac
```

175

Vòng lặp for

- Cú pháp

```
for variable in values
do
statements
done
```

– Ví dụ:

```
for foo in apple banana 43
do
echo $foo
done
```

```
for file in $(ls f*.sh); do
lpr $file
done
```

176

for (tt)

- Có thể sử dụng vòng lặp for như cú pháp C

```
for (( variable assignment ; condition ; iteration))
do
.....
repeat all statements between do and
done until condition is TRUE
done
```

– Ví dụ

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

177

Vòng lặp while

- Cú pháp:

```
while condition do
statements
done
```

• Ví dụ:

```
echo "Enter password"
read trythis
while [ "$trythis" != "secret" ]; do
echo "Sorry, try again"
read trythis
done
```

```
foo=1
while [ "$foo" -le 20 ]
do
echo "Here we go again"
foo=$((foo+1))
done
```

178

Vòng lặp until

- Cú pháp:
 - condition đúng
→ thoát vòng lặp
- Ví dụ:

```
until condition
do
statements
done
```

```
echo 'Please enter a number'
read n
until [ $n -lt 10 ]
do
echo $n
n='expr $n - 1'
done
```

179

Lệnh break, continue, exit

- **break**: thoát khỏi vòng lặp
- **exit**: thoát khỏi chương trình, trở về dấu nhắc lệnh
- **continue**: không thực hiện các lệnh còn lại trong vòng lặp mà quay lên thực hiện vòng lặp kế tiếp

180

2.4. Xử lý chuỗi, văn bản

- Sử dụng Regular expressions
- Xử lý chuỗi

181

Regular expressions

- Là dạng biểu thức dùng để so khớp văn bản, được dùng trong tìm kiếm chuỗi

biểu thức	Ý nghĩa	Ví dụ
^	bắt đầu của dòng	^tux → đầu dòng là chuỗi tux.
\$	Kết thúc của dòng	tux\$ → cuối dòng là chuỗi tux.
.	Đại diện cho chỉ 1 ký tự	Hack. → sau Hack là một ký tự bất kỳ (Hack1, Hacki)
[]	So khớp với các ký tự trong []	coo[kl] → cook hoặc cool
[^]	Ngoại trừ các ký tự trong []	9[^01] → 92, 93,... ngoại trừ 91 và 90

182

Regular expressions (tt)

biểu thức	Ý nghĩa	Ví dụ
[m-n]	So khớp với các ký tự từ m đến n	[1-5] → các ký tự số từ 1 đến 5
?	Ký tự trước ? Chỉ xuất hiện 0-1 lần	colou?r → color hoặc colour, không thể là colouur
+	Ký tự trước + phải xuất hiện ít nhất một lần	Rollno-9+ → Rollno-99 hoặc Rollno-9,... nhưng không thể là Rollno-
*	Ký tự trước * có thể xuất hiện từ 0 đến nhiều lần	co*I → cl, col, cool,...
()	Nhóm ký tự trong () được xử lý như một thực thể	ma(tri)?x → max hoặc matrix

183

Regular expressions (tt)

biểu thức	Ý nghĩa	Ví dụ
{n}	Phần tử đứng trước phải xuất hiện n lần	[0-9]{3} → ba ký tự số từ 0-9 liên tục
{n,}	Phần tử đứng trước phải xuất hiện ít nhất n lần	[0-9]{2,} → tối thiểu là hai ký tự số từ 0-9
{n, m}	Phần tử đứng trước phải xuất hiện từ n đến m lần	[0-9]{2,5} → từ 2 đến 5 ký tự số liên tục
	Hoặc	Oct (1st 2nd) → Oct 1st hoặc Oct 2nd.
\	Chỉ định ký tự đặc biệt như \$, ^, ., *, +	a\.b → a.b (bắt buộc có ký tự dấu chấm (.) giữa a và b)

184

Regular expressions (tt)

- Ví dụ:

- (?[a-zA-Z]+ ?)
- [0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}
- Hoặc [[:digit:]]{1,3}\.[[:digit:]]{1,3}\.[[:digit:]]{1,3}\.[[:digit:]]{1,3}

185

Xử lý chuỗi

- substring: Trả về chuỗi con trong chuỗi

- Cú pháp:

- \${string:position}: trả về chuỗi con của chuỗi string bắt đầu ở vị trí position.
- \${string:position:length}: trả về length ký tự trong chuỗi string bắt đầu ở vị trí \$position.

- Ví dụ:

- string="this is a substring test"
- substring=\${string:10:9}
- echo \$substring → substring
- var="Welcome to the geekstuff"
- echo \${var:15} → geekstuff

186

Xử lý chuỗi (tt)

- Toán tử # và ##

- \${string#substring}: trả về chuỗi đã loại bỏ chuỗi ngắn nhất match với \$substring từ đầu chuỗi \$string
- \${string##substring}: trả về chuỗi đã loại bỏ chuỗi dài nhất match với \$substring từ đầu chuỗi \$string
 - Ví dụ:
 - s1="Hi Mary. How are you."
 - s2=\${s1#* } → Mary. How are you
 - s3=\${s1##* } →you

187

Xử lý chuỗi (tt)

- Toán tử % và %%

- \${string%substring}: trả về chuỗi đã loại bỏ chuỗi ngắn nhất match với substring từ sau chuỗi string.
- \${string%%substring}: trả về chuỗi đã loại bỏ chuỗi dài nhất match với substring từ sau chuỗi string
 - Ví dụ:
 - s1="Hi Mary. How are you."
 - s4=\${s1% *} → Hi Mary. How are
 - s3=\${s1%% *} →Hi

188

Xử lý chuỗi (tt)

- \${#string}: độ dài của \$string

- Tìm kiếm và thay thế:

- \${string/s1/s2}

- Trả về chuỗi đã được thay thế s1 bằng s2 tại vị trí tìm thấy đầu tiên.

```
s1="Hi Mary. How are you"
s2="Mary"
s3="Don"
s4=${s1/$s2/$s3}
echo "s4=$s4"
```

- \${string//s1/s2}

- Trả về chuỗi đã được thay thế tất cả các s1 bằng s2

```
s="red - green - blue"
s1=${s//-/;}
echo "s1=$s1"
```

189

Xử lý chuỗi (tt)

- Tìm kiếm và thay thế:

- \${string/#s1/s2}

- Trả về chuỗi đã được thay thế s1 bằng s2 nếu s1 tìm thấy ở đầu chuỗi

- \${string/%s1/s2}

- Trả về chuỗi đã được thay thế s1 bằng s2 nếu s1 tìm thấy ở cuối chuỗi

190

2.5. Mảng

- Mảng thông thường
- Mảng kết hợp

191

Mảng thông thường

- Khai báo và khởi tạo:
 - `array_var=(1 2 3 4 5 6)`
 - `array_var[0]="test1"`
`array_var[1]="test2"`
`array_var[2]="test3"`
`array_var[3]="test4"`
`array_var[4]="test5"`
`array_var[5]="test6"`
 - Truy xuất phần tử mảng:
 - In một phần tử: `echo ${array_var[2]}` → 3
 - In tất cả các phần tử: `$ echo ${array_var[*]}`
 → test1 test2 test3 test4 test5 test6
 - Truy xuất số phần tử mảng: `$ echo ${#array_var[*]}` → 6

192

Mảng kết hợp

- Là loại mảng mà trường index có thể là một chuỗi
- Khai báo:
 - \$ declare -A ass_array
- Thêm phần tử vào mảng:
 - \$ ass_array=([index1]=val1 [index2]=val2)
 - Hoặc: \$ ass_array[index1]=val1
 - Ví dụ:
 - \$ declare -A fruits_value
 - \$ fruits_value=([apple]='100dollars' [orange]='150 dollars')

193

Mảng kết hợp

- Truy xuất nội dung phần tử mảng:
 - \$ echo "Apple costs \${fruits_value[apple]}"
→ Apple costs 100 dollars
- Liệt kê các trường index trong mảng:
 - \$ echo \${!array_var[*]}
 - Hoặc: \$ echo \${!array_var[@]}
 - Ví dụ:
 - \$ echo \${!fruits_value[*]} → orange apple

194

2.6. Hàm

- Cấu trúc hàm:
- Ví dụ:

```
function_name ()
{
    statements
}
```

```
func()
{
    echo "Function foo is executing"
}
echo "script starting"
func
echo "script ended"
```

Kết quả

script starting
 Function foo is executing
 script ending

195

Hàm (tt)

- Biến cục bộ (chỉ có hiệu lực bên trong hàm), khai báo dùng từ khoá local
- Biến toàn cục: khai báo không dùng từ khoá local (global)
- Phạm vi lưu trữ của biến cục bộ không còn hiệu lực khi hàm kết thúc

196

Hàm (tt)

- Ví dụ sử dụng biến cục bộ và toàn cục:

```
sample text="global variable"
foo() {
    local sample_text="local variable"
    echo "Function foo is executing"
    echo $sample_text
}
echo "script starting"
echo $sample_text
foo
echo "script ended"
echo $sample_text
exit 0
```

script starting
 global variable **Kết quả**
 Function foo is executing
 local variable
 script ended
 global variable

197

Hàm (tt)

- Hàm có thể trả về một giá trị
 - Trả về giá trị số: dùng lệnh return
 - Trả về chuỗi: dùng lệnh echo
 - Ví dụ:

*foo() {
 ...
 return 0
 }*

Hàm trả về số

*foo() {
 echo "string value"
 }
 ...
 x= \$(foo)*

**Hàm trả về
chuỗi**

198

Hàm (tt)

- Hàm và cách truyền tham số:

- shell không có cách khai báo tham số cho hàm như c, c++, ...
- Việc truyền tham số cho hàm tương tự truyền tham số trên dòng lệnh, ví dụ, để truyền tham số cho hàm foo, ta gọi như sau:
 - foo “param1” “param2” “param3”, ...
 - Để lấy nội dung đối số, bên trong hàm, ta sử dụng các biến môi trường \$*, \$1, \$2, ...

199

Hàm (tt)

- Ví dụ truyền tham số cho hàm:

```
yes_or_no() {
    echo "Is your name $* ?"
    while true
        do
            echo -n "Enter yes or no: "
            read x
            case "$x" in
                y | yes ) return 0;;
                n | no ) return 1;;
                * ) echo "Answer yes or no"
            esac
        done
}
```

```
echo "Original parameters are $*"
if yes_or_no "$1"
then
    echo "Hi $1, nice name"
else
    echo "Never mind"
fi
```

```
$ ./my_name.sh Rick Neil
Original parameters are Rick Neil
Is your name Rick ?
Enter yes or no: yes
Hi Rick, nice name
```

200

2.7. Các lệnh nội tại của shell

- Lệnh **:** (lệnh rỗng) tương đương với true

– Ví dụ: kiểm tra nếu file fred tồn tại thì không làm gì cả

```
rm -f fred
if [ -f fred ]; then
:
else
    echo file fred does not exist
fi
exit 0
```

- Lệnh **exec**: dùng để gọi một lệnh khác và gọi exit khi kết thúc lệnh

– Ví dụ:

```
echo "Try to execute mc program"
exec mc
echo "you can not see this message !"
```

201

Các lệnh nội tại của shell (tt)

- Lệnh exit n: thoát khỏi shell và trả về mã lỗi n (0: không có lỗi)
- Print (tương tự printf trong c), nhưng không hỗ trợ dạng số có dấu chấm động

– Ký tự “\”:

Chuỗi thoát (escape sequence)	ý nghĩa
\\\	Cho phép hiển thị ký tự \ trong chuỗi
\a	Phát liêng chuông (beep)
\b	Xóa backspace
\f	Đẩy dòng
\n	Sang dòng mới
\r	Về đầu dòng
\t	Căn tab ngang
\v	căn tab dọc
\ooo	Kí tự đơn với mã là ooo

202

Các lệnh nội tại của shell (tt)

- Print (tt)

- Sử dụng ký tự “%” để định dạng xuất, các tham số của lệnh print cách nhau một khoảng trắng

Kí tự định dạng	Ý nghĩa
d	Số nguyên
c	Ký tự
s	chuỗi
%	hiển thị ký hiệu %

- Ví dụ

```
$ printf "Your name %s. It is nice to meet you \n" NV An
Your name NV An. It's nice to meet you .
```

```
$ printf "%s %d\t %s" "Hi There" "15" "people"
Hi There 15 people
```

203

Các lệnh nội tại của shell (tt)

- Lệnh set

- Dùng để gán giá trị cho các biến môi trường \$1, \$2, \$3,... bằng cách loại bỏ các khoảng trắng không cần thiết và đặt nội dung của chuỗi truyền cho nó vào các biến tham số

\$ set This is parameter	Ví dụ
\$echo \$ 1	
This	
\$echo \$3	
parameter	

204

Các lệnh nội tại của shell (tt)

- Lệnh set (tt)

```
echo Current date is $(date)
set $(date)
echo The month is $2
echo The year is $6
exit 0
```

Kết quả kết xuất

```
./set_use.sh
Current date Fri March 13    16:06:16 EST 2001
The month is March
The year is 2001
```

205

Các lệnh nội tại của shell (tt)

- Lệnh unset: loại bỏ biến khỏi shell

```
foo="Hello World"
echo $foo
unset foo
echo $foo
```

206

Chương 3

LẬP TRÌNH C TRÊN LINUX

207



NỘI DUNG

- Giới thiệu ngôn ngữ C trên Linux
- Trình biên dịch gcc
- Nhắc lại cú pháp ngôn ngữ C
- Giới thiệu công cụ hỗ trợ biên dịch Makefile

208

3.1. Ngôn ngữ C trên Linux

- Ngôn ngữ C hỗ trợ rất tốt cho lập trình trên Linux. Tuy nhiên nó không phải là lựa chọn duy nhất. Có thể dùng Pascal , Assembler, Perl, Java, PHP, Python...
- Chương trình Linux tồn tại trên hai dạng :
 - Thực thi (file binary) giống như file ***.exe** trong DOS
 - Thông dịch (script) giống như file ***.bat** .
 - Hai dạng file này có thể hoán đổi cho nhau . Để chạy chương trình cần cấp quyền thực thi “x”

209

Ngôn ngữ C trên Linux (tt)

- Cộng đồng mã nguồn mở GNU cung cấp bộ biên dịch C/C++ bao gồm :
 - gcc trình biên dịch C
 - g++ trình biên dịch C++
 - gdb Debug
 - GNU make Trình quản lý mã nguồn và trợ giúp biên dịch
 - bash shell

210

Ngôn ngữ C trên Linux (tt)

- 3 công cụ cần thiết để viết lập trình trên ngôn ngữ C :
 - Trình soạn thảo (text editor): vi, gedit, ...
 - Trình biên dịch (compiler) : GNU Compiler Collection (GCC), CC
 - \$ gcc
 - \$ which gcc
 - Thư viện chuẩn của C (C standard library) : glibc
 - \$ locate glibc

211

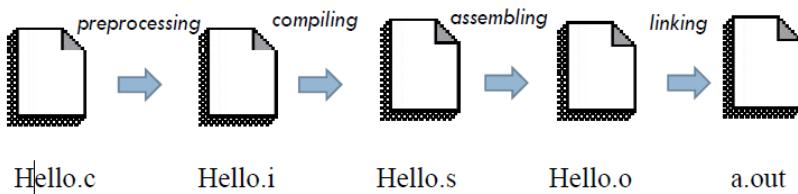
3.2. Trình biên dịch gcc

- Hệ điều hành UNIX luôn kèm theo bộ dịch ngôn ngữ lập trình C với tên là cc (C compiler). Trong Linux, bộ dịch có tên là **gcc** (GNU C Compiler).
- Quá trình biên dịch của gcc trải qua 4 bước:
 - Tiền xử lý (*preprocessing*)
 - \$gcc -E hello.c > hello.i
 - Biên dịch sang hợp ngữ (*compiling*)
 - \$gcc -S hello.i

212

Trình biên dịch gcc (tt)

- Chuyển hợp ngữ sang mã máy (*assembling*)
 - \$gcc -c hello.s
- Thiết lập các liên kết (*linking*)
 - \$gcc hello.o



213

Trình biên dịch gcc (tt)

- Các tùy chọn của gcc

Tùy chọn	Công dụng
-o FILE	Chỉ định tên của file output (khi biên dịch thành file thực thi, nếu không có -o filename thì tên file mặc định sẽ là a.out)
-c	Chỉ biên dịch mà không linking (i.e. chỉ tạo ra object file *.o)
-IDIRNAME	Chỉ tên thư mục DIRNAME là nơi chứa các file header (.h) mà gcc sẽ tìm trong đó (mặc định gcc sẽ tự tìm ở các thư mục chuẩn /usr/include, ...)
-LDIRNAME	Chỉ tên thư mục DIRNAME là nơi chứa các thư viện (.a, .so) mà gcc sẽ tìm trong đó (mặc định gcc sẽ tự tìm ở các thư mục chuẩn /usr/lib, ...)
-O [n]	Tối ưu mã thực thi tạo ra (e.g. -O2, -O3, hoặc -O)
-g	Chèn thêm mã phục vụ công việc debug
-E	Chỉ thực hiện bước tiền xử lý (preprocessing) mà không biên dịch
-S	Chỉ dịch sang mã hợp ngữ chứ không linking (i.e. chỉ tạo ra file *.s)
-lfoo	Link với file thư viện có tên là libfoo (e.g. -lm, -lpthread)
-ansi	Biên dịch theo chuẩn ANSI C/C++ (sẽ cảnh báo nếu code không chuẩn)

214

Trình biên dịch gcc (tt)

- Trình biên dịch gcc thường nằm trong **/usr/bin** hoặc **/usr/local/bin**. Khi biên dịch nó cần sự hỗ trợ của các file C header trong **/usr/include** hoặc **/usr/local/include**
- Các thư viện liên kết nằm trong **/lib** hoặc **/usr/local/lib**.
- Các thư viện chuẩn của gcc nằm trong **/usr/lib/gcc-lib**
- Chương trình nhị phân có thể nằm ở bất kỳ đâu nhưng khi thực thi ta cần chỉ đường dẫn thông qua biến môi trường PATH
- Cài biến môi trường : \$echo \$PATH
– PATH = **/bin:/user/bin:/user/local/bin:**

215

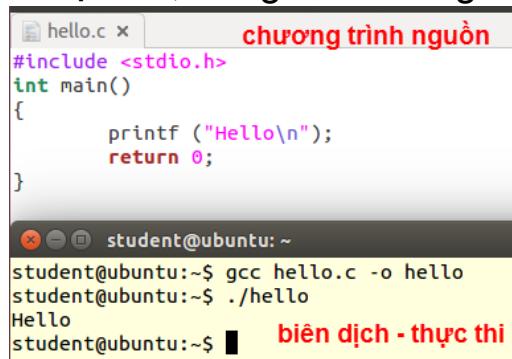
Trình biên dịch gcc (tt)

- Chương trình đầu tiên

- Dòng lệnh đầu tiên chỉ cho **gcc** biên dịch và liên kết file nguồn **hello.c**, tạo ra tệp tin thực thi, bằng cách dùng đối số **-o hello**.

- Dòng thứ hai thực hiện chương trình

- Dòng thứ ba hiển thị kết quả



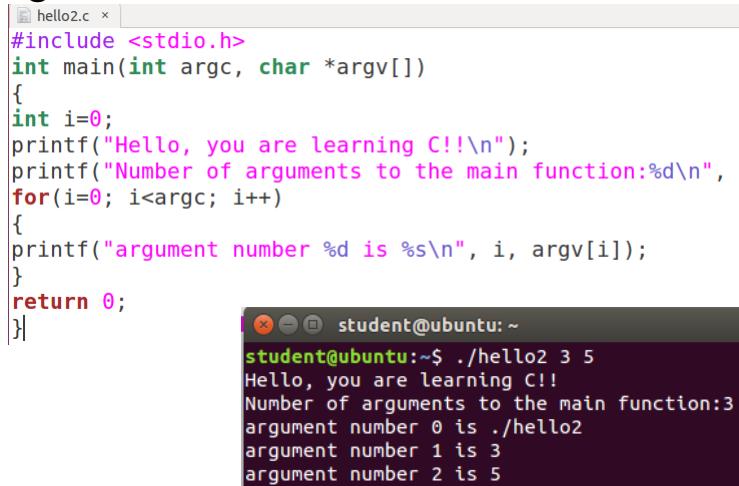
```
hello.c x           chương trình nguồn
#include <stdio.h>
int main()
{
    printf ("Hello\n");
    return 0;
}

student@ubuntu: ~
student@ubuntu:~$ gcc hello.c -o hello
student@ubuntu:~$ ./hello
Hello
student@ubuntu:~$ ■   biên dịch - thực thi
```

216

Trình biên dịch gcc (tt)

- Chương trình thứ hai: thực thi với tham số



```
hello2.c x
#include <stdio.h>
int main(int argc, char *argv[])
{
int i=0;
printf("Hello, you are learning C!!\n");
printf("Number of arguments to the main function:%d\n", argc);
for(i=0; i<argc; i++)
{
printf("argument number %d is %s\n", i, argv[i]);
}
return 0;
}

student@ubuntu:~$ ./hello2 3 5
Hello, you are learning C!!
Number of arguments to the main function:3
argument number 0 is ./hello2
argument number 1 is 3
argument number 2 is 5
```

217

3.3 Nhắc lại cú pháp ngôn ngữ C

- Cấu trúc một chương trình C
- Các lệnh nhập xuất
- Các cấu trúc điều khiển
- Hàm

218

Cấu trúc một chương trình C

- Phần khai báo chèn các tập tin tiêu đề (header file) vào chương trình. Có hai cách để xác định tập tin theo sau chỉ thị #include:
 - < >: tập tin thư viện như stdio.h, conio.h
 - “ ” : các tập tin tiêu đề do người lập trình tạo ra.
- Phần khai báo các biến toàn cục, các nguyên mẫu hàm
- Phần định nghĩa hàm **main**
- Phần định nghĩa các hàm khác

219

Các lệnh nhập xuất

- Xuất dữ liệu ra màn hình
 - **printf**: khai báo #include <stdio.h>
 - Cú pháp:
 - printf (<chuỗi định dạng>[, <đối số 1>, ...]);
 - Chuỗi định dạng: đặt giữa “ ”, bao gồm
 - Văn bản thường
 - Ký tự điều khiển: sau \

Ký tự điều khiển	Ý nghĩa
\a	Tiếng chuông
\b	Lùi lại một bước
\n	Xuống dòng
\t	Dấu tab
\\\	In dấu \
\?	In dấu ?
\“	In dấu “

220

Các lệnh nhập xuất

- Xuất dữ liệu ra màn hình
 - Chuỗi định dạng (tt)
 - Đặc tả (conversion specifier) gồm dấu phần % và một ký tự: xác định kiểu của biến muốn xuất. Biến muốn xuất sẽ được đặt ở phần đối số. Nếu muốn xuất nhiều biến thì các biến sẽ được liệt kê cách nhau bằng dấu phẩy.

Đặc tả	Ý nghĩa	Kiểu dữ liệu phù hợp
%c	Ký tự đơn	char
%d	Số nguyên có dấu	int, short, long
%f	Số thực	float, double
%s	Chuỗi ký tự	char[], char*
%u	Số nguyên không dấu	unsigned int/short/long

221

Các lệnh nhập xuất

- Xuất dữ liệu ra màn hình
 - Ví dụ:
 - int a = 2912, b = 1706;
 - printf ("%d cong %d bang %d", a, b, a + b);
 - Kết quả: 2912 cong 1706 bang 4618
 - Định dạng kiểu số, sử dụng cú pháp

`%nd` Dùng n ô để in số nguyên.

`%n.kf` Dùng n ô để in số thực và lấy k số lè.
 n = 0 hoặc bỏ nếu không quan tâm số ô)

222

Các lệnh nhập xuất

- Xuất dữ liệu ra màn hình

```
#include <stdio.h>
void main()
{
    int a = 2912, b = 176;
    float c = 176.85;
    printf("%10d", a);
    printf("%10d", b);
    printf("%10.2f", c);
    printf("%.2f", c);
}
```

Ví dụ

Kết quả (theo ô trên màn hình):

					2	9	1	2
						1	7	6
				1	7	6	.	8
1	7	6	.	8	5			

223

Các lệnh nhập xuất

- Nhập dữ liệu từ bàn phím
 - Cú pháp: scanf (<chuỗi định dạng>[, <đối số 1>, ...]);
 - Đối số là tên biến được đặt trước dấu &.
 - Chuỗi định dạng tương tự như printf nhưng không chứa các văn bản thường
 - Nhập các biến cách nhau khoảng trắng, tab hoặc xuống dòng.

```
#include <stdio.h>
void main()
{
    int a, b;
    scanf("%d%d", &a, &b);
}
```

224

Công cụ hỗ trợ biên dịch Makefile

- Những vấn đề khi biên dịch
 - Một chương trình đơn giản => chỉ có một vài file
- Chương trình lớn (thực tế):
 - Nhiều dòng lệnh
 - Nhiều module
 - Nhiều người viết
- Vẫn đề:
 - Khó quản lý một file lớn
 - Khi có thay đổi: thời gian biên dịch lâu
 - Nhiều người lập trình: không thể thay đổi cùng một file đồng thời
 - Chương trình chia thành nhiều module

225

Công cụ hỗ trợ biên dịch Makefile (tt)

- Giải pháp: chia project thành nhiều file
 - Chia thành nhiều module một cách hợp lý
 - Thời gian biên dịch ngắn nếu có thay đổi
 - Dễ dàng bảo trì cấu trúc project
- Make file là gì?
 - Công cụ hỗ trợ biên dịch
 - Chỉ biên dịch những phần cần thiết
 - Biên dịch trên nhiều platform khác nhau

226

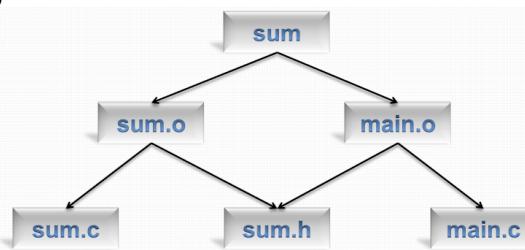
Công cụ hỗ trợ biên dịch Makefile (tt)

- Tập tin Makefile: dạng **script** chứa các thông tin
 - Cấu trúc project (file, sự phụ thuộc)
 - Các lệnh để tạo file
 - Lệnh **make** sẽ đọc nội dung Makefile, hiểu kiến trúc của project và thực thi các lệnh

227

Công cụ hỗ trợ biên dịch Makefile (tt)

- Cấu trúc project
 - Cấu trúc và sự phụ thuộc của project có thể được biểu diễn bằng một DAG (Directed Acyclic Graph)
- Ví dụ:
 - Chương trình chứa 3 file: main.c, sum.c, sum.h
 - File sum.h được dùng bởi cả 2 file main.c và sum.c
 - File thực thi là sum
-



228

Công cụ hỗ trợ biên dịch Makefile (tt)

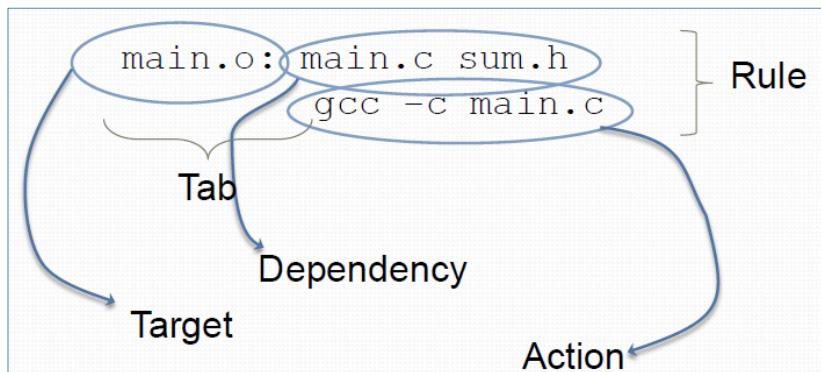
- Nội dung của Makefile

```
sum: main.o sum.o
      gcc -o sum main.o sum.o
main.o: main.c sum.h
      gcc -c main.c
sum.o: sum.c sum.h
      gcc -c sum.c
```

229

Công cụ hỗ trợ biên dịch Makefile (tt)

- Cú pháp



230

Công cụ hỗ trợ biên dịch Makefile (tt)

- target
 - target với các phụ thuộc:
 - sum.o: sum.csum.h
 - all: sum
 - target không có phụ thuộc:
 - clean:
 - Các target không bắt đầu bằng khoảng trắng hoặc TAB
- Action
 - Mỗi target sẽ kèm theo 0 hoặc nhiều action.
 - Mỗi dòng chứa action phải bắt đầu bằng TAB
 - <TAB> gcc-o sum sum.omain.o
 - <TAB> rm-fr\${OBJ}
 - <TAB> gcc-c sum.c

231

Công cụ hỗ trợ biên dịch Makefile (tt)

- Một số cú pháp sử dụng trong makefile
 - Các tham số:
 - -k: tiếp tục chạy nếu có lỗi
 - -n: in ra (nhưng không thực thi) các thao tác
 - -f <filename>: chỉ định tập tin biên dịch. Mặc định là makefile hoặc Makefile trong thư mục hiện hành
 - Để tạo file thực thi, cần truyền tên file thực thi cho make, nếu không make sẽ chọn tên file đích đầu tiên tìm thấy làm tên của chương trình
 - Ví dụ:
 - \$make
 - Tương đương với \$make makefile
 - Từ khóa all: tập tin đích đầu tiên, các file đích khác đều phụ thuộc vào tập tin này
 - all : myapp myapp1

232

Chương 4

PHP & MYSQL

233



Nội dung

-
- Cơ bản về PHP
 - Form và controls
 - Xử lý trang web
 - PHP và MySQL

234

4.1. Cơ bản về PHP

- Cài đặt PHP.
- Giới thiệu PHP.
- Cấu trúc và cú pháp của PHP.
- Chương trình đầu tiên.
- Hằng và biến.
- Một số hàm toán học.
- Các kiểu dữ liệu.
- Chuyển kiểu dữ liệu.
- Các toán tử
- Lệnh if/else.
- Lệnh for/foreach.
- Lệnh while, do/while.
- Hàm.
- Mảng.

235

Giới thiệu PHP

- PHP (Personal Home Page) ra đời năm 1994 do phát minh của Rasmus Lerdorf
- PHP bắt đầu được sử dụng trong môi trường chuyên nghiệp và trở thành chữ viết tắt của Hypertext Preprocessor.
- PHP là ngôn ngữ mã nguồn mở, là kịch bản trình chủ (server script) chạy trên phía server (server side).

236

Giới thiệu PHP (tt)

- Cho phép xây dựng ứng dụng web với các cơ sở dữ liệu mySQL, Oracle, SQL Server...
- PHP là công nghệ phía server (server-side), không phụ thuộc vào môi trường (cross-platform), có thể sử dụng được trên nhiều hệ điều hành: Unix, Linux, Windows...
- Do PHP dễ học, thời gian tạo các trang Web động tương đối ngắn hơn so với các ngôn ngữ khác nên PHP đã nhanh chóng trở thành một ngôn ngữ lập trình Web phổ biến

237

Cài đặt PHP

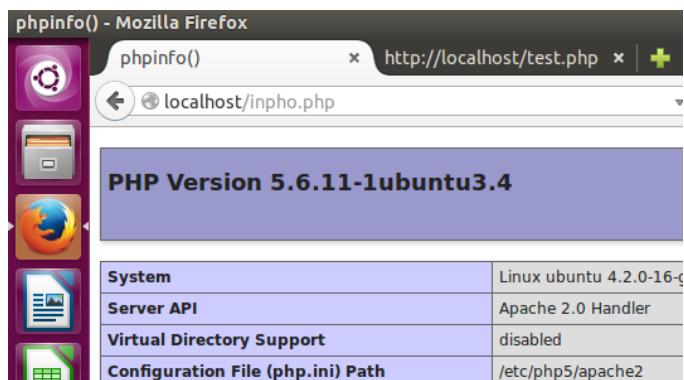
- Cài PHP
 - Cập nhật hệ thống: sudo apt-get update
 - Cài apache:
 - sudo apt-get install apache2
 - sudo apt-get install php5 libapache2-mod-php5
 - Có thể tìm các gói cần thiết để phát triển bằng lệnh:
 - suso apt-cache search php5
 - Xem thư mục web của apache:
 - cat /etc/apache2/sites-available/default-ssl.conf
 - Ví dụ trong Ubuntu15.04: **/var/www/html**

238

Cài đặt PHP (tt)

- Kiểm tra hoạt động của PHP
 - Chuyển đến thư mục /var/www/html, gõ lệnh
 - sudo nano info.php
 - Nhập nội dung sau:


```
<?php
echo phpinfo();
?>
```
 - Từ trình duyệt, nhập:
 - localhost/info.php



239

Cài đặt PHP (tt)

- Cài mysql
 - sudo apt-get install mysql-server
- Cài phpmyadmin
 - sudo apt-get install phpmyadmin
- Cấu hình phpmyadmin
 - Mở file /etc/apache2/apache2.conf
 - Thêm vào cuối file dòng:
 - Include /ect/phpmyadmin/apache.conf
 - Khởi động lại apache2:
 - sudo service apache2 restart
- Kiểm tra hoạt động phpmyadmin:
 - localhost/phpmyadmin

240

Cấu trúc và cú pháp của PHP

- Trang PHP là sự phối hợp của các thẻ HTML và PHP
- Các phương pháp nhúng code PHP trong trang HTML:

Thẻ mở	Thẻ đóng	Ghi chú
<?	?>	Cần cấu hình server cho phép hỗ trợ shorthand-support → ít dùng
<?php	?>	Thường dùng
<script language="php">	<script>	ít dùng

241

Cấu trúc và cú pháp của PHP

- Ví dụ

```

< ?php
    $num = 1 + 2;
    echo $num;
? >

< html >
< head >
< title > My First PHP Program < /title >
< /head >
< body >
    < ?php
        echo "I'm a lumberjack.";
    ? >
< /body >
< /html >

```

242

Cấu trúc và cú pháp của PHP

- Ta có thể kết hợp các thẻ HTML trong code PHP

```
< html >
< head >
< title > My First PHP Program < /title >
< /head >
< body >
< ?php
    echo "< h1 > I'm a lumberjack. < /h1 >";
    echo "< h2 > And I'm okay. < /h2 >";
?
< /body >
< /html >
```

I'm a lumberjack.

And I'm okay.

243

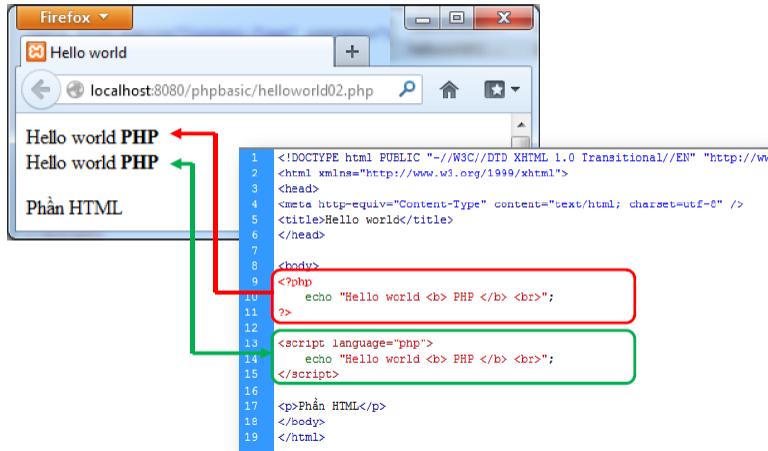
Cấu trúc và cú pháp của PHP

- Cú pháp PHP tương tự ngôn ngữ C
- Một số lưu ý:
 - Cuối câu lệnh có dấu ;
 - Mỗi phương thức đều bắt đầu { và đóng bằng dấu }
 - Ghi chú (comment):
 - Sử dụng ký hiệu // hoặc # để giải thích cho mỗi câu ghi chú
 - Sử dụng /* và */ cho mỗi đoạn ghi chú

244

Chương trình đầu tiên

Ví dụ “Hello world”



```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/1999/xhtml">
2 <html xmlns="http://www.w3.org/1999/xhtml">
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
5 <title>Hello world</title>
6 </head>
7
8 <body>
9 <?php
10 echo "Hello world <b> PHP </b> <br>";
11 ?>
12
13 <script language="php">
14 echo "Hello world <b> PHP </b> <br>";
15 </script>
16
17 <p>Phản HTML</p>
18 </body>
19 </html>

```



Sử dụng phương thức `echo "Nội dung"` để xuất thông tin lên trình duyệt.

245

Kiểu dữ liệu

- Kiểu dữ liệu trong PHP khá đa dạng, được chia thành 2 nhóm chính sau:
 - Scalar (cơ bản): boolean, int, float, string,...
 - Composite (đa hợp): array, object,...
- Kiểu dữ liệu trong PHP được khởi gán và chuyển đổi kiểu một cách tự động trong quá trình khai báo hằng và biến.

246

Kiểu dữ liệu (tt)

- Thay đổi kiểu dữ liệu
 - Ép kiểu (như ngôn ngữ C)
 - Tenbien= (kieudulieu) Tenbien;
 - Sử dụng hàm settype
 - settype(\$Tên_Biến, "Data_type");
 - Hàm gettype: trả về kiểu của biến

```
<?php
$don_gia = 7000;
$so_luong = 900;
$thanh_tien =
(double)($so_luong*$don_gia);
?>
```

```
<?php
$var="12-ABC";
$check=true;
settype($var,"integer");
echo $var;
echo "<br>";
settype($check,"string");
echo $check;
?>
```

247

Kiểu dữ liệu (tt)

- Kiểm tra kiểu dữ liệu: các hàm có tiền tố `is_`kieudulieu, các hàm này trả về true/false nếu kiểu dữ liệu của biến được kiểm tra là đúng
 - `is_array`, `is_bool`, `is_double`, `is_float`, `is_integer`, `is_long`, `is_real`, `is_string`, `is_object`,...

Ví dụ

```
$var = "test";
if (isset($var))
    echo "Variable is Set";
if (empty($var))
    echo "Variable is Empty";
```

248

Biến

- Qui tắc đặt tên cho biến:
 - Sử dụng tiền tố \$ trước tên biến, sau là ký tự hoặc dấu _, tiếp đó là ký tự, ký số hoặc dấu _
 - Biến có phân biệt chữ hoa, thường
- Khai báo:
 - Khai báo biến không có kiểu dữ liệu
 - Nên gán giá trị khởi đầu cho biến lúc khai báo
 - Cú pháp: \$tenbien [=giatri];

249

Biến

// Variable Variables

```
$varname = "my_variable";
$varname = "xyz";           // tạo biến: $my_variable = "xyz"
echo $varname."<br>";     // output: "my_variable"
echo $my_variable."<br>"; // output: "xyz"

$myvarname = "123";
$$myvarname = "456";        // tạo biến: $myvarname = "456"
echo ${'123'}."<br>";    // output: "456"
```

Ví dụ

250

Biến (tt)

- Các hàm kiểm tra sự tồn tại của biến:
 - Kiểm tra tồn tại: isset (\$tenbien1, \$tenbien2,...)
 - Trả về TRUE: tất cả các biến đều có giá trị
 - Trả về FALSE: nếu một biến bất kỳ không có giá trị

```
<?php
if(isset($_POST[ "btSubmit"], $_POST[ "username"]))
    echo "Xin chào ".$_POST["username"];
else
    echo "Vui lòng nhập tên đăng nhập";
?>
```

251

Biến (tt)

- Các hàm kiểm tra sự tồn tại của biến:
 - Kiểm tra giá trị rỗng empty(tenbien): kiểm tra biến có giá trị rỗng hay không
 - Trả về TRUE: biến có giá trị rỗng
 - Trả về FALSE: biến có giá trị khác rỗng
 - Các giá trị được xem là rỗng: "" (chuỗi rỗng), NULL, 0 (khi kiểu là integer), FALSE, array(), var \$var (biến trong lớp được khai báo nhưng không có giá trị)
 - Kiểm tra biến kiểu số is_numeric (tenbien):
 - Trả về TRUE: biến có giá trị kiểu số
 - Trả về FALSE: biến có giá trị không phải kiểu số

252

Biến (tt)

- **Tầm vực của biến.**

- Biến cục bộ: được khai báo trong hàm, khi ra khỏi hàm => biến cục bộ và giá trị của nó sẽ bị hủy bỏ

Ví dụ:

```
<?php
    $a = 1; // toàn cục
    function Test()
    {
        echo $a; // cục bộ
    }
    Test(); → không có giá trị
    echo $a; → 1
?>
```

253

-

Biến (tt)

- **Biến toàn cục:**

- Có thể truy xuất bất cứ nơi nào trong trang
- Truy cập biến toàn cục trong hàm: dùng từ khóa global phía trước biến hoặc dùng `$GLOBALS["tenbien"]`

```
<?php
    $a = 1;
    $b = 2;
    function Sum()
    {
        $_GLOBALS['b'] = $_GLOBALS['a'] + $_GLOBALS['b'];
    }
    Sum();
    echo $b; → 3
?>
```

```
<?php
    $a = 1;
    $b = 2;
    function Sum()
    {
        global $a, $b;
        $b = $a + $b;
    }
    Sum();
    echo $b; → 3
?>
```

254

Biến (tt)

– Biến static:

- Phía trước tên biến có từ khóa static
- Không mất đi giá trị khi ra khỏi hàm
→ giữ nguyên giá trị trước đó khi hàm được gọi một lần nữa

```
<?php
function Test()
{
    static $a = 0;
    echo $a;
    $a++;
}
Test(); →0
Test(); →1
Test(); →2
?>
```

255

Hằng

- Khai báo và sử dụng hằng
 - Sử dụng phát biểu define
 - Ví dụ: define ("MAX" , 100);

```
< html >
< head >
< title > My Movie Site < /title >
< /head >
< body >
< ?php
    define ('FAVMOVIE', 'The Life of Brian');
    echo 'My favorite movie is ';
    echo FAVMOVIE;
? >
< /body >
< /html >
```

256

Hằng

- Kiểm tra hằng: sử dụng hàm defined để kiểm tra sự tồn tại của hằng
- Ví dụ:

```
if ( defined("PI") )
    echo PI;
```

257

Một số hàm toán học

```
max ( array $values )
max ($value1, $value2 [, $value3...] )
min ( array $values)
min ($value1, $value2 [, $value3...] )
```

258

Một số hàm toán học (tt)

```

<?php
echo max(1, 3, 5, 6, 7); // 7
echo max(array(2, 4, 5)); // 5

echo max(0, 'hello'); // 0
echo max('hello', 0); // hello
echo max(-1, 'hello'); // hello

// With multiple arrays, max compares from left to right
// so in our example: 2 == 2, but 4 < 5
$val = max(array(2, 4, 8), array(2, 5, 7)); // array(2, 5, 7)

// If both an array and non-array are given, the array
// is always returned as it's seen as the largest
$val = max('string', array(2, 5, 7), 42); // array(2, 5, 7)
?>

```

259

Một số hàm toán học (tt)

- int count (\$arrvar [, int \$mode])

```

$aa[0] = 1;
$aa[1] = 3;
$aa[2] = 5;
$result = count($aa);
// $result == 3

```

```

$result = count(null);
// $result == 0

$result = count(false);
// $result == 1

```

```

<?php
$food = array('fruits' => array('orange', 'banana', 'apple'),
              'veggie' => array('carrot', 'collard', 'pea'));

// recursive count
echo count($food, COUNT_RECURSIVE); // output 8

// normal count
echo count($food); // output 2
?>

```

260

Một số hàm toán học (tt)

- number abs (mixed \$number)
- float exp (float \$arg)
- number pow (number \$base, number \$exp)
- float sqrt (float \$arg): tính căn bậc 2
- int rand ([int \$min, int \$max])
- int rand()

```
<?php
// Generate a seed
$seed = (float) microtime( ) * 100000000;
// Seed the pseudo-random number generator
srand($seed);
// Generate some random numbers
print rand()."<br>"; // between 0 and getmaxrand()
print rand(1, 6)."<br>"; // between 1 and 6 (inclusive)
?>
```

Ví dụ

261

Một số hàm toán học (tt)

- float round (float \$val [, int \$precision])
 - VD: numberformat.php
- float ceil (float \$value)

```
<?php
echo ceil(4.3); // 5
echo ceil(9.999); // 10
echo ceil(-3.14); // -3
?>
```

- float floor (float \$value)

```
<?php
echo floor(4.3); // 4
echo floor(9.999); // 9
echo floor(-3.14); // -4
?>
```

262

Các phép toán cơ bản trong PHP

Loại	Toán tử
	<code>new .</code>
	<code>. [] ()</code>
Toán học	<code>+ - * / % ++ --</code>
So sánh	<code>< > <= >= != === !==</code>
Luận lý	<code>&& ?: ,</code>
Xử lý bit	<code>! ~ << >> >>></code> AND OR XOR
Gán	<code>= += -= *= /= \%=</code> <code>>>= <<= &= = ^= .=</code>
Ép kiểu	(kiểu dữ liệu)

263

Cấu trúc lựa chọn

- **Lệnh if**

- Cú pháp:

```
if (condition) {
    code to be executed if condition is true;
}
```

- Ví dụ:

```
<?php
$t = date("H");
if ($t < "20") {
    echo "Have a good day!";
}
?>
```

264

Câu trúc lựa chọn

- Lệnh if ...else
 - Cú pháp:

```
if (condition) {
    code to be executed if condition is true;
} else {
    code to be executed if condition is false;
}
```

- Ví dụ:

```
$t = date("H");
if ($t < "20") {
    echo "Have a good day!";
} else {
    echo "Have a good night!";
}
```

265

Câu trúc lựa chọn

- Lệnh if...elseif....else
 - Cú pháp:

```
if (condition) {
    code to be executed if this condition is true;
} elseif (condition) {
    code to be executed if this condition is true;
} else {
    code to be executed if all conditions are false;
}
```

- Ví dụ:

```
$t = date("H");
if ($t < "10") echo "Have a good morning!";
elseif ($t < "20") echo "Have a good day!";
else echo "Have a good night!";
}
```

266

Câu trúc lựa chọn (tt)

- Biểu thức điều kiện:

– Cú pháp: (điềukiện)?

<kếtqua khi điều kiện đúng>:

<kết quả khi điều kiện sai>

– Dùng để thay thế cho câu trúc điều khiển if...else với một câu lệnh bên trong

```
<?php
$a = $_POST["a"];
$b = $_POST["b"];
if($a>$b)
    $so_lon = $a;
else
    $so_lon = $b;
?>
```

```
<?php
$a = $_POST["a"];
$b = $_POST["b"];
$so_lon = ($a>$b)?$a:$b;
?>
```

267

Câu trúc lựa chọn (tt)

- switch

– Cú pháp

```
switch (n) {
    case label1:
        code to be executed if n=label1;
        break;
    case label2:
        code to be executed if n=label2;
        break;
    case label3:
        code to be executed if n=label3;
        break;
    ...
    default:
        code to be executed if n is different from all labels;
}
```

268

Câu trúc lựa chọn (tt)

- switch
 - Ví dụ

```
$favcolor = "red";
switch ($favcolor) {
    case "red":
        echo "Your favorite color is red!";
        break;
    case "blue":
        echo "Your favorite color is blue!";
        break;
    case "green":
        echo "Your favorite color is green!";
        break;
    default:
        echo "Your favorite color is neither red, blue, nor green";
}
```

269

Câu trúc lặp

- Lệnh for
 - Cú pháp:

```
for (init counter; test counter; increment counter) {
    code to be executed;
}
```

- Ví dụ:

```
for ($x = 0; $x <= 10; $x++) {
    echo "The number is: $x <br>";
}
```

270

Cấu trúc lặp (tt)

- Lệnh foreach: thường được dùng để duyệt tập hợp

– Cú pháp:

```
foreach ($array as $value) {
    code to be executed;
}
```

– Ví dụ:

```
$colors = array("red", "green", "blue", "yellow");
foreach ($colors as $value) {
    echo "$value <br>";
}
```

271

Cấu trúc lặp (tt)

- Lệnh while:

– Cú pháp:

```
while (condition is true) {
    code to be executed;
}
```

– Ví dụ:

```
$x = 1;
while($x <= 5) {
    echo "The number is: $x <br>";
    $x++;
}
```

272

Cấu trúc lặp (tt)

- Lệnh do ... while:

– Cú pháp:

```
do {
    code to be executed;
} while (condition is true);
```

– Ví dụ:

```
$x = 1;
do {
    echo "The number is: $x <br>";
    $x++;
} while ($x <= 5);
```

273

break và continue

- break:

– thoát khỏi cấu trúc switch
 – thoát khỏi vòng lặp

```
<?php // kiểm tra số nguyên tố
$so = $_POST["so"];
$kq = true;
for($i=2; $i <= sqrt($so); $i++) {
    if($so%$i==0)
    {
        $kq = false;
        break;
    }
}
?>
```

274

break và continue

- continue:

- Các lệnh bên dưới continue không thực hiện
- Quay về đầu vòng lặp để kiểm tra biểu thức điều kiện và thực hiện vòng lặp tiếp theo

```
<?php // tính tổng các số lẻ từ 1 đến 10
$tong = 0;
for($i=1;$i<=10;$i++)
{
    if ($i%2==0) continue;
    $tong = $tong + $i;
}
echo $tong; →25
?>
```

275

Hàm

- Khai báo hàm

- Hàm không tham số

```
function functionName() {
    code to be executed;
}
```

- Hàm có tham số

```
function familyName($fname) {
    echo "$fname Refsnes.<br>";
}
```

276

Hàm

- Hàm với tham số mặc định

```
function setHeight($minheight = 50) {
    echo "The height is : $minheight <br>";
}
```

- Hàm có trị trả về

```
function sum($x, $y) {
    $z = $x + $y;
    return $z;
}
echo "5 + 10 = " . sum(5, 10) . "<br>";
echo "7 + 13 = " . sum(7, 13) . "<br>";
echo "2 + 4 = " . sum(2, 4);
```

277

Hàm

- Truyền tham chiếu:

```
<?php
function doubleVal(&$var)
{
    $var = $var * 2;
}
$a = 5;
doubleVal ($a);
echo "\$a is: $a";
?>
```

Kết quả:
 \$a = 10

278

Mảng

- Các cách khai báo mảng 1 chiều:

– \$cars = array ("Volvo", "BMW", "Toyota");

– Hoặc

– \$cars[0] = "Volvo";

 \$cars[1] = "BMW";

 \$cars[2] = "Toyota";

– Ví dụ:

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

279

Mảng (tt)

- Lấy số phần tử mảng

– count (\$array [, int \$mode]);

 – \$mode: COUNT_RECURSIVE (1): Đếm tất cả các phần tử con

- Duyệt mảng

```
$cars = array("Volvo", "BMW", "Toyota");
$arrlength = count($cars);
for($x = 0; $x < $arrlength; $x++) {
    echo $cars[$x];
    echo "<br>";
}
```

280

Mảng (tt)

- **Mảng kết hợp**

- Là loại mảng mà mỗi phần tử được truy xuất theo khóa (key)

- Tạo mảng
 - \$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
 - Hoặc
 - \$age['Peter'] = "35";
\$age['Ben'] = "37";
\$age['Joe'] = "43";

- Ví dụ

```
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
echo "Peter is " . $age['Peter'] . " years old.";
```

281

Mảng (tt)

- **Mảng kết hợp**

- Duyệt mảng

```
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
foreach($age as $x => $x_value)
{
    echo "Key=" . $x . ", Value=" . $x_value;
    echo "<br>";
}
```

282

Mảng

- Một số hàm thao tác trên mảng
 - var_dump (array) : xuất nội dung thông tin mảng
 - is_array(array) : kiểm tra mảng
 - reset(array) : khởi tạo lại mảng
 - array_push(array, elements): thêm phần tử cuối mảng
 - array_pop(array): lấy phần tử cuối mảng
 - array_unshift (array, elements) : thêm phần tử đầu mảng
 - array_shift (array) : lấy phần tử đầu mảng
 - array_merge (array, array) : trộn 2 mảng
 - shuffle (array) : sắp xếp mảng ngẫu nhiên
 - array_reverse : đảo mảng

283

Mảng (tt)

- Sắp xếp mảng
 - sort() – Sắp xếp tăng dần
 - rsort() - Sắp xếp giảm dần
 - asort() - Sắp xếp tăng dần theo giá trị (mảng kết hợp)
 - ksort() - Sắp xếp tăng dần theo khóa (mảng kết hợp)
 - arsort() - Sắp xếp giảm dần theo giá trị (mảng kết hợp)
 - krsort() - Sắp xếp giảm dần theo khóa(mảng kết hợp)

284

Mảng (tt)

- Truy xuất phần tử mảng
 - Truy xuất phần tử tại vị trí con trỏ
 - current (array &\$array)
 - Truy xuất phần tử đứng sau phần tử hiện tại
 - next (array &\$array)
 - Truy xuất phần tử trước phần tử hiện tại
 - prev (array &\$array)
 - Truy xuất phần tử cuối mảng
 - end (array &\$array)

285

Mảng (tt)

```
<?php
$transport = array('foot', 'bike', 'car', 'plane');
$mode = current($transport); // $mode = 'foot';
$mode = next($transport); // $mode = 'bike';
$mode = next($transport); // $mode = 'car';
$mode = prev($transport); // $mode = 'bike';
$mode = end($transport); // $mode = 'plane';
?>
```

286

Mảng (tt)

- Reset lại mảng

 - reset (array &\$array)

```
<?php
$array = array('step one', 'step two', 'step three', 'step four');
// by default, the pointer is on the first element
echo current($array) . "<br />\n"; // "step one"
// skip two steps
next($array);
next($array);
echo current($array) . "<br />\n"; // "step three"
// reset pointer, start again on step one
reset($array);
echo current($array) . "<br />\n"; // "step one"
?>
```

287

4.2 Form và các control

- Được dùng để nhận dữ liệu từ phía người dùng
- Giúp gởi yêu cầu của người dùng đến trang xử lý trong ứng dụng web
- Thẻ <form> dùng để chứa các thành phần khác của form
- Những thành phần nhập liệu được gọi là Form Field
 - text field
 - password field
 - multiple-line text field

288

Form

- Thẻ form: chứa các thành phần khác.
 - <Form name="..." action="..." method="...">
 - <!-- các thành phần của Form -->
 - </Form>
 - name : tên FORM
 - action :
 - action="URL": trang web nhận xử lý dữ liệu từ form này khi có sự kiện click của button SUBMIT.
 - action= "script code" mà form sử dụng
 - method: Xác định phương thức chuyển dữ liệu (POST, GET (mặc định))

289

Form

```
<html>
<body>
<form Name="Dangnhap" Action="/admin/xlDangnhap.php" Method="Post">
.....
</form>
</body>
</html>
```

```
<form action="<?php echo ($PHP_SELF) ?>" method="Post">
...
</form>
```

290

Các thành phần của form

- | | |
|------------------------|----------------------------|
| • Text field | Button, Generalized Button |
| • Password field | • Multiple-line text field |
| • Hidden Text field | • Label |
| • Check box | • Dropdown list |
| • Radio button | • GroupBox |
| • File Form Control | • Bảng |
| • Submit Button, Reset | |

291

4.3 Xử lý trang web

- Truyền dữ liệu giữa các trang
- Tự động chuyển hướng trang web
- Cơ chế Truyền và Nhận dữ liệu giữa các trang web
- Truyền/Nhận qua phương thức GET
- Truyền/Nhận qua phương thức POST

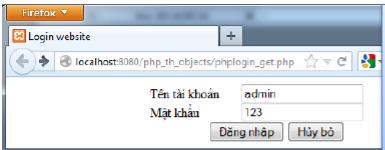
292

Truyền dữ liệu giữa các trang

- Sử dụng các đối tượng:
 - `$_GET`
 - `$_POST`
 - `$_REQUEST`
 - `$_COOKIE`
 - `$_SESSION`

293

Truyền dữ liệu giữa các trang (tt)

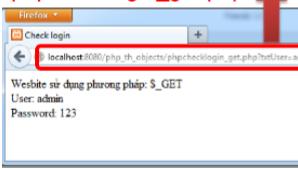


`phplogin_get.php`

```
<form name="frmLogin" method="get"
action="phpchecklogin_get.php">
...
</form>
```

Ví dụ sử dụng `$_GET`

http://localhost:8080/php_th_objects/phpchecklogin_get.php?txtUser=admin&txtPassword=123&bSubmit=%C4%90%C4%83ng+nh%E1%BA%ADp

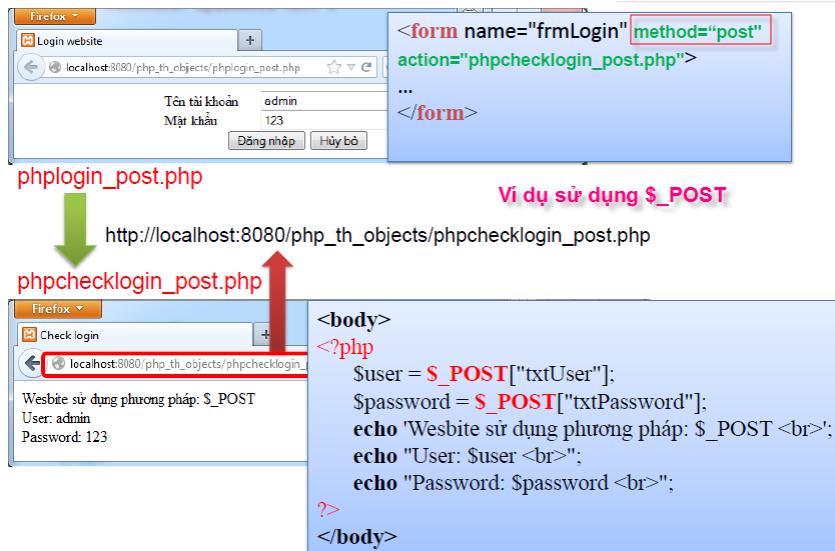


```
<body>
<?php
$user = $_GET["txtUser"];
$password = $_GET["txtPassword"];
echo 'Website sử dụng phương pháp: $_GET';
echo "User: $user <br>";
echo "Password: $password <br>";
?>
</body>
```

294

Truyền dữ liệu giữa các trang (tt)

Ví dụ sử dụng \$_POST



The diagram illustrates the process of transmitting data between two PHP pages using the `$_POST` superglobal array. It shows a browser window for `phplogin_post.php` containing a login form with fields for 'Tên tài khoản' (admin) and 'Mật khẩu' (123). An arrow points from this form to a second browser window for `phpchecklogin_post.php`. The code for `phpchecklogin_post.php` is displayed in a box:

```
<form name="frmLogin" method="post"
action="phpchecklogin_post.php">
...
</form>
```

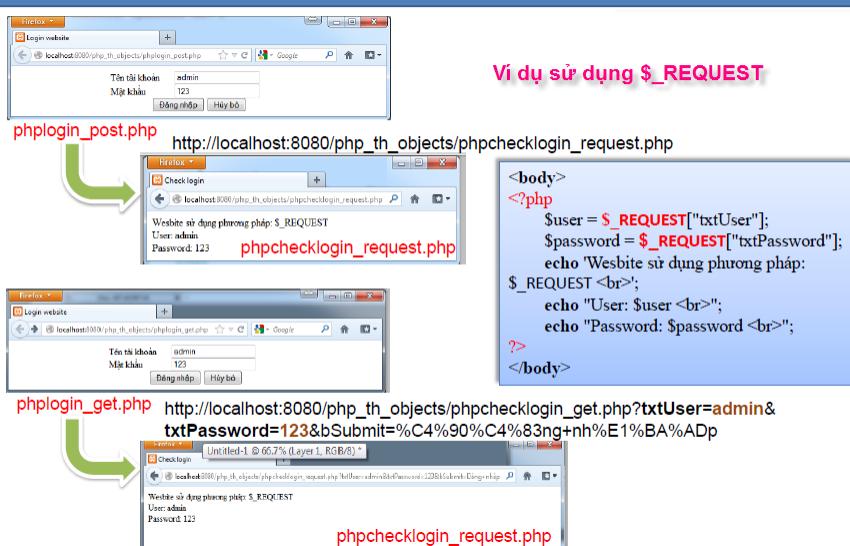
Below the browser windows, the URL `http://localhost:8080/php_th_objects/phpchecklogin_post.php` is shown. The code for `phpchecklogin_post.php` is also provided:

```
<body>
<?php
$user = $_POST["txtUser"];
$password = $_POST["txtPassword"];
echo 'Website sử dụng phương pháp: $_POST';
echo "User: $user<br>";
echo "Password: $password<br>";
?>
</body>
```

295

Truyền dữ liệu giữa các trang (tt)

Ví dụ sử dụng \$_REQUEST



The diagram illustrates the process of transmitting data between three PHP pages using the `$_REQUEST` superglobal array. It shows a browser window for `phplogin_post.php` containing a login form with fields for 'Tên tài khoản' (admin) and 'Mật khẩu' (123). A green arrow points from this form to a second browser window for `phpchecklogin_request.php`. The code for `phpchecklogin_request.php` is displayed in a box:

```
<body>
<?php
$user = $_REQUEST["txtUser"];
$password = $_REQUEST["txtPassword"];
echo 'Website sử dụng phương pháp: $_REQUEST';
$_REQUEST <br>;
echo "User: $user<br>";
echo "Password: $password<br>";
?>
</body>
```

Below the browser windows, the URL `http://localhost:8080/php_th_objects/phpchecklogin_request.php` is shown. A green arrow points from the `phpchecklogin_request.php` window to a third browser window for `phplogin_get.php`. The code for `phplogin_get.php` is displayed in a box:

```
Untitled-1 ① 65% (Layer 1, RGB/8)
<?php
$txtUser = $_GET['txtUser'];
$txtPassword = $_GET['txtPassword'];
$bSubmit = $_GET['bSubmit'];
if($bSubmit == "Đăng nhập") {
    echo "Website sử dụng phương pháp: $_REQUEST";
    echo "User: $txtUser";
    echo "Password: $txtPassword";
}
?>
```

The final URL shown is `http://localhost:8080/php_th_objects/phpchecklogin_get.php?txtUser=admin&txtPassword=123&bSubmit=%C4%90%C4%83ng+nh%E1%BA%ADp`.

296

Đối tượng \$_GET

- Dữ liệu gửi từ trình duyệt lên server qua phương thức GET là phần dữ liệu được nhập trực tiếp sau địa chỉ URL, được phân biệt với tên file script bằng dấu ?
 – Ví dụ: khi ta gõ vào trình duyệt địa chỉ URL sau:
http://www.phpvn.org/topic.php?TOPIC_ID=161 Khi đó, trình duyệt sẽ gửi theo địa chỉ trên một cặp: biến = giá trị, trong đó biến có tên là TOPIC_ID và giá trị là 161 (TOPIC_ID=161).

297

Đối tượng \$_GET

- Dữ liệu gửi từ trình duyệt lên server có thể đưa lên nhiều cặp biến=gia_trị bằng cách phân cách chúng bởi dấu &:
 – Ví dụ:
http://www.phpvn.org/index.php?method=Reply&TOPIC_ID=161&FORUM_ID=20 Với địa chỉ URL trên, trình duyệt gửi lên 3 cặp biến=gia_trị theo phương thức GET: method=Reply, TOPIC_ID=161 và FORUM_ID=20.

298

Đối tượng \$_GET

- Khi trình duyệt gửi các thông tin này lên máy chủ, PHP sẽ tự động sinh ra một mảng có tên là `$_GET[]` để chứa tất cả các cặp biến và giá trị đó.
 - Chỉ số của mảng chính là một chuỗi mang tên của tên biến
 - Giá trị của phần tử tại chỉ số đó chính là giá trị của biến do trình duyệt gửi lên.
 - Ví dụ:
http://www.phpvn.org/post.php?method=Reply&TOPIC_ID=161&FORUM_ID=20
 PHP sẽ tự động sinh ra một mảng `$_GET` có nội dung sau:
 - `$_GET["method"] = "Reply"` // Ứng với cặp method=Reply
 - `$_GET["TOPIC_ID"] = 161` // Ứng với cặp TOPIC_ID=161
 - `$_GET["FORUM_ID"] = 20` // Ứng với cặp FORUM_ID=20

299

Đối tượng \$_GET

- Ví dụ: sử dụng đối tượng `$_GET`
 - Trang giao diện: `giaodien.php`

```

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Example GET</title>
</head>
<body>
<a href='chitietsach.php?Ma=N001'> Chi tiết </a>
</body>
</html>

```

300

Đối tượng \$_GET

- Ví dụ: sử dụng đối tượng \$_GET
 - Trang xử lý PHP: chitietsach.php

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Example GET</title>
</head>
<body>
<?php
echo "Mã sách lấy được là".$_GET["Ma"];
?>
</body>
</html>
```

301

Đối tượng \$_GET

- Ví dụ: sử dụng đối tượng \$_GET
 - Trang xử lý PHP: chitietsach.php (dùng hàm isset để kiểm tra sự tồn tại của biến)

```
<html>
<head><title>Example GET</title></head>
<body>
<?php
if (isset ($_GET["Ma"]))
{
    if ($_GET["Ma"]=="SGK") echo "Bạn chọn mua sách giáo khoa";
    else echo "Mã sách " . $_GET["Ma"]." không phải là sách giáo khoa!"; }
else{ echo "Dữ liệu không hợp lệ !"; }
?>
</body>
</html>
```

302

Đối tượng \$_POST

- Dữ liệu gửi từ trình duyệt lên server qua phương thức POST là phần dữ liệu được lưu trữ trong phần thân Request. Việc truy xuất các phần tử dữ liệu trên server được thực hiện tương tự như đối tượng \$_GET.
- Ví dụ: khi ta gõ vào trình duyệt địa chỉ URL sau:
 - Truyền theo phương thức GET
 - http://www.phpvn.org/topic.php?TOPIC_ID=161
 - Truyền theo phương thức POST
 - http://www.phpvn.org/topic.php
 - Khi đó, trình duyệt cũng sẽ gửi lên server một cặp: **biến = giá trị** (lưu trong phần thân Request), trong đó biến có tên là TOPIC_ID và giá trị là 161 (**TOPIC_ID=161**).

303

Đối tượng \$_POST

```

<body>
<form method="POST" action="">
<p>User Name:<input type="text" name="txtUser" size="20"> </p>
<p>Password:<input type="password" name="txtPass" size="20"></p>
<p>Sex: <Select name ="selSex">
<option value =1>Male </option>
<option value =0>Female </option>
</select></p>
<input type="submit" name="bSubmit" value="Submit" >
</form>
<?php
if (isset($_POST["bSubmit"]))&& ($_POST["bSubmit"]=="Submit")){
echo "<script language='javascript'>window.open('http://php.net')</script>";
//hoặc dùng hàm: header('Location: http://www.php.net/');
}?
</body>

```

304

Đối tượng \$_POST

Code PHP:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>Sample POST</title></head>
<body>
<form method="POST" action="">
<p>User Name:<input type="text" name="txtUser">
<p>Password:<input type="password" name="txtPass">
<p>Sex: <Select name ="selSex">
    <option value =1>Male </option>
    <option value =0>Female </option>
</select></p>
<input type="submit" name="bSubmit" value="Submit">
</form>
<?php
if (isset($_POST["bSubmit"]))&& ($_POST["bSubmit"]=="Submit")){
    echo "<script language='javascript'>window.open('http://php.net')</script>";
    //hoặc dùng hàm: header('Location: http://www.php.net/');
} else
    echo "không";
?>
</body>s
</html>
```

Firefox - Sample POST
 localhost:8080/php_th_objects/php_post_SlideEd02L.php

User Name: admin
 Password: ***
 Sex: Male
 Submit
 không

Khi người dùng nhập User Name là: admin, Password là: 123 và chọn Sex là Male, khi đó, mảng **\$_POST** sẽ có các phần tử sau:
\$_POST["txtUser"] = admin
\$_POST["txtPass"] = 123
\$_POST["selSex"] = 1

Dùng hàm **header(URL)** để thực hiện chuyển trang

305

Đối tượng \$_COOKIE

- Là 1 đoạn dữ liệu được truyền đến browser từ server, đoạn dữ liệu này sẽ được browser lưu trữ (trong memory hoặc trên đĩa) và sẽ gửi ngược lên lại server mỗi khi browser tải 1 trang web từ server.
- Những thông tin được lưu trữ trong cookie phụ thuộc vào website trên server.
- Cookie được tạo ra bởi website và gửi tới browser, do vậy 2 website khác nhau (cho dù cùng host trên 1 server) sẽ có 2 cookie khác nhau gửi tới browser.
- Mỗi browser quản lý và lưu trữ cookie theo cách riêng của mình, cho nên 2 browser cùng truy cập vào 1 website sẽ nhận được 2 cookie khác nhau.

306

Đối tượng \$_COOKIE

- Sử dụng Cookie trong PHP:
 - Đặt (set) cookie:
 - `$_COOKIE[tên_cookie] = giá_trị;`
 - Đọc (get) lại giá trị của cookie:
 - `$_COOKIE[tên_cookie]`

307

Đối tượng \$_COOKIE

```

<?php
$t="1111";
setcookie("a",$t);
?>
<html>
<body>
Giá trị gởi lên cookies:
<?php echo $t; ?>
<a href ="b.php"> qua trang b </a>
</body>
</html>

```

```

<html>
<body>
<a href="a.php"> qua trang a</a>
<?php
if (isset($_COOKIE['a'])) {
    echo "gia tri lay duoc
".$_COOKIE['a'];
}
else
    echo "khong lay duoc";
?>
</body>
</html>

```

308

Đối tượng \$_SESSION

- Là đoạn dữ liệu được lưu trên server, khi browser có yêu cầu lấy dữ liệu từ session thì server cung cấp.
- Website sẽ quyết định khi nào session bắt đầu và kết thúc.
- Mỗi session sẽ có một định danh (ID).

309

Đối tượng \$_SESSION

- Các hàm liên quan đến Session:
 - session_start(): khởi tạo session.
 - session_register(tên biến): đăng ký biến session
 - \$_SESSION[tên_session] = giá_trị: đặt giá trị cho session
 - \$_SESSION[tên_session]: đọc giá trị từ session
 - session_destroy(): hủy tất cả các dữ liệu trong session
 - session_unset(): hủy tất các biến trong session
 - session_unregister(tên biến) hủy 1 biến trong session

310

Đối tượng \$_SESSION

```

<?php
session_start();
$t=time(); $_SESSION['username'] = 'guest';
$_SESSION['password'] = $t;
?>
<html>
<body>
Giá trị của session đã được gán:<br> username =
guest <br> time = <?php echo $t; ?>
<br> Click
<a href="b.php">vào day</a> de kiem tra.
</body>
</html>

```

```

<?php
session_start();
?>
<html>
<body>
Giá trị session lấy được
<a href="a.php">file a.php</a>:<br>
username =
<?php echo $_SESSION['username']; ?>
<br> time =
<?php echo $_SESSION['password']; ?>
</body>
</html>

```

311

Cơ chế Truyền và Nhận dữ liệu giữa các trang web

- Trang web nhập dữ liệu:
 - Sử dụng đối tượng `<form>`
 - Nhập liệu thông qua các formfield
 - Thực hiện việc truyền dữ liệu thông qua Submit
- Trang web nhận dữ liệu (URL): Sử dụng các biến toàn cục của PHP
 - `$_POST["FieldName"]`
 - `$_GET["FieldName"]`
 - `$_REQUEST["FieldName"]`

```

<FORM ACTION="URL" METHOD="GET/POST">
...
<input type="submit" value="Xử lý">
</FORM>

```

312

Trang web nhập dữ liệu

Ví dụ: Trang booksearch.php

```
<html>
<body>
<h1>Tìm sách</h1>
<form action="bookresult.php" Method="GET" >
Từ khóa : <input type="text" name="txtKey"/>
<input type="submit" value="Tìm"/>
</form>
</body>
</html>
```

Tìm sách

Từ khóa :

313

Trang web nhập dữ liệu

Trang bookresult.php

```
<html>
<body>
<?php
    $key = $_REQUEST["txtKey"];
?>
<h1>Tìm sách</h1>
Từ khóa tìm sách là : <?php echo $key; ?>
</body>
</html>
```

Tìm sách

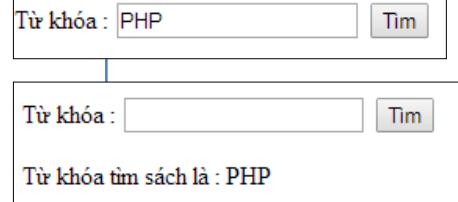
Từ khóa tìm sách là : C#

314

Trang web nhập dữ liệu

Trang bookprocess.php

```
<html>
<body>
<form action="bookprocess.php" Method="GET" >
    Từ khóa : <input type="text" name="txtKey"/>
    <input type="submit" value="Tìm"/>
</form>
<?php
    $key = $_REQUEST["txtKey"];
    if (isset($key)) {
        print "Từ khóa tìm sách là : $key";
    }
?>
</body>
</html>
```



Tùy chọn :

Tùy chọn :

Tùy chọn tìm sách là : PHP

315

Truyền/Nhận qua phương thức GET

- Tham số truyền đi qua địa chỉ URL
- Nhận dữ liệu thông qua biến toàn cục của PHP
 - \$_GET["FieldName"]
 - \$_REQUEST["FieldName"]
- Ưu điểm
 - Người dùng có thể bookmark địa chỉ URL
 - Người dùng có thể giả lập phương thức GET để truyền dữ liệu mà không cần thông qua FORM
- Khuyết điểm
 - Không thích hợp để truyền dữ liệu có tính bảo mật
 - Dung lượng dữ liệu truyền đi có giới hạn
 - URL submit bằng phương thức GET được lưu lại trên server

316

Truyền/Nhận qua phương thức POST

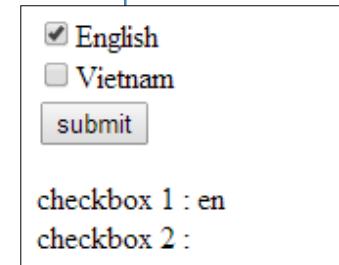
- Tham số truyền đi được ẩn bên trong FORM
- Nhận dữ liệu thông qua biến toàn cục của PHP
 - `$_POST["FieldName"]`
 - `$_REQUEST["FieldName"]`
- Ưu điểm
 - Bảo mật hơn phương thức GET
 - Không giới hạn dung lượng dữ liệu truyền đi
- Khuyết điểm
 - Kết quả trang web trả về không thể bookmark
 - Có thể gây ra lỗi nếu người dùng muốn quay lại trang kết quả (nhấn nút Back hoặc Refresh) do bị expired
 - Dữ liệu có thể không truyền đi được do vấn đề về security

317

Một số ví dụ

- Truyền/nhận dữ liệu từ checkbox: trang checkbox.php

```
<html><body>
<form method="get" action="checkbox.php">
<input type="checkbox" name="chk1" value="en">English
<br>
<input type="checkbox" name="chk2" value="vn">Vietnam<br>
<input type="submit" value="submit"><br>
</form>
<?php
echo "checkbox 1 : " . $_REQUEST["chk1"] . "<br>";
echo "checkbox 2 : " . $_REQUEST["chk2"];
?>
</body></html>
```



checkbox 1 : en
checkbox 2 :

318

Một số ví dụ

- Truyền/nhận dữ liệu từ Radio button: trang radio.php

```

<html><body>
<form action="radio.php" method="GET">
<input type=RADIO" NAME="rdGT"
VALUE="Nam">Nam<br>
<input type=RADIO" NAME="rdGT" VALUE="Nu">Nữ<br>
<input type="SUBMIT" VALUE="Submit">
</form>
<? php
    if (isset($_GET['rdGT']))
        echo "Gioi tinh : " . $_GET[ 'rdGT'];
?>
</body></html>

```

Nam
 Nữ

 Gioi tinh : Nam

319

Một số ví dụ

- Truyền/nhận dữ liệu từ DropDownList: trang dropdownlist.php

```

<form method="POST" action="dropdownlist.php">
<select name="chon">
    <option value="Nhạc"<?php if($_POST["chon"]=="Nhạc") echo "selected"; ?>
        >Âm nhạc</option>
    <option value="Thể thao"<?php if($_POST["chon"]=="Thể thao") echo "selected"; ?>
        >Thể thao</option>
    <option <?php if($_POST["chon"]=="Hội hoà") echo "selected"; ?>
        >Hội hoà</option>
</select>
<input type="submit" name="submit" value="Chọn"/>
</form>
Sở thích của bạn là: &nbsp;
<?php
    if (isset($_POST["chon"])) echo $_POST["chon"] . "<br/>";
?>
...

```

Âm nhạc ▾ Chọn
 Sở thích của bạn là: Nhạc

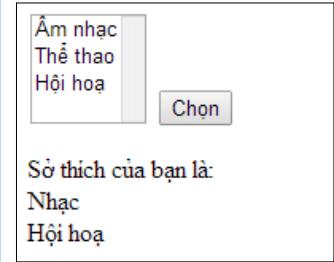
Hội hoà ▾ Chọn
 Sở thích của bạn là: Hội hoà

320

Một số ví dụ

- Truyền/nhận dữ liệu từ ListBox: trang dropdownlist.php

```
<form method="POST" action="listbox.php">
<select name="chon[]" multiple>
    <option value="Nhạc">Âm nhạc</option>
    <option value="Thể thao" >Thể thao</option>
    <option>Hội hoạ</option>
</select>
<input type="submit" name="submit" value="Chọn"/>
</form>
Sở thích của bạn là: <br/>
<?php
    if(isset($_POST["chon"]))
        foreach ($_POST["chon"] as $item){ echo $item. "<br/>";}
?>
```



Âm nhạc
Thể thao
Hội hoạ

Chọn

Sở thích của bạn là:
Nhạc
Hội hoạ

321

4.4 PHP và MySQL

- Làm việc với cơ sở dữ liệu MySql
- Sử dụng MySqli (MySQL Improved Extension)

322

4.4.1 MySQL

- Tạo kết nối
- Chọn CSDL
- Truy vấn dữ liệu
- Đóng kết nối
- Thông báo lỗi

323

Tạo kết nối

- Cú pháp:
 - `mysql_connect(servername,username,password);`
 - `servername`: tham số tùy chọn, xác định server cần phải kết nối tới.
Giá trị mặc định là “localhost:3306”
 - `username`: tham số tùy chọn, xác định tên người dùng đăng nhập vào hệ thống. Giá trị mặc định là tên của người dùng làm chủ tiến trình của server.
 - `password`: tham số tùy chọn, xác định mật khẩu của người dùng.
Giá trị mặc định là “”

324

Tạo kết nối

- Ví dụ: tạo kết nối

```
<?php
$db=mysql_connect("localhost","root","");
if(!$db)
{
    echo "Không thể kết nối CSDL";
    exit;
}
?>
```

325

Chọn CSDL

- Cú pháp:
 - mysql_select_db(database,connection);
 - database: tham số bắt buộc, xác định tên của CSDL cần làm việc.
 - connection: tham số tùy chọn, xác định kết nối. Nếu không xác định thì kết nối cuối cùng được mở bởi hàm mysql_connect() hoặc hàm mysql_pconnect() sẽ được sử dụng.
- Kết quả trả về của hàm này là TRUE nếu chọn CSDL thành công, ngược lại kết quả sẽ bằng FALSE

326

Chọn CSDL

- Ví dụ: chọn CSDL là ql_ban_sua

```
$db_selected = mysql_select_db("ql_ban_sua", $db);
// kiểm tra CSDL
if (!$db_selected)
{
    die ("Không thể sử dụng CSDL : " . mysql_error());
}
```

327

Truy vấn dữ liệu

- Cú pháp:
 - mysql_query(query,connection);
 - query: tham số bắt buộc, là câu lệnh truy vấn được gửi đi.
 - connection: tham số tùy chọn, xác định kết nối. Nếu không xác định thì kết nối cuối cùng được mở bởi hàm mysql_connect() hoặc hàm mysql_pconnect() sẽ được sử dụng.
 - mysql_query() sẽ trả về kết quả của câu lệnh truy vấn nếu thực hiện thành công, ngược lại sẽ trả về FALSE

328

Truy vấn dữ liệu

- Ví dụ: thực hiện truy vấn dữ liệu có trong bảng hang_sua trong CSDL ql_ban_sua
 - \$sql= "SELECT * FROM hang_sua";
 - \$result = mysql_query(\$sql);

329

Đóng kết nối

- Cú pháp:
 - mysql_close(connection);
 - connection: tham số tùy chọn, xác định kết nối. Nếu không xác định thì kết nối cuối cùng được mở bởi hàm mysql_connect() hoặc hàm mysql_pconnect() sẽ được sử dụng.
- Kết quả trả về là TRUE nếu đóng kết nối thành công, ngược lại sẽ trả về FALSE nếu thất bại.
- Ví dụ: đóng kết nối đã mở
 - mysql_close(\$db);

330

Thông báo lỗi

- Trong quá trình làm việc với CSDL lỗi có thể phát sinh
→ cần thông báo những lỗi phát sinh này bằng cách sử dụng hàm mysql_error()
- Cú pháp: mysql_error(connection);
- Kết quả trả về là câu thông báo lỗi nếu có lỗi phát sinh, ngược lại kết quả trả về sẽ là một chuỗi rỗng "".
- Kết hợp hàm mysql_error() với hàm die() hoặc hàm exit()
để vừa thông báo lỗi vừa kết thúc công việc

331

Thông báo lỗi

- Ví dụ: thông báo lỗi nếu không thể tạo kết nối

```
$db = mysql_connect("localhost","root","");
if (!$db)
{
    die('Không thể kết nối: ' . mysql_error());
}
```

332

Đếm số lượng mẫu tin

- Cú pháp:
 - mysql_num_rows(data);
 - data: Xác định con trỏ dữ liệu, là kết quả trả về của hàm mysql_query().
- Hàm này có kết quả trả về là số lượng mẫu tin nếu thành công, ngược lại kết quả trả về sẽ là FALSE nếu thất bại.

333

Đếm số lượng mẫu tin

- Ví dụ: đếm số lượng mẫu tin có trong bảng hang_sua

```
$sql= "SELECT * FROM hang_sua";
$result = mysql_query($sql);
echo mysql_num_rows($result); → 7
```

334

Hiển thị dữ liệu

- Duyệt dữ liệu: có các cách sau
 - Duyệt dữ liệu theo dạng mỗi mẫu tin là một dòng
 - Duyệt theo dạng mỗi mẫu tin là một mảng
 - Duyệt theo dạng mỗi mẫu tin là một đối tượng

335

Hiển thị dữ liệu

- Duyệt dữ liệu theo dạng mỗi mẫu tin là một dòng:
`mysql_fetch_row`
 - Cú pháp: `mysql_fetch_row(data)`
 - data: Xác định con trỏ dữ liệu, là kết quả trả về của hàm `mysql_query()`
 - Kết quả trả về một mảng chứa giá trị của một dòng dữ liệu với mỗi phần tử là nội dung của một cột → truy cập bằng cách gọi từng phần tử của mảng `$row[0], $row[1], $row[2], ...`

336

Hiển thị dữ liệu

- `mysql_fetch_row (tt)`

```
mysql_select_db("ql_ban_sua");
$result = mysql_query("SELECT * FROM KHACH_HANG");
if (mysql_num_rows($result) <> 0) {
    print_r (mysql_fetch_row($result));
}
mysql_close($db);
```

Kết quả: Array ([0] => kh001
 [1] => Nguyễn Văn A
 [2] => 1
 [3] => Quận 1
 [4] => 12345678
 [5] => nva@gmail.com)

337

Hiển thị dữ liệu

- `mysql_fetch_row (tt)`

- Duyệt tất cả các mẫu tin: kết hợp cấu trúc lặp while và hàm `mysql_fetch_row()` đặt trong cấu trúc lặp while.
- Ví dụ: duyệt & in tất cả các mẫu tin có trong bảng khách hàng

```
if (mysql_num_rows($result) > 0) {
    while($row = mysql_fetch_row($result))
    {
        print_r($row);
    }
}
```

338

Hiển thị dữ liệu

- Duyệt dữ liệu theo dạng mỗi mẫu tin là một mảng:
`mysql_fetch_array`
 - Cú pháp: `mysql_fetch_array(data)`
 - data: Xác định con trỏ dữ liệu, là kết quả trả về của hàm `mysql_query()`
 - Kết quả trả về là một mảng chứa giá trị của một dòng dữ liệu với mỗi phần tử là nội dung của một cột → truy cập bằng cách gọi từng phần tử của mảng: `$row["tên cột 1"]`, `$row["tên cột 2"]`, `$row["tên cột 3"]`, ...

339

Hiển thị dữ liệu

- `mysql_fetch_array (tt)`

```
if (mysql_num_rows($result)>>0) {
    print_r(mysql_fetch_array($result));
}
```

Kết quả: Array ([Ma_khach_hang] => kh001 [Ten_khach_hang] => Nguyễn Văn A [Phai] => 1 [Dia_chi] => Quận 1 [Dien_thoai] => 12345678 [Email] => nva@gmail.com)

340

Hiển thị dữ liệu

- **mysql_fetch_array (tt)**

- Duyệt tất cả các mẫu tin: kết hợp cấu trúc lặp while và hàm mysql_fetch_array() đặt trong cấu trúc lặp while.
- Ví dụ: duyệt & in tất cả các mẫu tin có trong bảng khách hàng

```
if (mysql_num_rows($result)>>0) {
    while ($array = mysql_fetch_array($result))
    {
        print_r($array);
    }
}
```

341

Hiển thị dữ liệu

- Duyệt dữ liệu theo dạng mỗi mẫu tin là một đối tượng: mysql_fetch_object
 - Cú pháp: mysql_fetch_object(data)
 - data: Xác định con trỏ dữ liệu, là kết quả trả về của hàm mysql_query()
 - Kết quả trả về là một mẫu tin trong bộ các mẫu tin như là một đối tượng → truy cập bằng cách gọi từng thuộc tính của đối tượng: \$tên_đối_tượng → tên_cột_1, \$tên_đối_tượng → tên_cột_2, ...

342

Hiển thị dữ liệu

- mysql_fetch_object (tt)

```
if (mysql_num_rows($result)<>0)
{
    print_r(mysql_fetch_object($result));
}
```

Kết quả: stdClass Object (

- [Ma_khach_hang] => kh001
- [Ten_khach_hang] => Nguyễn Văn A
- [Phai] => 1
- [Dia_chi] => Quận 1
- [Dien_thoai] => 12345678
- [Email] => nva@gmail.com)

343

Hiển thị dữ liệu

- mysql_fetch_object (tt)

- Duyệt tất cả các mẫu tin: kết hợp cấu trúc lặp while và hàm mysql_fetch_object() đặt trong cấu trúc lặp while.
- Ví dụ: duyệt & in tất cả các mẫu tin có trong bảng khách hàng

```
if (mysql_num_rows($result)<>0) {
    while($object = mysql_fetch_object($result))
    {
        print_r($object);
    }
}
```

344

Thực thi câu lệnh SQL

- Sử dụng hàm mysql_query().
- Hàm này được dùng để gửi một truy vấn (lấy dữ liệu, thêm mới, xoá, cập nhật) tới một kết nối MySQL.
 - Select ...
 - Insert ...
 - Update ...
 - Delete ...

345

4.4.2 MySQL Improved Extension (MySqlI)

- Kết nối CSDL

```
$con = mysqli_connect("localhost","my_user","my_password","my_db");
```

– Hoặc

```
$con = mysqli_connect("localhost","my_user","my_password");
mysqli_select_db($con, "my_db");
```

– Cần kiểm tra kết nối thành công:

```
if (!$con) {
    die("Connection error: " . mysqli_connect_errno());
    //hoặc die("Connection error: " . mysqli_connect_error());
}
```

346

MySQL Improved Extension (MySql)

- Kết nối CSDL

– Ví dụ:

```
$servername = 'localhost';
$username = 'root';
$password = '123456';
$conn = mysqli_connect($servername, $username, $password);
if(!$conn) {
    echo "Connect Failed!". mysqli_connect_error($conn);
}
else{
    echo "Successsful";
}
```

347

MySQL Improved Extension (MySql)

- Thực thi câu lệnh truy vấn dữ liệu

– Cú pháp

mysqli_query (connection, query, [resultmode]);

- connection: biến kết nối
- query: câu lệnh SQL
- resultmode: tùy chọn:
 - MYSQLI_USE_RESULT: dùng khi dữ liệu lớn
 - MYSQLI_STORE_RESULT: mặc định

348

MySQL Improved Extension (MySql)

- Lấy dữ liệu

```
$con=mysqli_connect("localhost","my_user","my_password","my_db");
if (mysqli_connect_errno())
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
$result = mysqli_query($con, "Select * From Products");
if(mysqli_num_rows ($result) > 0) {
    while ($row = mysqli_fetch_assoc ($result))  {
        echo $row['ProductId'].'.'.$row['ProductName']]. "<br />";
    }
}
mysqli_close($con);
```

349

MySQL Improved Extension (MySql)

- Chèn dữ liệu

```
$con=mysqli_connect("localhost","my_user","my_password","my_db");
if (mysqli_connect_errno())
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
$sqlInsert = "INSERT INTO Products VALUES (10, 'Iphone5 Plus')";
$result = mysqli_query($con, $sqlInsert);
mysqli_close($con);
```

350

MySQL Improved Extension (MySql)

- Cập nhật dữ liệu

```
$con=mysqli_connect("localhost","my_user","my_password","my_db");
if (mysqli_connect_errno())
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
$sqlUpdate = "Update Products Set ProductName = 'Iphone7 Plus'
Where ProductId =10";
$result = mysqli_query($con, $sqlUpdate);
mysqli_close($con);
```

351

MySQL Improved Extension (MySql)

- Xóa dữ liệu

```
$con=mysqli_connect("localhost","my_user","my_password","my_db");
if (mysqli_connect_errno())
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
$sqlDelete = "Delete From Products Where ProductId =10";
$result = mysqli_query($con, $sqlDelete);
mysqli_close($con);
```

352