CHƯƠNG 1 : GIỚI THIỆU, CÀI ĐẶT VÀ CẤU HÌNH APACHE, PHP VÀ MYSQL

I_ APACHE HTTP SERVER

1_Giới thiệu Apache HTTP Server

Dự án Apache là một sự cố gắng phát triển phần mềm cộng tác nhắm đến việc tạo ra một HTTP server mạnh mẽ, có hạng thương mại, được đề cao,và mã nguồn thực hiện miễn phí. Dự án được tham gia quản lý bởi một nhóm người tình nguyện trên toàn thế giới sử dụng internet và Web để truyền thông , dựng kế hoạch và phát triển server . Những người tình nguyện này được biết đến như là nhóm Apache. Thêm nữa, hàng trăm người sử dụng đã đóng góp các ý tưởng, mã và các tài liệu cho dự án.

Vào khoảng tháng 2 năm 1995, phần lớn phần mềm server được ưa chuộng trên web là tên miền HTTP daemon công cộng được phát triển bởi Rob McCool tại trung tâm quốc gia của các ứng dụng siêu máy tính, trường đại học Illinois, Urbana-Champaign.

Tuy nhiên, sự phát triển httpd đó đã bị ngưng trệ sau khi Rob rời NCSA vào khoảng giữa năm 1994, và rất nhiều nhà phát triển web đã phát triển phần mở rộng của chính họ và khắc phục lỗi trong sự cần thiết của một sự phân phát chung. Một nhóm nhỏ của các nhà phát triển web này, đã kết hợp với nhau thông qua e-mail là chính, kết hợp cùng với nhau cho mục đích phối hợp những thay đổi của họ (trong hình thức các miếng vá).

Bằng cách dùng httpd 1.3 làm nền, họ đã thêm vào toàn bộ các miếng vá đã được công bố và các tính năng cao cấp khác, thử nghiệm trên chính các server của họ, và cho ra đời phiên bản công khai chính thức đầu tiên (0.6.2) của server Apache trong tháng 4 năm 1995.

Server Apache ban đầu đã là một sự thành công lớn, nhưng họ cho rằng mã ban đầu cần phải được kiểm tra kỹ lưỡng và thiết kế lại. Trong suốt tháng 5 năm và tháng 6 năm 1995, Robert Thau đã thiết kế một kiến trúc server mới(mã được đặt tên là Shambhala) nó bao gồm một cấu trúc module và API cho việc mở rộng được tốt hơn. Nhóm làm việc đã chuyển sang nền server mới này từ tháng sáu và đã thêm các đặc điểm từ phiên bản 0.7.x, đưa đến kết quả trong Apache 0.8.8 (và các anh em của nó) trong tháng tám.

Sau khi phát hành rộng rãi bản thử nghiệm beta, rất nhiều lỗ hổng trong các nền khác nhau đã được tìm thấy, một tập tài liệu mới (của David Robinson), và quá trình thêm rất nhiều các chức năng trong dạng của các module chuẩn của họ, Apache 1.0 đã được phát hành vào ngày1 tháng 12 năm 1995.

Theo đánh giá của Netcraft (**http://www.netcraft.com/survey/**) chỉ ra rằng ngày nay Apache được sử dụng một cách rộng rãi hơn so với tất cả các web server đã được tổng hợp.

2_Sử dụng Apache với Microsoft Windows

2.1_Các yêu cầu:

Apache 1.3 được thiết kế trên Windows NT 4.0 và Windows 2000. Trình cài đặt nhị phân sẽ chỉ làm việc với họ vi xử lý x86,ví dụ như của hãng Intel. Apache cũng có thể chạy trên Windows 95 và 98. Trong mọi trường hợp, TCP/IP networking phải được cài đặt.
Nếu chạy trên NT 4.0 thì phải chắc rằng máy đã được cài đặt Service Pack 3 hoặc Service 6.
Chú ý: "Winsock 2" phải được yêu cầu cho Apache 1.3.7 và các bản sau này.
Nếu chạy trên Windows 95, bản nâmg cấp "Winsock2" phải được cài đặt trước khi Apache chạy. "Winsock2" cho Windows 95 có sẵn tại các địa chỉ:
http:// www.microsoft.com/windows95/downloads. Bản cập nhật Winsock2 phải được cài đặt lại sau khi cài đặt Windows 95 dialup networking.

2.2_Downloading Apache cho Windows

Thông tin về phiên bản mới nhất cho Apache có thể được tìm thấy trên webserver của Apache tại http://www.apache.org/httpd. Tại đây sẽ liệt kê phiên bản phát hành và các bản phát hành thử.
Có thể download bản nhị phân của Apache cho Windows được đặt tên như: apache_1_3_#-win32-with_src.msi nếu muốn nghiên cứu mã nguồn của Apache thì ta có thể tải bản apache_1_3_#-win32-no_src.msi. Các bản này đều đã có đủ Apache runtime. Trước khi cài đặt Apache runtime, trong máy PC phải được cài Microsoft Installer version 1.10. Windows 2000 và Windows ME đều đã được hỗ trợ Microsoft Installer,còn nếu không thì ta phải download trên trang web của Microsoft.

2.3_Cài đặt Apache trên Windows

Chạy file Apache .msi mà ta đã download về. Sẽ có một số lời nhắc như:

- Tên của ta và tên của công ty ta, và trên Windows NT/2000, có thể ta muốn tất cả user truy cập vào Apache như là một dịch vụ, hoặc nếu ta muốn cài đặt để chạy khi ta chọn Start Apache shortcut.
- Tên Server của ta, tên Domain và tài khoản quản trị của ta.
- Thư mục mà ta muốn cài đặt Apache (mặc định là C:\Program Files\Apache Group\Apache mặc dù ta có thể thay đổi điều này tới bất kỳ thư mục nào ta thích)
- kiểu cà đặt. Tuỳ chọn "Complete" sẽ cài đặt tất cả mọi thứ, bao gồm mã nguồn nếu ta download gói -with_src.msi . Chọn cài đặt "Custom" nếu ta chọn không cài đặt tài liệu, hoặc mã nguồn từ gói.

Trong quá trình cài đặt, Apache sẽ cấu hình các file trong thư mục conf cho sự lựa chọn thư mục cài đặt của ta.Tuy nhiên, nếu có bất kỳ một file nào trong thư mục này thì chúng cũng không bị ghi đè. Thay vào đó, bản copy mới sẽ được gán với đuôi .default.

Sau khi cài đặt Apache, ta sẽ phải biên tập file cấu hình trong thư mục conf, điều này là bắt buộc. Các file này sẽ được cấu hình trong quá trình cài đặt để chuẩn bị cho Apache được chạy từ thư mục mà nó đã được cài đặt, với các tài liệu được đáp ứng trong thư mục con htdocs.Có rất nhiều các tuỳ chọn sẽ được thiết lập trước khi ta thực sự bắt đầu sử dụng Apache.Tuy nhiên, để có thể bắt đầu một cách nhanh chóng, các file sẽ được thực hiện như khi đã được cài đặt.Nếu ta muốn rỡ bỏ Apache, các file cấu hình sẽ không bị bỏ đi. Ta phải xoá cây thư mục ("C:\Program Files\Apache Group" là mặc định) nếu ta thấy không cần thiết giữ các file cấu hình và các file web của ta.

2.4_Chạy Apache trên Windows

Có hai cách ta có thể chạy Apache:

- Như là một "service" (chỉ được kiểm tra trên NT/2000 , nhưng bản thực nghiệm cũng có sẵn cho 95/98). Đây là tuỳ chọn tốt nhất nếu ta muốn Apache tự động khởi động khi máy của ta boot, và giữ cho Apache luôn chạy khi ta log-off.
- Từ một console window. Đây là tuỳ chọn tốt nhất cho người sử dụng trên Windows 95/98 .

Các bước thực hiện trước khi bắt đầu Apache như là một dịch vụ Windows

Để chạy Apache từ một cửa sổ , chọn thuỳ chọn "Start Apache as console app" từ menu Start menu (trong Apache 1.3.4 và các bản sớm hơn, tuỳ chọn này được gọi là "Apache Server"). Điều này sẽ mở ra một cửa sổ console và bắt đầu Apache chạy trong đó. Màn hình này vẫn sẽ còn đó cho đến khi ta dừng Apache. Để dừng Apache khi nó đang chạy, có thể chọn biểu tượng"Shutdown Apache console app" từ menu Start(Điều này khônng có trong Apache 1.3.4 hoặc các bản sớm hơn, hoặc có thể dùng các lệnh điều khiển Apache trong màn hình console.
Trong các phiên bản Apache 1.3.13 và trước đó, ta có thể gõ Ctrl+C hoặc Ctrl+Break để tắt màn hình Apache console.Và trên Windows NT/2000 với phiên bản 1.3.13, Apache sẽ dừng nếu ta chọn 'Close' từ menu hệ thống (bấm vào biểu tượng trên góc trên trái của màn hình console) hoặc bấm vào nút (X) trên góc phải màn hình console.

Thử Apache cho Windows

Nếu gặp trục trặc khi bắt đầu khởi động Apache, hãy theo các bước dưới để cô lập vấn đề. Điều này được ứng dụng nếu ta bắt đầu Apache với shortcut "Start Apache as a console app" từ menu Start và màn hình console đột nhiên tắt hoặc nếu ta gặp trục trặc khi khởi động Apache như là một dịch vụ.
Chạy "Command Prompt" từ Start Menu - Programs . Chuyển đến thư mục mà ta đã cài Apache, đánh lệnh apache, và đọc thông báo lỗi. Sau đó xem trước file error.log để tìm hiểu các lỗ. Nếu ta chấp nhận các mặc định khi ta cào đựt Apache, câu lệnh sẽ là:

```
c:
cd "\program files\apache group\apache"
apache
Wait for Apache to exit, or press Ctrl+C
more <logs\error.log
```

Sau khi xem file error.log, ta có thể thấy được lỗi đó là gì và biết cách khắc phục chúng và thử lại lần nữa.
Sau khi khởi động, Apache sẽ chạy (hoặc trong một màn hình console window hoặc là một dịch vụ) nếu được listening tới cổng 80 (Nếu ta không thay đổi Port, Listen hoặc BindAddress chỉ dẫn trong các file cấu hình). Để kết nối vào server và truy cập vào trang web mặc định, chạy trình duyệt và nhập vào địa chỉ URL như sau:

```
http://localhost/
```

Điều này sẽ phản hồi lại với một trang chào mừng, và một liên kết đến trang sách học Apache. Nếu không có gì xảy ra hoặc có một lỗi xuất hiện, hãy xem file error.log trong thư mục logs. Nếu máy của ta không kết nối vào internet, ta phải nhập vào dòng địa chỉ URL như sau:

```
http://127.0.0.1/
```

Một khi các cài đặt đang làm việc, ta sẽ phải cấu hình chính xác bằng cách biên tập các file trong thư mục conf.

Bởi vì Apache không thể chia sẻ cùng một cổng với một ứng dụng TCP/IP khác, ta phải dừng hoặc dỡ bỏ các dịch vụ nào đó trước. Điều này bao gồm cả các web server khác và các sản phẩm firewall như Blackile.Nếu ta chỉ có thể chạy Apache khi dừng các dịch vụ này, hãy cấu hình lại Apache hoặc các sản phẩm khác sao cho chúng không chung cổng listen TCP/IP.Có lẽ ta phải chạy dòng lệnh "netstat -an" để xem cổng nào đã được sử dụng.

2.5_Cấu hình Apache trên Windows

Apache được cấu hình bởi các file trong thư mục conf . Đây cũng là các file được sử dụng để cấu hình cho phiên bản Unix nhưng cũng có một số chỉ dẫn khác trên phiên bản Apache cho Windows.

Bắt đầu cấu hình Apache server bằng cách xem trước httpd.conf và các lời hướng dẫn của nó. Mặc dù các file access.conf và srm.conf đều tồn tại, đây là các file cũ thường không được sử dụng bởi hầu hết các nhà quản trị , và bạn cũng không thấy những lời hướng dẫn ở đây.

httpd.conf chứa tài liệu thật tuyệt vời, băng cách theo các hướng dẫn cấu hình mặc định được khuyên dùng khi bắt đầu với Apache server. Bắt đầu bằng cách đọc các chú thích này để hiểu file cấu hình ,và thực hiện một số nhỏ các thay đổi,bắt đầu Apache trong một màn hình console với mỗt thay đổi. Nếu bạn gặp phải lỗi, thật dễ dàng để sao lưu để cấu hình những gì đã làm lúc cuối cùng. Bạn sẽ có một ý tưởng tốt để hiểu thay đổi nào gây nên lỗi cho server.

Những điều khác nhau chính trong Apache cho Windows là:

- Bởi vì Apache cho Windows là đa luồng, nó không sử dụng một tiến trình riêng biệt cho mỗt yêu cầu như Apache trên Unix. Thay vào đó chỉ có 2 tiến trình đang chạy: một tiến trình cha và một tiến trình con để đón chờ các yêu cầu. Trong tiến trình con mỗi yêu cầu được đón chờ bởi một luồng riêng biệt. Bởi vậy,"cách thức"-các lời chỉ dẫn quản lý là khác nhau:
    - MaxRequestsPerChild - Giống như chỉ thị Unix, nó điều khiển có bao nhiêu yêu cầu mà một tiến trình sẽ phục vụ trước khi tồn tại. Tuy nhiên, không giống như Unix, một tiến trình sẽ phục vụ tất cả các yêu cầu cùng một lúc, không chỉ một, bởi vậy nếu điều này được lập, sẽ có một số lớn được sử dụng. Điều khuyên dùng là mặc định, MaxRequestsPerChild 0, không làm cho tiến trình kết thúc mãi mãi.
    - ThreadsPerChild - Chỉ thị này là mới, và chỉ cho server biết có bao nhiêu luồng nó sẽ dùng. Đây là con số lớn nhất các kết nối mà server có thể đón chờ cùng một lúc; phải chắc chắn và thiết lập số này đủ lớn cho trang của bạn nếu bạn có nhiều việc thành công. Giá trị mặc định khuyên là ThreadsPerChild 50.
- Các chỉ thị chấp nhận tên file làm đối số bây giờ phải dùng tên file của windows thay vì như unix trước đây.Tuy nhiên, bởi vì Apache sử dụng tên kiểu cách Unix cục bộ, bạn phải sử dụng dấu gạch chéo trước, không phải là dấu gạch chéo sau. Các tên chữ cái của ổ đĩa có thể được dùng, nếu bị bỏ sót, ổ đĩa với khả năng thực thi của Apache sẽ được thừa nhận.
- Apache cho Windows có khả năng load các module trong thời gian chạy mà không cần biên dịch lại server. Nếu Apache được biên dịch một cách thông thường, nó sẽ cài đặt một số các module tuỳ chọn trong thư mục \module. Để kích hoạt điều này, hoặc các module khác, chỉ thị LoadModule phải được dùng. Ví dụ, để kích hoạt module trạng thái, ta làm như sau (thông tin thêm về các chỉ thị kích hoạt trạng thái trong file access.conf):

```
LoadModule status_module modules/mod_status.so
```

- Thông tin về việc tạo các module có khả năng tải cũng có sẵn. Chú ý một vài module của các hãng thứ 3 có thể được phát hành với các kiểu cách đặt tên cũ,

ApacheModuleFoo.dll. Thường xuyên thiết lập lệnh LoadModule theo hướng dẫn trong các tài liệu của các hãng thứ 3.

- Apache cho Windows phiên bản 1.3 được thực hiện trong các cuộc gọi đồng bộ. Điều này gây ra một vấn đề to lớn cho các tác CGI , người sẽ không thấy các kết quả không được đệm được gửi trực tiếp tới trình duyệt. Điều này không được mô tả cho CGI trong Apache, nhưng nó có tác động hiệu quả với cổng Windows. Apache 2.0 được cải tiến để thực hiện các tình trạng không đồng bộ được mong chờ, và chúng ta hi vọng tìm ra rằng sự thực thi trong Win NT/2000 cho phép CGI được đối xử như tài liệu đã được cung cấp.
- Apache cũng có thể nạp các phần mở rộng của ISAPI (ví dụ: Internet Server Applications), như thể chúng được dùng bởi Microsoft's IIS, và các server Windows khác. Chú ý rằng Apache không nạp các bộ lọc ISAPI.

Chạy Apache trong một Console Window

Biểu tượng menu Start và trình quản lý dịch vụ NT có thể cung cấp một giao diện đơn giản cho việc quản trị Apache. Nhưng một số trường hợp thì dễ dàng hơn khi làm việc từ dòng lệnh.
Khi làm việc với Apache ta phải biết bằng cách nào nó sẽ tìm các file cấu hình. Ta có thể chỉ ra một file cấu hình cụ thể trên dòng lệnh qua hai cách:

- -f chỉ ra một đường dẫn tới một file cấu hình chuyên biệt:

```
apache -f "c:\my server\conf\my.conf"
apache -f test\test.conf
```

- -n chỉ ra file cấu hình của một dịch vụ Apache đã được cài đặt (Apache 1.3.7 và sau này):

```
apache -n "tên dịch vụ "
```

Trong các trường hợp này, ServerRoot thật sự sẽ được thiết lập trong file cấu hình.
Nếu ta không chỉ ra một tên file cấu hình với -f hoặc -n, Apache sẽ sử dụng tên file được biên dịch trong server, thông thường là "conf/httpd.conf". Gọi Apache với chuyển dịch –V sẽ hiển thị giá trị này được gán nhãn là SERVER_CONFIG_FILE. Apache tiếp đến sẽ định rõ ServerRoot của nó bằng cách cố gắng thực hiện theo trình tự như sau:

- Một chỉ thị ServerRoot thông qua một chuyển tác -C .
- Chuyển tác -d trên dòng lệnh.
- Thư mục hiện tại đang làm việc.
- Một đầu vào đăng ký , được tạo nếu bạn đã thực hiện cài đặt nhị phân.
- Server root đã được biên dịch trong server.

Server root được biên dịch trong server thông thường là "/apache". Thực hiện apache với chuyển tác -V sẽ hiển thị giá trị này được gán nhãn là HTTPD_ROOT.
Khi được thực hiện từ menu start, Apache thông thường bỏ qua các đối số, bởi vậy sử dụng đầu vào nơi đăng ký là kỹ thuật được yêu thích hơn cho console Apache.
Trong khi cài đặt nhị phân, một khoá đăng ký sẽ được cài đặt, ví dụ:
HKEY_LOCAL_MACHINE\Software\Apache Group\Apache\1.3.13\ServerRoot
Khoá này được biên dịch trong server và có thể cho phép bạn kiểm tra các phiên bản mới mà không gây nên hậu quả cho phiên bản hiện tại. Dĩ nhiên ta phải cẩn thận không cài đặt phiên bản mới lên trên phiên bản cũ trong hệ thống file.

Nếu bạn không thực hiện cài đặt nhị phân khi đó Apache sẽ trong một số tình huống gây lỗi về việc thiếu khoá đăng kí. Cảnh báo này có thể bỏ qua nếu có thể tìm thấy các file cấu hình của nó. Giá trị của khoá này là thư mục "ServerRoot" , chứa thư mục conf . Khi Apache bắt đầu, nó sẽ đọc file httpd.conf từ thư mục này. Nếu file này chứa một chỉ thị ServerRoot khác với thư mục mà có khoá trước đó, Apache sẽ quên khoá đăng kí và sử dụng thư mục từ file cấu hình. Nếu bạn copy thư mục Apache hoặc các file cấu hình tới một vị trí mới điều quan trọng là ta cập nhật thư mục ServerRoot trong file httpd.conf tới vị trí mới.
Để chạy Apache từ dòng lệnh như là một ứng dụng console, sử dụng lệnh sau:

```
apache
```

Apache sẽ thực thi và sẽ còn chạy cho đến khi ta nhấn phím ctrl-C.

Điều khiển Apache trong một màn hình Console

Ta có thể chỉ cho Apache trong khi đang chạy phải dừng bằng cách mở một màn hình console khác và chạy lệnh:

```
apache -k shutdown
```

Chú ý: Tuỳ chọn này chỉ có thể có hiệu lực với Apache 1.3.3 và sau này.
Với các phiên bản trước, bạn phải dùng ctrl-C trong màn hình console để tắt server.
Từ phiên bản 1.3.3 tới 1.3.12, điều này sẽ thay cho việc nhấn ctrol-c trong một màn hình console Apache, bởi vì nó cho phép Apache kết thúc bất kỳ các giao dịch hiện thời và thu gọn nhanh chóng.
Tới phiên bản 1.3.13 khi nhấn Control-C trong màn hình đang chạy sẽ làm sạch Apache khá tốt, và bạn có thể dùng –k stop như là một bí danh cho –k shutdown .Các phiên bản trước đó không hiểu –k stop.
Ta cũng có thể chỉ cho Apache khởi động lại. Điều này làm cho nó đọc lại các file cấu hình. Bất kỳ giao dịch nào trong phát triển đều được phép hoàn thiện mà không cần ngắt quãng. Để khởi động lại Apache, chạy:

```
apache -k restart
```

Chú ý: Tuỳ chọn này chỉ có được trong Apache 1.3.3 và sau này. Với các phiên bản trước, ta cần phải dùng Control-C trong màn hình Apache console để tắt server, và khởi động lại với lệnh của Apache.
Chức năng khác rất hữu dụng là tuỳ chọn kiểm tra các file cấu hình .Để kiểm tra các file cấu hình , chạy:

```
apache -t
```

Đây là sự thay đổi sau này đặc biệt hữu dụng với các file cấu hình trong khi Apache thậm chí đang chạy. Ta có thể thực hiện một số thay đổi, khẳng định là cú pháp lệnh "apache -t" là chính xác, kế đến khởi động lại Apache với "apache -k restart". Apache sẽ đọc lại các file cấu hình, cho phép bất kỳ giao dịch nào trong sự tiến triển để hoàn thành mà không phải ngắt quãng.Bất kỳ yêu cầu mới nào cũng sẽ được phục vụ sử dụng cấu hình mới.
Chú ý: Với những người quen thuộc với Apache cho Unix các lệnh này cung cấp một sự tương đương với Windows để kill -TERM *pid* và kill -USR1 *pid*. Tuỳ chọn dòng lệnh được dùng, -k, đã được chọn như là một nhắc nhở lệnh "kill" được sử dụng trong Unix.

II_HỆ QUẢN TRỊ CSDL MySQL

MySQL, cơ sở dữ liệu SQL mã nguồn mở thông dụng nhất , được cung cấp bởi MySQL AB. MySQL AB là một công ty thương mại thực hiện việc tạo ra các dịch vụ cung cấp cho doanh nghiệp đó xung quanh cơ sở dữ liệu MySQL.

1_MySQL là một hệ quản trị CSDL.

Một CSDL là một tập hợp cấu trúc của dữ liệu. Nó có thể là bất kỳ một cái gì từ một danh sách bán hàng đơn giản cho tới gallery ảnh hoặc số lượng lớn các thông tin trong một mạng doanh nghiệp. Để thêm, truy nhập và xử lý dữ liệu được lưu trữ trong một CSDL máy tính, ta cần một hệ quản trị CSDL như MySQL. Từ khi các máy tính thực hiện tốt việc xử lý lượng lớn dữ liệu, quản trị CSDL đóng một vai trò chính yếu trong việc tính toán, như là các công cụ đơn lẻ, hoặc một phần của các ứng dụng khác.

2_MySQL là một hệ quản trị CSDL quan hệ.

Một CSDL quan hệ lưu trữ dữ liệu trong trong một số bảng chuyên biệt tốt hơn là việc đặt toàn bộ dữ liệu trong một nơi lưu trữlớn. Điều này làm tăng thêm tốc độ và sự linh hoạt. Các bảng được liên kết với nhau bằng cách định nghĩa các quan hệ tạo cho nó khả năng kết nối dữ liệu từ một vài bảng khác nhau theo yêu cầu. SQL là một phần của MySQL trong "Structured Query Language"- ngôn ngữ chuẩn thông dụng nhất được dùng để truy nhập các CSDL.

3_MySQL là một phần mềm mã nguồn mở

Mã nguồn mở có nghĩa là nó có thể được sử dụng bởi bất kỳ ai cho mục đích sử dụng hoặc thay đổi nào. Bất kỳ ai cũng có thể download MySQL từ internet và sử dụng nó mà không phải trả bất kỳ một thứ gì. Bất kỳ ai có ý thích cũng có thể nghiên cứu mã nguồn và thay đổi chúng theo yêu cầu của riêng mình. MySQL dùng GPL (GNU General Public License) 'http:// www.gnu. Org', để định ra ta có thể được làm gì và không được làm gì với phần mềm trong các hoàn cảnh khác nhau. Nếu ta cảm thấy khó chịu với GPL hoặc muốn nhúng MySQL trong một ứng dụng thương mại thì ta có thể mua một bản quyền thương mại từ các nhà cung cấp.

4_Lý do dùng MySQL

MySQL rất nhanh, đáng tin cậy và dễ dàng để sử dụng. Nếu điều đó là cái mà bạn đang mong muốn, bạn có thể dùng thử nó. MySQL cũng có một tập các đặc điểm rất thiết thực được phát triển trong một sự hợp tác rất chặt chẽ với người sử dụng. Bạn có thể đem so sánh một cách công phu giữa MySQL và một số hệ quản trị CSDL khác trong trang web chấm điểm của nhà cung cấp.
MySQL đã được phát triển một cách sáng tạo để nắm bắt các CSDL rất lớn và nhanh hơn rất nhiều các giải pháp hiện tại và đã thành công trong việc được sử dụng trong các môi trường sản xuất đòi hỏi cao trong vài năm. Thông qua quá trình phát triển không ngừng, ngày nay, MySQL cung cấp một tập các hàm rất hữu ích và dồi dào. Sự kết nối, tốc độ và sự bảo mật đã làm cho MySQL trở nên thích ứng cao cho việc truy cập các CSDL trên internet.

5_Các đặcđiểm về mặt kỹ thuật của MySQL

MySQL là một hệ thống client/server bao gồm một SQL server đa luồng cho phép hỗ trợ nhiều thiết bị đầu cuối khác nhau, một vài chương trình client khác nhau và các thư viện, các công cụ quản trị và một vài giao diện lập trình.
Các nhà cung cấp cũng cung cấp MySQL như là một thư viện đa luồng mà ta có thể kết nối trong ứng dụng của ta để đạt tới một sản phẩm nhỏ hơn, nhanh hơn, dễ dàng quản lý hơn.
MySQL có nhiều các phần mềm được phân phối có sẵn.
Điều này thật sự thuận tiện cho ta trong việc tìm ứng dụng yêu thích của ta hoặc ngôn ngữ hỗ trợ MySQL.

II_Hypertext Preprocessor - PHP

1_Khái niệm PHP

PHP (một cách chính thức là "PHP: Hypertext Preprocessor") là một ngôn ngữ script được nhúng bên server HTML.
Ví dụ:

```
<html>
  <head>
    <title>Example</title>
  </head>
  <body>

    <?php
    echo "Hi, I'm a PHP script!";
    ?>

  </body>
</html>
```

Chú ý về sự khác nhau của một script được viết bằng các ngôn ngữ khác nhau giống như perl hoặc C – thay vì viết một chương trình với rất nhiều lệnh để xuất ra HTML, ta viết một script HTML với một số mã nhúng để thực hiện một công việc gì đó (trong trường hợp này, đưa ra một số văn bản) . Mã PHP được đóng kín trong các tag bắt đầu và các tag kết thúc đặc biệt cho phép ta nhảy vào và nhảy ra chế độ PHP.
Điều nhận ra PHP từ những gì giống Javascript bên phía Client là mã chương trình được thực hiện bên phía server. Nếu ta đã có một script giống như trên bên phía server của ta, client sẽ nhận các kết quả từ việc chạy script đó, mà không còn cách nào để xác định điều gì bên dưới mã lệnh. Thậm chí ta có thể cấu hình webserver của ta để xử lý tất cả các file HTML của ta với PHP, và vì vậy không còn cách nào mà những người sử dụng có thể biết điều gì ta lên kế hoạch.

2_PHP có thể làm được những gì ?

Tại hầu hết các mức cơ bản nhất, PHP có thể làm bất kỳ điều gì mà các chương trình CGI khác có thể làm, ví dụ như tập hợp các dạng dữ liệu, sinh ra nội dung các trang web động, hoặc gửi và nhận các cookie.
Có lẽ đặc điểm mạnh nhất và thuận tiện nhất trong PHP là nó hỗ trợ khả năng rộng lớn các cơ sở dữ liệu. Viết một trang web có tương tác cơ sở dữ liệu trở nên đơn giản một cách đáng kinh ngạc. Các cơ sở dữliệu dưới đây hiện tại đã được hỗ trợ :

Adabas D	Ingres	Oracle (OCI7 and OCI8)
dBase	InterBase	Ovrimos
Empress	FrontBase	PostgreSQL
FilePro (read-only)	mSQL	Solid
Hyperwave	Direct MS-SQL	Sybase
IBM DB2	MySQL	Velocis
Informix	ODBC	Unix dbm

PHP cũng hỗ trợ cho việc "nói chuyện" với các dịch vụ khác sử dụng các thao tác như IMAP,SNMP,NNTP,POP3,HTTP và vô số giao thức khác. Ta cũng có thể mở các socket mạng mới và tương tác sử dụng các giao thức khác.

3_Bản tóm tắt lịch sử của PHP

PHP đã được nghĩ đến trong khoảng cuối năm 1994 bởi Rasmus Lerdorf. Các phiên bản không phát hành trước đó đã được dùng trên chính trang chủ của anh ta để theo dõi ai đang tìm bản lý lịch trực tuyến của anh ta. Phiên bản đầu tiên được dùng bởi những người khác đã có sẵn trong khoảng thời gian trước năm 1995 và đã được biết đến như là các công cụ trang chủ cá nhân. Nó bao gồm một bộ máy phân tích từ loại một cách đơn giản mà chỉ được hiểu là một số ít các macro đặc biệt và một số các tiện ích mà được dùng một cách thông dụng trên các trang chủ trước đó. Bộ phân tích từ loại đã được viết lại vào giữa năm 1995 và có tên là PHP/FI phiên bản 2. FI có được lạ do từ một gói khác của Rasmus đã được viết lại được biên dịch định dạng dữ liệu HTML. Anh ta đã kết hợp các script các công cụ trang chủ cá nhân với trình biên dịch form và thêm vào hỗ trợ mSQL và PHP/FI đã ra đời. PHP/FI đã phát triển lên một cách đáng kinh ngạc và mọi người đã bắt đầu đóng góp mã cho nó.

Để thống kê một cách nghiêm khắc là một điều phức tạp, nhưng ước lượng khoảng cuối năm 1996 PHP/FI đã được dùng trên ít nhất 15000 trang web trên khắp thế giới. Khoảng giữa năm 1997 con số này đã tăng lên trên 50000 trang web. Giữa năm 1997 cũng đã thấy một sự thay đổi trong việc phát triển PHP. Nó thay đổi từ việc sở hữu dự án cưng của Rasmus rằng một nhóm người đã đóng góp vào, để có thêm nhiều sự thống nhất có trật tự của nhóm sao cho đạt hiệu quả cao.

Bộ phân tích từ loại đã được viết lại một cách hỗn tạp bởi Zeev Suraski và Andi Gutmans và bộ phân tích từ loại mới này đã định hình nền tảng cho phiên bản 3 của PHP. Nhiều mã tiện ích từ PHP/FI đã được dùng cho PHP3 và nhiều trong số đó đã được viết lại một cách hoàn toàn.

Phiên bản PHP4 dùng bộ máy scripting Zend để phân phối sự thực hiện cao cấp hơn, hỗ trợ một mảng các thư viện và các mở rộng của các hãng thứ 3 rộng rãi hơn và chạy như là một module server địa phương với toàn bộ các web server được ưa chuộng.

Ngày nay, PHP 3 hoặc PHP 4 hiện tại chuyên chở một số lượng các sản phẩm thương mại như web server Red Hat’s Stronghold. Ngày nay, theo ước lượng thì PHP được dùng trong khoảng 5.1 triệu trang trên toàn thế giới và hơn cả IIS server của Microsoft (khoảng 5.03 triệu trang).

4_Cài đặt PHP trên Windows

Cài đặt trên các hệ thống Windows 98/Me/NT/2000/XP

Có hai cách chính để cài đặt PHP cho Windows: Cách thông thường hoặc bằng cách sử dụng chương trình InstallShield installer.

4.1_Windows InstallShield

Trình cài đặt PHP trênWindows có trên trang web tại www.php.net. Nó cài đặt phiên bản CGI của PHP và, cho IIS, PWS, và Xitami, cấu hình web server tốt.

Chú ý rằng phiên bản này không cài đặt bất kỳ phần mở rộng hoặc server api phiên bản của PHP nào.

Cài đặt HTTP server mà ta đã chọn trên hệ thống của ta và đảm bảo rằng nó hoạt động tốt.

Chạy trình cài đặt và theo các hướng dẫn được cung cấp bởi trình cài đặt tự động. Có hai loại cài đặt được hỗ trợ, standard, cung cấp các giá trị mặc định cho tất cả các thiết lập có thể , và advance, yêu cầu một số câu hỏi để hoàn tất quá trình cài đặt.

Trình cài đặt tự động hội tụ đủ các thông tin để thiết lập file php.ini và cấu hình webserver để sử dụng PHP.

Một khi tiến trình cài đặt được hoàn thành, trình cài đặt sẽ báo tin cho ta nếu ta muốn khởi động lại hệ thống , khởi động lại server, hoặc chỉ bắt đầu dùng PHP.

4.2_Hướng dẫn cách cài đặt từ file nhị phân zip trên Windows

Phần hướng dẫn cài đặt này sẽ giúp ta cài đặt và cấu hình PHP trên các webserver Windows 9x/Me/NT/2000/XP .Phần hướng dẫn này có thể phù hợp với các phiên bản của :

Apache 1.3.x
Apache 2.0.x (bản thí nghiệm)
OmniHTTPd 2.0b1 và trên nữa
Oreilly Website Pro
Xitami
Netscape Enterprise Server, iPlanet

PHP 4 cho Windows với hai cách thức thực hiện - một chương trình CGI có khả năng thực thi (php.exe), và một vài module SAPI (ví dụ php4isapi.dll). Dạng sau thì mới đối với PHP4 và cung cấp cải tiến một cách đáng kể các thao tác và một vài chức năng mới.Tuy vậy, chú ý rằng các module SAPI không thực sự được xem như là sản phẩm chất lượng cao. Trong trường hợp cá biệt, với module ISAPI, bạn sẽ gặp phải những vấn đề có tính đăc biệt trầm trọng trên các nền cũ hơn .

Lý do cho điều này là các module PHP SAPI đang sử dụng phiên bản luồng an toàn cho mã PHP, điều này mới so với PHP,và cũng chưa được kiểm nghiệm và kiểm chứng một cách đầy đủ để có thể xem như là hoàn toàn vững chắc và ổn định, và sự thực là cũng có một số nhỏ lỗi .Mặt khác, một vài người đưa ra báo cáo các kết quả rất tốt với các module,và có rất ít các báo cáo có vấn đề với phiên bản module Apache.

Nếu ta chọn một trong các module SAPI và dùng Windows 95, hãy chắc rằng ta phải tải bản cập nhật DCOM tại địa chỉ:

http://download.microsoft.com/msdownload/dcom/95/x86/en/dcom95.exe"

Cho module ISAPI, yêu cầu một bản ISAPI 4.0 tương thích Web server

(đã được kiểm tra trên IIS 4.0, PWS 4.0 và IIS 5.0). IIS 3.0 không được hỗ trợ; Ta sẽ phải tải về và cài đặt WinNT4.0 và tuỳ chọn IIS 4.0 nếu ta muốn PHP hỗ trợ .

Các bước dưới đây sẽ thao tác trên tất cả các trình cài đặt trước khi có các chỉ dẫn rõ ràng với server.
Giải nén file được tải vào một thư mục nào đó.
C:\PHP\ là tốt nhất để bắt đầu.

Bạn cần phải chắc rằng các file dlls mà php dùng có thể được tìm th ấy một cách dễ dàng. Các dlls rõ ràng sẽ liên quan đến loại webserver nào ta dùng và nơi nào ta muốn chạy php như là một cgi hoặc là một module server. Php4ts.dll thường xuyên được sử dụng. Nếu ta đang dùng một server module (ví dụ isapi hoặc apache) khi đó ta cần các file dlls liên quan từ thư mục sapi . Nếu ta đang sử dụng các php dll mở rộng khi đó ta sẽ cần chúng . Để chắc rằng các dll có thể được tìm thấy, ta có thể copy chúng vào thư mục hệ thống (winnt/system hoặc windows/system) hoặc bạn phải chắc rằng chúng chứa trong cùng một thư mục với thư mục có chứa php thực thi hoặc dll webserver của ta (ví dụ php.exe, php4apache.dll).
Chép file php.ini vào thư mục Windows hoặc Winnt, winnt4.
Biên tập file php.ini :
Ta cần phải thay đổi thiết lập 'extension_dir' tới thư mục cài đặt php hoặc nơi ta đặt các file 'php_*.dll', ví dụ c:\php
Đặt 'doc_root' chỉ tới webservers document_root. Ví dụ: c:\apache\htdocs hoặc c:\webroot
Chọn các mở rộng ta muốn được tải khi php,chú ý một số mở rộng đã được tạo ra trong các phiên bản của Windows.
Ta có thể bỏ các dấu chú thích trên các dòng như 'extension=php_*.dll' trong php.ini để tải các mở rộng.
Chú ý rằng thư mục mibs được cung cấp với phát hành của Windows chứa các file hỗ trợ cho SNMP. Thư mục này sẽ được chuyển tới DRIVE:\usr\mibs (DRIVE là ổ đĩa nơi ta cài PHP)

4.3_Cài đặt các chức năng mở rộng trên Windows

Sau khi cài đặt PHP và một webserver trên windows, ta có lẽ cũng muốn cài đặt thêm một số mở rộng để thêm các chức năng. Bảng dưới đây mô tả một vài chức năng có sẵn. Ta cũng có thể thêm các chức năng bằng cách bỏ các dấu chú thích đứng trước các file php_*.dll trong file php.ini.
Chú ý:
Một vài mở rộng dll yêu cầu theo mở rộng của PHP , ta nên copy các file dll từ thư mục dlls trong gói phát hành vào thư mục windows/system hoặc winnt/system32.
Nếu ta đang có các file dll này rồi, ta chỉ ghi đè chúng trong trường hợp một vấn đề gì đó làm việc không chính xác. Trước khi ghi đè chúng, tốt nhất là ta sao lưu chúng hoặc chuyển chúng vào một thư mục khác chỉ để đề phòng một cái gì đó lỗi.
Download phiên bản sau nhất của Microsoft Data Access Components (MDAC)
Cho nền của bạn, đăc biệt với người dùng Microsoft Windows 9x/NT4.
MDAC có sẵn tại địa chỉ http://www.microsoft.com/data/ .
Cũng cần chú ý rằng một số mở rộng đòi hỏi các thư viện của các hãng thứ 3, ví dụ php_oci8.dll cần các thư viện client Oracle 8 được cài đặt trên hệ thống của ta. Điều này không được cung cấp với PHP.
Chú ý: Các dlls cho phần mở rộng của PHP được bắt đầu bằng tiền tố ‘php_’ điều này tránh sự lộn xộn giữa các mở rộng của PHP và các thư viện của các hãng thứ 3 cung cấp.
Chú ý:
Trong PHP 4.0.5 hỗ trợ MySQL, ODBC, FTP, Calendar, BCMath, COM, PCRE,
Session, WDDX và XML được kết hợp vào.

5_Cấu hình PHP với Web server

5.1_Cài đặt PHP trên Windows với Apache 1.3.x

Có hai cách để thiết lập PHP để làm việc được với Apache 1.3.x trên Windows.
Một là sử dụng file nhị phân CGI (php.exe), cách còn lại là dùng Apache module .dll. Trong cách sau, ta phải dừng Apache server, và biên tập file httpd.conf hoặc srm.conf để cấu hình Apache làm việc với PHP.

5.2_Cài đặt PHP cho Apache như là một module

Phiên bản 4.1 giới thiệu module sapi an toàn hơn, theo đánh giá thì ta nên cấu hình PHP như là một module của Apache thì tốt hơn.
Để thực hiện điều này, ta phải nạp file php4apache.dll trong Apache httpd.conf.
Chú ý:
Bất kỳ khi nào ta nạp php4apache.dll , php4apache.dll cần có php4ts.dll để được tính đến trong phân bố PHP 4. php4apache.dll dựa trên php4ts.dll được nạp sớm như là Apache tải php4apache.dll. Nếu php4ts.dll không được tìm thấy, ta sẽ nhận được một lỗi :
Cannot load c:/php/sapi/php4apache.dll into server
Vậy thật sự php4ts.dll được nạp từ đâu ?
php4ts.dll được tìm trong thứ tự như sau :

1) trong thư mục nơi apache.exe từ đó được bắt đầu
2) trong thư mục mà php4apache.dll từ đó được nạp
3) trong thư mục %SYSTEMROOT%\System32, %SYSTEMROOT%\system và %SYSTEMROOT% .
   Chú ý: %SYSTEMROOT%\System32 chỉ đúng với Windows NT/2000/XP)
4) trong toàng bộ %PATH%

Thông thường ta chỉ cần copy nó vào thư mục %SYSTEMROOT%\System32.
Sau khi ta đã thiết lập một cách đúng đắn, ta sẽ cấu hình Apache để nạp module PHP4 chỉ cần thêm vài dòng vào file httpd.conf:
LoadModule php4_module c:/php/sapi/php4apache.dll
AddModule mod_php4.c
AddType application/x-httpd-php .php
Chú ý: Đặc biệt các phiên bản mới hơn của Apache không cần định hướng AddModule .
Ta phải đặt file php.ini ở một trong hai nơi sau:
1) trong thư mục cài đặt Apache (ví dụ. c:\apache\apache)
2) trong thư mục %SYSTEMROOT% .

5.3_Cài đặt PHP cho Apache như là CGI nhị phân

Nếu bạn muốn cài đặt PHP như là CGI nhị phân, trước hết hãy đọc trang này trước:

http://www.cert.org/advisories/CA-1996-11.html

và sau đó nếu ta thật sự chắc chắn, cần chèn những dòng sau vào file conf:
ScriptAlias /php/ "c:/php/"
AddType application/x-httpd-php .php

Action application/x-httpd-php "/php/php.exe"
Để phòng ngừa xa, ta nên thay "/php" ScriptAlias thành một cái gì đó ngẫu nhiên hơn, để ngăn chặn nhị phân được gọi trực tiếp, như là một sự bảo mật rủi ro.
Nhớ rằng khi ta hoàn thành việc khởi động lại server, ví dụ,
NET STOP APACHE
tiếp đến là
NET START APACHE
Để sử dụng chức năng mã nguồn được làm sáng, thêm dòng sau vào file apache httpd.conf :
AddType application/x-httpd-php-source .phps
Chú ý, điều này chỉ làm việc khi ta cài đặt php là module sapi.
Nếu bạn thích dùng đặc điểm này với cgi nhị phân, hãy tạo một file mới và dùng hàm show_source("path/to/original_file.php");

5.4_Gạch trái hay gạch phải trong đường dẫn?

Trên Win-Apache, tên đường dẫn có thể gồm cả gạch trái và gạch phải.
Ví dụ:
LoadModule php4_module C:\php\sapi\php4apache.dll
làm việc tốt như:
LoadModule php4_module C:/php/sapi/php4apache.dll
Ta cũng có thể trộn lẫn hai kiểu gạch:
LoadModule php4_module C:\php/sapi\php4apache.dll

CHƯƠNG 2 : NGÔN NGỮ LẬP TRÌNH SCRIPT PHP

I_CÚ PHÁP CƠ BẢN

1_Sự thoát khỏi mã HTML

Có 4 cách để thoát khỏi mã HTML và thực hiện " chế độ mã PHP"

Ví dụ 1: Các cách để thoát khỏi mã HTML:

```
1.  <? echo ("this is the simplest, an SGML processing instruction\n"); ?>

2.  <?php echo("if you want to serve XHTML or XML documents, do like this\n"); ?>

3.  <script language="php">
        echo ("some editors (like FrontPage) don't
              like processing instructions");
    </script>

4.  <% echo ("You may optionally use ASP-style tags"); %>
    <%= $variable; # This is a shortcut for "<%echo .." %>
```

Cách thứ nhất chỉ có thể được thực hiện có nếu các tag ngắn đã được làm cho có thể. Điều này có thể được thực hiện bằng cách cho phép cấu hình short open tag thiết lập trong file cấu hình của PHP, hoặc bằng cách biên dịch PHP với tuỳ chọn –enable-short-tags để cấu hình.

Cách thứ hai là phương pháp thông thường được yêu thích hơn, là nó cho phép thế hệ tiếp theo XHTML có thể dễ dàng được thực thi với PHP.

Cách thứ ba là chỉ ra rõ PHP l à một ngôn ng ữ script.

Cách thứ tư thì chỉ được phép nếu các tag kiểu ASP đã được phép sử dụng các tag asp thiết lập cấu hình.

Chú ý: Hỗ trợ cho các tag ASP đã được thêm vào trong phiên bản 3.0.4

Tag đóng cho khối sẽ bao gồm đường thẳng mới kéo dài ngay lập tức nếu nó được hiển thị.

2_Ngăn cách các chỉ dẫn

Các chỉ dẫn được ngăn cách giống như trong ngôn ngữ C hoặc Perl-kết thúc mỗi câu lệnh là một dấu chấm phẩy.
Tag đóng (?>) cũng ngụ ý tới cuối của dòng lệnh, bởi vậy sau đây là các câu lênh tương đương:

```
<?php
   echo "This is a test";
?>
<?php echo "This is a test" ?>
```

3_Các lời chú thích

PHP hỗ trợ các kiểu chú giải như của 'C','C++' và shell Unix . Ví dụ:

```
<?php
   echo "This is a test"; // This is a one-line c++ style comment
   /* This is a multi line comment
   yet another line of comment */
   echo "This is yet another test";
   echo "One Final Test"; # This is shell-style style comment
?>
```

Các kiểu chú giải "one line" thật sự chỉ chú giải tới cuối dòng hoặc khối hiện tại của mã lệnh PHP, được đưa lên trước.

```
<h1>This is an <?php # echo "simple";?> example.</h1>
<p>The header above will say 'This is an example'.
```

Bạn sẽ phải cẩn thận không được đặt các chú thích của 'C', nó có thể xảy ra khi ghi chú các khối lớn.

```
<?php
 /*
   echo "This is a test"; /* This comment will cause a problem */
 */
?>
```

II_CÁC KIỂU DỮ LIỆU

1_Kiểu số nguyên

Các số nguyên có thể được chỉ ra bằng cách sử dụng bất kỳ các cú pháp sau:

```
$a = 1234; # số thập phân
$a = -123; # Một số âm
$a = 0123; # số hệ cơ số 8 (tương đương số 83 hệ thập phân)
$a = 0x12; # số hệ 16 (tương đương số 18 hệ thập phân)
```

Kích cỡ của một số nguyên phụ thuộc vào nền, mặc dù một giá trị lớn nhất khoảng 2 tỷ cũng là một giá trị thông thường (đó là 32 bit đã được đánh trị).

2_Các số thực dấu chấm động

Các số thực dấu chấm động ("doubles") có thể được chỉ ra bằng cách dùng một trong các cú pháp sau:

```
$a = 1.234; $a = 1.2e3;
```

Kích cỡ của một số thực dấu chấm động phụ thuộc vào nền, mặc dù một giá trị lớn nhất tương đương 1.8e308 với mức đúng đắn đến 10 chữ số thập phân cũng chỉ là một giá trị thông thường(đó là định dạng 64 bit IEEE).

Chú ý:

Cũng khá thông thường rằng một số thập phân giống như 0.1 hoặc 0.7 không thể được chuyển đổi thành các bản sao nhị phân của chính bản thân chúng mà không có một chút sai lệch nào. Điều này có thể dẫn đến những kết quả khó hiểu.

Ví dụ: floor((0.1+0.7)*10) thông thường sẽ trả về 7 thay vì chính xác là 8 là vì kết quả cuối cùng có thể là một số: 7.9999999999.....

Điều này có liên quan đến một sự thật là không thể chính xác trong một số các biểu thức phân số trong ký hiệu thập phân với một sự hạn chế về các số. Ví dụ: 1/3 trong dạng thập phân trở thành 0.333333333...
Bởi vậy không bao giờ có giá trị chính xác cho các kết quả là một số động và không bao giờ so sánh các số trỏ động với sự ngang bằng. Nếu ta thật sự muốn sự đúng đắn cao hơn, ta sẽ dùng các hàm toán học chuyên biệt hoặc thay vì các hàm gmp.

3_Các chuỗi

Các chuỗi có thể được chỉ ra bằng cách dùng một trong hai tập phân định.
Nếu chuỗi được đóng lại trong dấu nháy kép(“), số lượng bên trong chuỗi sẽ được mở rộng ra (phụ thuộc vào một số hạn chế của từ loại). Cũng như C và Perl, dấu xổ ngược (“\”) có thể được sử dụng trong việc chỉ ra các ký tự đặc biệt:

Thứ tự	**Ý nghĩa**
\n	xuống dòng (LF hoặc 0x0A trong ASCII)
\r	trở về đầu dòng (CR hoặc 0x0D trong ASCII)
\t	Tab ngang (HT hoặc 0x09 trong ASCII)
\\	xổ ngược
\$	dấu đô la
\"	Nháy kép
\[0-7]{1,3}	thứ tự của các ký tự phù hợp với biểu thức thông thường là một ký tự trong hệ cơ số 8
\x[0-9A-Fa-f]{1,2}	thứ tự của các ký tự phù hợp với biểu thức thông thường là một ký tự trong hệ cơ số 16

Nếu ta cố gắng để đưa ra bất kỳ một ký tự nào khác , cả dấu xổ trái và ký tự sẽ là đầu ra. Trong PHP3, một lời cảnh báo sẽ được phát ra tại mức E_NOTICE khi điều này xảy ra. Trong PHP4, không một lời cảnh báo nào được đưa ra.
Cách thứ hai để đưa ra một chuỗi sử dụng dấu nháy đơn(“ ‘ “).Khi một chuỗi được bao bởi dấu nháy đơn, chỉ có những bỏ qua sẽ được hiểu là “\\” và “\”.Điều này nhằm mục đích tạo sự thuận lợi, bởi vậy ta có thể dùng dấu nháy đơn hoặc dấu xổ ngược trong một chuỗi trích dẫn đơn. Số lượng sẽ không được trải rộng ra trong một chuỗi trích dẫn đơn.
Cách khác để định giới các chuỗi là sử dụng cú pháp (“<<<”). Trước hết sẽ cung cấp một định danh sau khi thực hiện <<<, tiếp theo là chuỗi, và sau đó là định danh giống ban đầu để đóng chú thích.
Định danh đóng phải bắt đầu trong cột đầu tiên của dòng. Cũng vậy, định danh được sử dụng phải theo sau các quy tắc đặt tên giống nhau như là các nhãn khác trong PHP:Nó chỉ chứa các ký tự chữ và số và các dấu gạch dưới, và phải bắt đầu với ký tự không phải là số hoặc gạch dưới.

```
<?php
$str = <<<EOD
Example of string
spanning multiple lines
using heredoc syntax.
EOD;

/* More complex example, with variables. */
class foo {
   var $foo;
   var $bar;

   function foo() {
      $this->foo = 'Foo';
      $this->bar = array('Bar1', 'Bar2', 'Bar3');
   }
}

$foo = new foo();
$name = 'MyName';

echo <<<EOT
My name is "$name". I am printing some $foo->foo.
Now, I am printing some {$foo->bar[1]}.
This should print a capital 'A': \x41
EOT;
?>
```

Các chuỗi Strings may be concatenated using the '.' (dot) operator. Note that the '+' (addition) operator will not work for this. Please see String operators for more information.
Characters within strings may be accessed by treating the string as a numerically-indexed array of characters, using C-like syntax. See below for examples.

Ví dụ. Vài ví dụ về chuỗi

```php
<?php
/* Assigning a string. */
$str = "This is a string";

/* Appending to it. */
$str = $str . " with some more text";

/* Another way to append, includes an escaped newline. */
$str .= " and a newline at the end.\n";

/* This string will end up being '<p>Number: 9</p>' */
$num = 9;
$str = "<p>Number: $num</p>";

/* This one will be '<p>Number: $num</p>' */
$num = 9;
$str = '<p>Number: $num</p>';

/* Get the first character of a string  */
$str = 'This is a test.';
$first = $str[0];

/* Get the last character of a string. */
$str = 'This is still a test.';
$last = $str[strlen($str)-1];
?>
```

Sự chuyển đổi chuỗi

Khi một chuỗi được xác định là một giá trị số, giá trị kết quả và kiểu được mô tả như sau:
Chuỗi sẽ ước lượng như là một số double nếu nó chứa bất kỳ một trong số các ký tự '.','e', or 'E'. Trong trường hợp khác, nó sẽ ước lượng là một số nguyên.
Giá trị được xác định bởi phần ban đầu của chuỗi. Nếu chuỗi bắt đầu với số liệu là số, nó sẽ là giá trị được dùng. Trong trường hợp khác, giá trị sẽ là 0 . Số liệu dạng số là một dấu hiệu tuỳ chọn, được theo sau bởi một hoặc nhiều con số(tuỳ chọn chứa một dấu chấm thập phân), được theo sau bởi một tuỳ chọn số mũ. Số mũ là một ký tự'e' hoặc 'E' đượctheo sau bởi một hoặc nhiều số.
Khi biểu thức đầu tiên là một chuối, dạng cảu biến sẽ phụ thuộc vào biểu thức thứ hai.

```
$foo = 1 + "10.5";              // $foo is double (11.5)
$foo = 1 + "-1.3e3";            // $foo is double (-1299)
$foo = 1 + "bob-1.3e3";         // $foo is integer (1)
$foo = 1 + "bob3";              // $foo is integer (1)
$foo = 1 + "10 Small Pigs";     // $foo is integer (11)
$foo = 1 + "10 Little Piggies"; // $foo is integer (11)
$foo = "10.0 pigs " + 1;        // $foo is integer (11)
$foo = "10.0 pigs " + 1.0;      // $foo is double (11)
```

Để có thêm thông tin về sự chuyển đổi này, sem thêm trang Unix manual cho strod(3).
Nếu bạn muốn kiểm tra bất kỳ mọt ví dụ nào trong phần này, bạn cố thể cắt và dán các vị dụ và chèn dòng sau để xem điều gì sẽ xảy ra:

```
echo "\$foo==$foo; type is " . gettype ($foo) . "<br>\n";
```

4_Các mảng

Các mảng thật sự vừa giống các bảng hỗn tạp (Các mảng có liên kết với nhau) vừa giống các mảng chỉ số(các vector).

4.1_Mảng một chiều

PHP hỗ trợ cả mảng vô hướng và mảng có hướng. Trong thực tế, không có gì khác biệt giữa chúng. Ta có thể tạo một mảng sử dụng các hàm list() hoặc array(), hoặc ta có thể thiết lập các giá trị của các phần tử của mảng một cách rõ ràng.

```
$a[0] = "abc";
$a[1] = "def";
$b["foo"] = 13;
```

Ta cũng có thể tạo một mảng bằng cách đơn giản là thêm các giá trị vào mảng. Khi ta gán một giá trị vào một biến mảng dùng dấu ngoặc đơn trống, giá trị sẽ được thêm vào cuối mảng.

```
$a[] = "hello"; // $a[2] == "hello"
$a[] = "world"; // $a[3] == "world"
```

Các mảng có thể được sắp xếp bằng cách dùng các hàm asort() , arsort() , ksort() , sort() , uasort() , usort() , và uksort().
Ta có thể đếm số các phần tử trong một mảng bằng cách dùng hàm count().
Ta có thể duyệt qua một mảng bằng cách dùng các hàm next() và prev(). Một cách thông thường khác để duyệt qua các phần tử của mảng là dùng hàm each().

4.2_Các mảng đa chiều

Các mảng nhiều chiều thật sự khá đơn giản. Cho mỗi chiều của mảng, ta có thể thêm giá trị khác[khoá] vào cuối:

```
$a[1]     = $f;             # one dimensional examples
$a["foo"]  = $f;
```

```
$a[1][0]    = $f;          # two dimensional
$a["foo"][2] = $f;          # (you can mix numeric and associative indices)
$a[3]["bar"] = $f;          # (you can mix numeric and associative indices)

$a["foo"][4]["bar"][0] = $f;   # four dimensional!
```
Trong PHP 3 không cho phép tham chiếu các mảng đa chiều trực tiếp trong các chuỗi. Ví dụ sau đây sẽ không cho kết quả mong muốn:
```
$a[3]['bar'] = 'Bob';
echo "This won't work: $a[3][bar]";
```
Trong PHP 3, ở trên sẽ cho đầu ra là: This won't work: Array[bar].
Ta có thể làm lại như sau:
```
$a[3]['bar'] = 'Bob';
echo "This will work: " . $a[3][bar];
```
Tuy nhiên, trong PHP 4, tất cả các vấn đề có thể được dùng mẹo bằng cách đóng tham chiếu mảng(bên trong chuỗi) trong dấu ngoặc nhọn:
```
$a[3]['bar'] = 'Bob';
echo "This will work: {$a[3][bar]}";
```

Ta có thể điền đầy các mảng đa chiều bằng nhiều cách, nhưng một điều phức tạp để hiểu là bằng cách nào để dùng lệnh array() để cho các mảng kết hợp. Có 2 đoẫnm điền đầy mảng một chiều trong cùng một cách:
```
# Example 1:

$a["color"]= "red";
$a["taste"] = "sweet";
$a["shape"]          = "round";
$a["name"]           = "apple";
$a[3]                = 4;

# Example 2:
$a = array(
    "color" => "red",
    "taste" => "sweet",
    "shape" => "round",
    "name"  => "apple",
    3       => 4
);
```
Hàm array() có thể được lồng vào nhau để cho ra các mảng nhiều chiều:
```
<?php
$a = array(
    "apple"  => array(
        "color"  => "red",
        "taste"  => "sweet",
        "shape"  => "round"
    ),
    "orange"  => array(
        "color"  => "orange",
        "taste"  => "tart",
        "shape"  => "round"
    ),
    "banana"  => array(
        "color"  => "yellow",
        "taste"  => "paste-y",
        "shape"  => "banana-shaped"
    )
);
```

```
echo $a["apple"]["taste"];    # will output "sweet"
?>
```

5_Các đối tượng

Khởi tạo đối tượng

Để khởi tạo một đối tượng, ta dùng câu lệnh new để đưa đối tượng tới một biến.

```
<?php
class foo {
   function do_foo() {
      echo "Doing foo.";
   }
}

$bar = new foo;
$bar->do_foo();
?>
```

Để thảo luận kỹ hơn, tham khảo phần các lớp và các đối tượng.

6_Kiểu linh hoạt

PHP không yêu cầu (hoặc hỗ trợ) định nghĩa kiểu rõ ràng trong việc khai báo biến. một kiểu biến được định nghiã bởi ngữ cảnh trong đó biến được sử dụng. Điều đó muốn nói rằng nếu ta gán một giá trị chuỗi cho một biến var , var sẽ trở thành một chuỗi. Nếu tiếp theo ta gán một giá trị số nguyên cho biến var, nó sẽ trở thành một số nguyên.

Một ví dụ về việc chuyển đổi kiểu tự động là toán tử cộng '+'. Nếu bất kỳ một toán hạng là một số double, khi đó toàn bộ các toán hạng được gán trị là các số double, và giá trị trả về là một số double. Trường hợp khác, các toán hạng sẽ được thể hiện là các số nguyên, và kết quả cũng sẽ là một số nguyên. Chú ý rằng điều này không làm thay đổi bản thân các toán hạng; điều thay đổi là bằng cách nào các toán hạng được đin giá.

```
$foo = "0";  // $foo is string (ASCII 48)
$foo++;      // $foo is the string "1" (ASCII 49)
$foo += 1;   // $foo is now an integer (2)
$foo = $foo + 1.3;  // $foo is now a double (3.3)
$foo = 5 + "10 Little Piggies"; // $foo is integer (15)
$foo = 5 + "10 Small Pigs";     // $foo is integer (15)
```

Nếu muốn kiểm tra bất kỳ các ví dụ nào trong mục này, ta có thể cắt và dán các ví dụ và chèn các dòng dới đây để ta có thể tự kiểm tra.

```
echo "\$foo==$foo; type is " . gettype ($foo) . "<br>\n";
```

Chú ý: Cách thức chuyển đổi tự động sang mảng hiện chưa được xác đin rõ.

```
$a = 1;       // $a is an integer
$a[0] = "f";  // $a becomes an array, with $a[0] holding "f"
```

Trong khi ví dụ trên có lẽ xem như nó cho kết quả rõ ràng trong $A trở thành một mảng, phần tử đầu tiên của nó là ‘f’, có thể xem như sau:

```
$a = "1";     // $a is a string
$a[0] = "f";  // What about string offsets? What happens?
```

Từ khi PHP hỗ trợ chỉ mục trong các chuỗ dựa trên các khoảng trống sử dụng chung cú pháp như chỉ số mảng, ví dụ trên chỉ ra một vấn đề: có thể $a trở thành một mảng với phần tử đầu tiên của nó là ’f’ , hoặc có lẽ ‘f’ trở thành ký tự đầu tiên của chuỗi $a ?

Đối với PHP 3.0.12 và PHP 4.0b3-RC4, kết quả của việc tự động chuyển đổi kiểu được xem như không được xác định.

Ép kiểu

Ép kiểu trong PHP làm việc giống như trong C: Tên của kiểu mong đợi được viết trong các dấu ngoặc trước biến mà nó sẽ được ép kiểu.

```
$foo = 10;   // $foo is an integer
$bar = (double) $foo;   // $bar is a double
```

Các loại ép kiểu cho phép là:

- (int), (integer) – ép sang số nguyên
- (real), (double), (float) – ép sang số double
- (string) – ép sang kiểu chuỗi
- (array) – ép sang kiểu mảng
- (object) – ép sang kiểu đối tượng

Chú ý rằng các tab và các khoảng trắng cũng được phép trong các dấu ngoặc, bởi vậy các biểu thức dưới đây cho kết quả tương tự:

```
$foo = (int) $bar;
$foo = ( int ) $bar;
```

Nó có thể không được thật sự rõ ràng chính xác rằng điều gì sẽ xảy ra khi ép kiểu giữa các kiểu nào đó. Ví dụ sau sẽ chỉ ra điều đó.

Khi ép kiểu từ một biến kiểu vô hướng hoặc một biến kiểu chuỗi sang một mảng, biến đó sẽ trở thành phần tử đầu tiên của mảng:

```
$var = 'ciao';
$arr = (array) $var;
echo $arr[0];  // outputs 'ciao'
```

Khi ép kiểu từ một biến vô hướng hoặc một biến chuỗi sang một đối tượng, biến đó sẽ trở thành một thuộc tính của đối tượng, tên của thuộc tính đó sẽ là ‘vô hướng’

```
$var = 'ciao';
$obj = (object) $var;
echo $obj->scalar;  // outputs 'ciao'
```

III_CÁC BIẾN SỐ

1_Khái niệm cơ bản

Biến số ở trong PHP được mô tả bởi kí hiệu của đồng đôla theo sau bởi tên của biến .Tên của biến được phân biệt bởi chữ hoa và chữ thường .

Tên của các biến theo sau bởi các quy tắc giống như các nhãn khác ở trong PHP .Một tên biến đúng bắt đầu với một kí tự hoặc một đường gạch chân , được theo sau bởi bất cứ số lượng kí tự nào ,các con số hoặc các đường gạch chân .Khi diến đạt đúng thì nó sẽ được thể hiện là:' [a-zA-Z_\x7f-\xff] [a-zA-Z0-9_\x7f-\xff]*

Chú ý:Do mục đích sử dụng của chúng ta ,một kí tự là a-z,A_Z và các kí tự của bộ mã ASCII từ 127 đến 255 (ox7f-oxff).

```
$var = "Bob";
$Var = "Joe";
echo "$var, $Var";      // outputs "Bob, Joe"

$4site = 'not yet';     // invalid; starts with a number
$_4site = 'not yet';    // valid; starts with an underscore
$täyte = 'mansikka';    // valid; 'ä' is ASCII 228.
```

Trong PHP3 các biến luôn luôn được gán bởi các giá trị . Điều đó có nghĩa là khi ta gán một biểu thức cho một biến thì toàn bộ giá trị của biểu thức nguồn sẽ được sao chép đến biến đích . Điều này có nghĩa là ,ví dụ như sau khi gán một giá trị của một biến cho biến kia, sự thay đổi của một trong số những biến này sẽ không ảnh hưởng đến biến kia . Để biết thêm thông tin về sự gán trị này hãy xem mục Expressions

PHP4 cung cấp một cách khác để gán các giá trị cho các biến gán bằng cách tham chiếu . Điều này có nghĩa là biến mới dễ dàng tham chiếu (hay nói cách khác là'trở thành một biệt hiệu ' hoặc 'trỏ tới') biến gốc .Những sự thay đổi này của biến mới làm ảnh hưởng đến biến gốc và ngược lại . Điều này cũng có nghĩa là sẽ không có một sự sao chép nào cả ,do vậy sự gán trị xảy ra rất nhanh .Tuy nhiên bất cứ sự tăng tốc nào cũng có thể được lưu ý là chỉ trong những vòng lặp kín hoặc khi gán những mảng lớn cho các đối tượng.

Để gán trị bằng cách tham chiếu ,một cách đơn giản hãy để một dấu & trước biến mà đang được gán (biến nguồn) Ví dụ đoạn mã sau đây cho xuất hiện chuỗi 'My name is Bob' hai lần :

```
<?php
$foo = 'Bob';              // Assign the value 'Bob' to $foo
$bar = &$foo;              // Reference $foo via $bar.
$bar = "My name is $bar";  // Alter $bar...
echo $foo;                 // $foo is altered too.
echo $bar;
?>
```

Cần lưu ý một điều quan trọng là chỉ có tên của các biến mới có thể được gán bởi tham chiếu.

```
<?php
```

```
$foo = 25;
$bar = &$foo;      // This is a valid assignment.
$bar = &(24 * 7);  // Invalid; references an unnamed expression.

function test() {
  return 25;
}

$bar = &test();    // Invalid.
?>
```

2_Các biến được định nghĩa trước

PHP cung cấp một số lượng lớn các biến số được định nghĩa trước cho bất cứ script nào mà nó chạy.Rất nhiều các biến này ,tuy nhiên , không thể chứng minh một cách đầy đủ rằng chúng phụ thuộc trên những server đang chạy ,phiên bản và thiết lập của một server và các nhân tố khác .Một vài những biến này sẽ không sẵn sàng khi PHP chạy trên dòng lệnh .
Bất chấp những nhân tố bày , đây là danh sách những biến đã được định nghĩa trước có sắn dưới một quá trình cài đặt sẵn có của PHP 3 chạy như một môđun dưới trình cài đặt sẵn có của Apache 1.3.6.
Để có danh sách những biến được định nghĩa trước này (và rất nhiều các thông tin hữu ích khác) hãy xem (và sử dụng) Phpinfo().
Chú ý : Danh sách này không có hết cả những khía cạnh mà bạn mong chờ .Nó chỉ đơn giản là một nguyên tắc dẫn đến các trình tự gì của những biến được định nghĩa trước mà bạn có thể mong chờ để truy cập vào script của ta.

2.1_Các biến Apache

Những biến này được tạo ra bởi Apache webserver.Nếu bạn đang chạy một webserver nào khác sẽ không có sự đảm bảo rằng nó sẽ được cung cấp những biến tương tự như thế ,nó có thể bỏ qua một số hoặc cung cấp những biến khác không được liệt kê ổ đây . Điều đó nói rằng một số lượng lớn những biến này được giải thích trong CGI 1.1 ,do đó bạn có thể mong đợi những cái này .

GATEWAY_INTERFACE

Xem lại điều gì của CGI chỉ rõ server đang sử dụng, ví dụ ‘CGI/1.1’

SERVER_NAME

Tên của server host dưới đó script hiện tại đang thực hiện. Nếu script đang chạy trên một host ảo, điều này sẽ là giá trị được định nghĩa cho host ảo đó.

SERVER_SOFWARE

Chuỗi nhận định server, được lấy ra trong các phần đầu phản hồi các yêu cầu.

SERVER_PROTOCOL

Tên và xem lại các thông tin giao thức thông qua đó trang web đã được yêu cầu, ví dụ ‘HTTP/1.0’;

REQUEST_METHOD

Phương thức yêu cầu nào đã đượ dùng để truy cập vào trang web; ví dụ: ‘GET’,’HEAD’,’P ÓT’,’PUT’.

QUERY_STRING

Thư mục tài liệu gốc dưới đó script hiện tại đang chạy, được định nghĩa trong file cấu hình của server.

HTTP_ACCEPT

Các nội dung của Accept-Charset: Phần đầu của yêu cầu hiện tại, nếu có một yêu cầu. Ví dụ: 'iso-8859-1,*,utf-8'.

HTTP_ACCEPT_ENCODING

Các nội dung của Accept-Encoding: Phần đầu của yêu cầu hiện tại, nếu có nó
Ví dụ: ‘gzip’

HTTP_ACCEPT_LANGUAGE

Các nội dung của Accept-Languagẻ: Phần đầu của yêu cầu hiện tại, nếu có nó, ví dụ:’en’

HTTP_CONNECTION

Các nội dung của Connection: Phần đầu từ yêu cầu hiện tại, nếu có nó
Ví dụ: ‘Keep-Alice’.

HTTP_HOST

Các nôi dung của Host : Phần đầu của trang web hiện tại, nếu có nó. HTTP_REFERER
Địa chỉ của trang được quy cho bộ trình duyệt tới trang hiện tại. Điều này được thiết lập bởi bộ trình duyệt của người sử dụng, không pahỉ tất cả các trình duyệt sẽ được thiết lập hư thế này.

HTTP_USER_AGENT
REMOTE_ADDR
REMOTE_PORT
SCRIPT_FILENAME
SERVER_ADMIN
SERVER_PORT
SERVER_SIGNATURE
PATH_TRANSLATED
SCRIPT_NAME
REQUEST_URI

Nội dung của các biến này ta có thể tra cứu trong PHP Manual.

2.2_Các biến môi trường

Các biến môi trường này được phập vào trong vùng đặt tên toàn cục của PHP từ môi trường dưới đó PHP đang chạy. Có rất nhiều thứ được cung cấp bởi nền shell nơi mà PHP đang chạy và các hệ thống khác nhau cũng đang chạy trên các shell khác nhau,một danh sách cuối cùng là không thể .

Các biến môi trường khác bao gồm các biến CGI, được đặt tại đó mà không chú ý đến việc PHP đang chạy như là một module hoặc bộ xử lý CGI.

Các biến PHP

Các biến được tạo bởi chính bản thân PHP. Các biến $HTTP_*_VARS chỉ có sẵn nếu cấu hình track_vars được bật nên. Khi đã được bật nên, các biến thông thường được xác lập, ngay cả khi chúng là các mảng trống. Điều này ngăn ngừa những người có ác ý muốn lừa các biến này.

Chú ý: Giống như PHP 4.0.3, track_vars luôn luôn được mở, mặc cho việc thiết lập file cấu hình .

Nếu định hướng register_globals được thiết lập, khi đó các biến này cũng sẽ được tạo sắn trong phạm vi toàn cục của script, ví dụ, một vài từ các mảng $HTTP_*_VARS. Chức năng này sẽ được sử dụng trong trường hợp cẩn thận, và được tắt đi nếu không cần thiết; Trong khi các biến $HTTP_*_VARS được bảo vệ, các biến toàn cục trống tương đương có thể được ghi đè bởi dữ liệu nhập của người sử dụng, có lẽ vì mục đích hiểm độc. Nếu ta không thể tắt register_globals, ta phải thực hiện các bước nào là cần thiết đề đảm bảo dữ liệu ta đang sử dụng là an toàn.

Argv

Mảng các đối số được gửi tới script. Khi script được chạy trên dòng lệnh, điều này đưa ra kiểu mẫu C truy cập vào các tham số dòng lệnh. Khi được gọi thông qua phương thức GET, điều này sẽ chứa các chuỗi truy vấn.

Argc

Chứa một số lượng các tham số dòng lệnh được gửi tới script(nếu được chạy trên dòng lệnh).
PHP_SELF
HTTP_COOKIE_VARS
HTTP_GET_VARS
HTTP_POST_VARS
HTTP_POST_FILES
HTTP_ENV_VARS
HTTP_SERVER_VARS
Các biến này ta có thể tham khảo thêm trong PHP manual.

3_Phạm vi của biến

Phạm vi của biến là ngữ cảnh trong đó nó được định nghĩa. Cho hầu hết các phần toàn bộ các biến PHP chỉ có một phạm vi đơn. Phạm vi đơn này bao gồm mở rộng và các file được yêu cầu , ví dụ:

```
$a = 1;
include "b.inc";
```

Đây là biến $a sẽ có sẵn trong script included b.inc. Tuy nhiên, trong các hàm định nghĩa của người sử dụng một hàm phạm vi cục bộ được giới thiệu. Bất kỳ biến nào được dùng trong một hàm thì được mặc định giới hạn tới hàm phạm vi cục bộ. ví dụ:

```
$a = 1; /* global scope */

Function Test () {
   echo $a; /* reference to local scope variable */
}

Test ();
```

Script này sẽ không tạo ra bất kỳ đầu ra nào bởi vì cú pháp hiển thị được gán cho một phiên bản biến $a cục bộ, và nó không được gán trị trong phạm vi này. Ta có thể chú ý rằng đây là một điều khác biệt nhỏ từ ngôn ngữ C trong đó các biến toàn cục trong C được tự động gán trị tới các hàm nếu không được tự động hoá việc ghi đè bởi một định nghĩa cục bộ. Điều này có thể dẫn đến một vài vấn đề trong đó mọi người có thể tình cờ thay đổi một biến toàn cục. Trong PHP , các biến toàn cục phải được khai báo tổng thể bên trong một hàm nếu chúng chuẩn bị được dùng trong hàm đó. Ví dụ:

```
$a = 1;
$b = 2;

Function Sum () {
   global $a, $b;

   $b = $a + $b;
}

Sum ();
echo $b;
```

script trên sẽ cho đầu ra là “3”. Bằng cách khai báo biến tổng thể $a và $b bên trong hàm, toàn bộ các tham chiếu đến các biến đó sẽ được gán cho phiên bản cục bộ. Không có một giới hạn trong số lượng các biến toàn cục mà có thể được thao tác bởi một hàm.
Cách thứ hai để truy cập các biến từ phạm vi toàn cục là bằng cách sử dụng điểm đặc biệt của PHP - định nghĩa mảng $GLOBALS. Ví dụ trước có thể được viết lạ như sau:

```
$a = 1;
$b = 2;

Function Sum () {
```

```
   $GLOBALS["b"] = $GLOBALS["a"] + $GLOBALS["b"];
}

Sum ();
echo $b;
```
Mảng $GLOBALS là một mảng liên kết với tên biến cục bộ có khoá và các nội dung của biến đó có giá trị của phần tử của mảng.
Một đặc điểm quan trọng khác của phạm vi biến là biến tĩnh. Một biến tĩnh chỉ tồn tại trong một hàm phạm vi cục bộ, nhưng nó không làm mất giá trị của nó khi chương trình thực hiện việc thoát khỏi phạm vi này. Giống như ví dụ dưới đây:
```
Function Test () {
   $a = 0;
   echo $a;
   $a++;
}
```
Hàm này khá vô ích khi mỗi lần nó được gọi nó thiết lập $a giá trị 0 và in "0". $a++ tăng lên một đơn vị cho biến $a mỗi khi hàm được gọi. Để tạo một hàm đếm hữu dụng mà không bị mất giá trị hiện tại, biến $a được khai báo tĩnh:
```
Function Test () {
   static $a = 0;
   echo $a;
   $a++;
}
```
Bây giờ mỗi khi hàm Test()được gọi nó sẽ in giá trị của biến $a và tăng giá trị của nó.
Các biến tĩnh cũng cung cấp một cách để có liên quan đến các hàm đệ qui. Một hàm đệ qui là hàm mà nó gọi chính bản thân nó. Phải cẩn thận khi viết một hàm đệ qui bởi vì nó có thể tạo một lời gọi đệ qui vô hạn định. Ta phải chắn chắn rằng phải có một cách hợp lý để kết thúc lời gọi đệ qui. Ví dụ sau hàm sẽ được gọi đệ qui để đếm đến 10, sử dụng biến tĩnh $count để biết khi kết thúc.
```
Function Test () {
   static $count = 0;

   $count++;
   echo $count;
   if ($count < 10) {
      Test ();
   }
   $count--;
}
```
4_Các biến có thể thay đổi được

Đôi khi thật tiện lợi để có thể có tên các biến có thể thay đổi được. Điều đó có nghĩa là, một tên biến mà có thể được thiết lập và sử dụng một cách linh hoạt. Một biến thông thường được thiết lập với một câu lệnh như sau:
```
$a = "hello";
```
Một biến có thể thay đổi lấy giá trị của một biến và xem như là tên của một biến .Trong ví dụ trên, hello, có thể được sử dụng như là tên của một biến bằng cách sử dụng hai dấu $, ví dụ:
```
$$a = "world";
```
Tại điểm này hai biến đã được định nghĩa và được lưu trữ trong cây cú pháp PHP: $a với nội dung "hello" và $hello với nội dung "world". Bởi vậy, câu lệnh sau:
```
echo "$a ${$a}";
```
đưa ra đầu ra chính xác tương tự như:
```
echo "$a $hello";
```
cả hai ví dụ trên đều tạo ra:hello world

Trong trường hợp sử dụng các biến có thể thay đổi với các mảng, ta phải giải quyết một vấn đề mơ hồ. Đó là, nếu ta viết $$a[1] tiếp theo cú pháp cần để biết nếu ta có ý định dùng $a[1] như là một biến, hoặc nếu ta muốn $$a như là biến và tiếp đến [1] chỉ số từ biến đó. Cú pháp để giải quyết điều mập mờ này là: ${a[1] } trong trường hợp thứ nhất và ${$a}[1] cho trường hợp thứ hai.

5_Các biến từ bên ngoài PHP

5.1_Các form HTML(GET và POST)

Khi một form được trình tới một script của PHP, bất kỳ các biến từ form đó sẽ rự động được tạo thành có sẵn tới script bởi PHP. Nếu tuỳ chọn cấu hình track_vars được thiết lập là on, khi đó các biến này sẽ được xác định trong các mảng kết hợp $HTTP_POST_VARS,$HTTP_GET_VARS và/hoặc $HTTP_POST_FILES phù hợp với nguồn của biến trong câu hỏi.

Ví dụ: Biến form đơn giản:

```
<form action="foo.php" method="post">
   Name: <input type="text" name="username"><br>
   <input type="submit">
</form>
```

Khi form trên được đệ trình, giá trị từ đầu vào text sẽ được có hiệu lực trong $HTTP_POST_VARS[‘username’]. Nếu định hướng cấu hình register_globals được thiết lập là on, khi đó biến cũng sẽ có hiệu lực như $username trong phạm vi toàn cục.

PHP cũng hiểu các mảng trong ngữ cảnh của các biến form.Ta có thể, ví dụ, nhóm các biến liên quan với nhau, hoặc sử dụng chức năng này để nhận các giá trị từ nhiêu lựa chọn đầu vào:

Ví dụ: các biến form phức tạp hơn

```
<form action="array.php" method="post">
   Name: <input type="text" name="personal[name]"><br>
   Email: <input type="text" name="personal[email]"><br>
   Beer: <br>
   <select multiple name="beer[]">
      <option value="warthog">Warthog
      <option value="guinness">Guinness
      <option value="stuttgarter">Stuttgarter Schwabenbräu
      </select>
   <input type="submit">
</form>
```

Trong PHP3, cách dùng biến form mảng được giới hạn tới các mảng một chiều đơn. Trong PHP 4, không có một giới hạn nào được áp dụng.

5.2_Các tên biến IMAGE SUBMIT

Khi đệ trình một form, có thể sử dụng một hình ảnh thay vì nút submit chuẩn với một tag như sau:

```
<input type=image src="image.gif" name="sub">
```

Khi người sử dụng click vào một nơi nào đó trong hình ảnh, form kèm theo sẽ được chuyển tới server với hai biến thêm vào, sub_x và sub_y. Các biến này chứa các toa độ của người dùng click vào trong hình ảnh. Kinh nghiệm có thể cho ta thấy rằng các tên biến thật sự được gửi tới bởi trình duyệt chứa một dấu chấm câu còn hơn là một dấu gạch dưới chân, nhưng PHP tự động chuyển dấu chấm câu thành dấu gạch dưới.

5.3_HTTP Cookies

PHP hỗ trợ HTTP cookies một cách trong suốt như được định nghĩa bởi Netscape’s Spec. Cookies là một kỹ xảo cho việc lưu trữ dữ liệu trong các trình duyệt từ xa và theo đó theo

dõi hoặc nhận biết các người sử dụng trả về. Ta có thể thiết lập các cookies bằng cách dùng hàm SetCookies(). Các cookies là một phần của phần đầu HTTP, bởi vậy hàm SetCookie phải được gọi trước bất kỳ đầu ra được gửi tới trình duyệt. Đây cũng là giới hạn như là đối với hàm Header(). Bất kỳ cookies nào được gửi ta từ client sẽ đượctự động trả về một biến PHP giống như là phương thức GET và POST dữ liệu.
Nếu ta muốn gán nhiều giá trị tới một cookie đơn, chỉ cần thêm [ ] tới tên cookie. Ví dụ:

```
SetCookie ("MyCookie[]", "Testing", time()+3600);
```

Chú ý rằng một cookie sẽ thay thế một cookie trước đó bởi tên giống như trong trình duyệt của ta nếu đường dẫn hoặc domain là không khác nhau. Bởi vậy, đối với ứng dụng xe chứa hàng ta có lẽ cần phải giữ một quầy tính tiền và cho qua về phía này. Ví dụ:

```
$Count++;
SetCookie ("Count", $Count, time()+3600);
SetCookie ("Cart[$Count]", $item, time()+3600);
```

5.3_Các biến môi trường

PHP tạo các biến môi trườgn một cách tự động có sẵn như là các biến PHP thông thường.

```
echo $HOME;  /* Shows the HOME environment variable, if set. */
```

Từ khi thông tin được nhập vào thông qua thông qua các cơ chế GET,POST và Cookie cũng tạo các biến PHP một cách tự động, đôi khi nó cũng tốt nhất cho việc đọc một biến từ môi trường trong khi chắc chắn rằng ta đang lấy phiên bản đúng. Hàm getenv() có thể được dùng cho trườgn hợp này. Ta cũng có thể thiết lập một biến môi trường với hàm putenv().

5.4_Các điểm trong các tên biến thay thế

Thông thường, PHP không thực hiện thay đổi các tên của các biến khi chúng được đưa qua trong một script. Tuy nhiên, phải chú ý rằng dấu chấm không phải là một ký tự đúng trong môt tên biến PHP. Nguyên nhân ta có thể xem như sau:

```
$varname.ext;  /* invalid variable name */
```

Bây giờ cú pháp phân tích ta thấy là một biến được đánh tên là $varname, được theo sau bởi một chuỗi các thao tác liên tiếp giống nhau, được theo sau bởi chuỗi trống. Có thể thấy rằng điều này không cho kết quả như mong đợi.
Lý do của điều này đó là , điều quan trọng cần chú ý rằng PHP sẽ thay thế các dấu chấm trong các biến mới đến bằng các dấu gạch dưới một cách tự động.

5.5_Xác định các kiểu biến

Bởi vì PHP xác định các kiểu của các biến và chuyển đổi chúng (một cách thông thường) là cấn thiết, nó không thường xuyên chỉ ra một cách rành mạch kiểu gì của một biến đã đưa ra tại bất kỳ một thời điểm nào. PHP bao gồm một vài hàm chỉ ra kiểu của một biến. Chúng là các hàm gettype(),is_long(),is_double(),is_string(),is_array(),is_object().

IV_CÁC HẰNG SỐ

PHP định nghĩa một số hằng số và cung cấp một cơ chế cho việc định nghĩa nhiều hơn tại thời gian chạy. Các hằng số cũng khá giống các biến số, các hằng phải được định nghĩa bằng việc sử dụng hàm define(), và chúng không thể được định nghĩa lại sau này tới giá trị khác.
Các hằng được tiền định nghĩa (luôn luôn có hiệu lực) là:

```
_FILE_
```

Tên của file script được tách ngay tức thì. Nếu được dùng trong một file mà được bao gồm hoặc được đòi hỏi, khi đó tên của file bao gồm được căn cứ vào, và không cùng tên với tên của file nguồn.
LINE
Số lượng dòng trong file script hiện tại đang được tách ra. Nếu được dùng trong một file mà được bao gồm hoặc được yêu cầu, khi đó vị trí của fie bao gồm được căn cứ vào.
PHP_VERSION
Chuỗi mô tả phiên bản của PHP hiện đang được dùng.
PHP_OS
Tên của hệ điều hành trên đó PHP đang thực hiện.
TRUE
Một giá trị thực tế
FALSE
Một giá trị sai
E_ERRO
Thể hiện một lỗi khác với một lỗi cú pháp từ đó mà khả năng lấy lại được là không thể.
E_WARNING
Thể hiện một điều kiện mà PHP biết một điều gì đó bị sai, nhưng sẽ không tiếp tục được bằng bất kỳ cách nào, các sự việc này có thể được bắt bởi chính bản thân script.
Một vài biến môi trường khác ta có thể tra trong PHP Manual như:
E_PARSE
E_NOTICE
E_ALL
Ví dụ về việc định nghĩa các hằng số:

```
<?php
define("CONSTANT", "Hello world.");
echo CONSTANT; // outputs "Hello world."
?>
```

Sử dụng _FILE_ và _LINE_

```
<?php
function report_error($file, $line, $message) {
   echo "An error occured in $file on line $line: $message.";
}

report_error(__FILE__,__LINE__, "Something went wrong!");
?>
```

V_CÁC BIỂU THỨC

Các biểu thức là các nền tảng quan trọng nhất của PHP.Trong PHP, hầu như cái gì ta viết đều là biểu thức. Cách đơn giản và chính xác nhất để định nghĩa một biểu thức là " bất kỳ cái gì đều có một giá trị".
Những kiểu cách đơn giản nhất của các biểu thức là các hàm và các biến. Khi ta gõ "$a=5", ta đang gán '5' cho $a. Có thể thấy '5'có giá trị 5, hoặc nói cách khác '5' là một biểu thức với giá trị của 5 (trong trường hợp này, '5' là một hằng số nguyên).
Sau sự gán này, ta thấy rằng giá trị của $a sẽ là 5, bởi vậy nếu ta viết $b=$a, ta sẽ cho rằng kết qủa cũng tương tự như khi ta viết $b=5. Nói cách khác, $a là một biểu thức với giá trị của 5. Nếu mọi việc đều đúng, điều này là chính xác điều gì sẽ xảy ra.
Các ví dụ hơi phức tạp hơn một chút của các biểu thức là các hàm. Ví dụ, ta hãy xem ví dụ sau:

```
function foo () {
   return 5;
}
```

Cho rằng bạn đã biết qua về khái niệm của các hàm, bạn sẽ cho rằng khi gõ $c=foo() thì cũng giống như viết $c=5, và bạn cũng đã đúng. Các hàm là các biểu thức với giá trị là giá trị được trả về của chúng. Khi foo() trả về giá trị 5, giá trị của biểu thức 'foo()' là 5. Thông thường không chỉ trả về một giá trị tĩnh ngoài ra còn tính toán một thứ gì đó.
Dĩ nhiên là các giá trị trong PHP không phải là các số nguyên. PHP hỗ trợ 3 kiểu giá trị vô hướng: các giá trị số nguyên,các giá trị con trỏ động và các giá trị chuỗi (Các giá trị vô hướng là các giá trị mà ta không thể "bẻ" thành các các thành phần nhỏ hơn, không như các mảng). PHP cũng hỗ trợ hai kiểu phức: các mảng và các đối tượng. Một trong các kiểu giá trị này có thể được gán vào trong các biến hoặc được trả về từ các hàm.
Trước đó, người sử dụng của PHP/FI 2 sẽ không cảm nhận được bất kỳ thay đổi nào. Tuy vậy, PHP đạt được các biểu hiện nhiều hơn trong cùng một cách mà nhiều ngôn ngữ khác đã làm. PHP là một ngôn ngữ hướng biểu thức, trong sự cảm nhận rằng mọi thứ đều là biểu thức. Hãy để ý đến ví dụ trước đây, '$a=5'. Thật dễ dàng để thấy rằng có hai giá trị được gọi lên từ đây, giá trị của hằng số nguyên '5', và giá trị của $a mà đang được cập nhật tới 5. Nhưng sự thật là chỉ có một giá trị thêm vào được gọi ra ở đây, và đó là giá trị của sự gán trị của chính bản thân nó . Sự gán trị kiểu này ước lượng tới giá trị được gán, trong trường hợp này là 5. Trong thực tiễn, nó có ý nghĩa là '$a=5', không chú ý đến gì nó thực hiện, là một biểu thức với giá trị 5. Do đó, viết một thứ gì đó như '$b=($a=5)' cũng giống như viết '$a=5;$b=5;') (một dấu hai chấm đánh dấu sự kết thúc của một câu lệnh). Ta cũng có thể thực hiện việc gán trị như sau:'$b=$a=5'.
Một ví dụ tốt khác của sự hướng biểu thức là các toán tử tăng và giảm. Người sử dụng của PHP/FI 2 và rất nhiều ngôn ngữ khác có thể quen với kí hiệu biến++ và biến--. Đây là các toán tử tăng và giảm. Trong PHP/FI 2, cú pháp '$a++' không có giá trị (không phải là một biểu thức), và vì vậy ta không thể gán nó và sử dụng nó trong bất kỳ trường hợp nào.PHP nâng cao các khả năng tăng và giảm bằng cách tạo các biểu thức này, tương tự như C. Trong PHP, giống như C, có hai dạng tăng tiền tố và tăng hậu tố. Cả hai cách này về bản chất làm tăng giá trị của biến và ảnh hưởng tới các biến là như nhau. Sự khác nhau là với giá trị cuả biểu thức tăng. Dạng tiền tố, được viết '++$biến', để xác định giá trị của biến đã được tăng (PHP tăng biến trước khi đọc giá trị của nó). Dạng hậu tố, được viết như 'biến++' xác định tới giá trị thông thường của $biến, trước khi nó được tăng(PHP tăng biến sau khi đọc giá trị của nó).
Một dạng thông dụng nhất của các biển thức là các biểu thức so sánh. Các biểu thức này xác định tới 0 hoặc 1,nghĩa là tương ứng với FALSE hoặc TRUE. PHP hỗ trợ > (lớn hơn), >= (lớn hơn hoặc bằng), = (bằng), != (khác), < (nhỏ hơn), <= (nhỏ hơn hoặc bằng). Các biểu thức này được sử dụng một cách thông dụng nhất bên cạnh sự thực hiện các điều kiện, như là cấu trúc điều kiện if.
Ví dụ cuối cùng của các biểu thức chúng ta sẽ đề cập đến ở đây được kết hợp với các biểu thức gán toán tử. Ta thật sự biết rằng nếu ta muốn tăng $a lên 1, ta có thể đơn giản viết '$a++' hoặc '++$a'. Nhưng nếu ta muốn cộng thêm hơn một lần cho nó, ví dụ như 3 ?
Ta có thể viết '$a++' thành nhiều lần nhưng điều này là không hợp lý và tỏ ra bất tiện. Cách thông dụng nhất là viết '$a=$a+3'. '$a+3' xác định tới giá trị của $a cộng 3, và được gán trở lại trong biến $a. Trong PHP, cũng như trong một số ngôn ngữ giống C, ta có thể viết tương tự như vậy trong một cách ngắn hơn, nhưng lại sáng sủa hơn và dễ hiểu hơn. Cộng thêm 3 vào giá trị hiện tại của $a có thể được viết '$a+=3'. Điều này thực sự chính xác là" lấy giá trị của $a, cộng thêm 3, và gán trở lại $a".Ngoài việc cách viết này vừa ngắn vừa dễ hiểu, nó còn được thực hiện nhanh hơn. Giá trị của '$a+=3', giống như giá trị của một phép gán thông thường, là giá trị được gán. Chú ý rằng đây không phải là 3, mà là giá trị kết hợp của $a cộng 3(Đây là giá trị được gán vào trong biến $a).
Có hơn một biểu thức có thể xem là kỳ quặc nếu ta không thấy điều này trong các ngông ngữ khác, toán tử 3 yếu tố:
$first ? $second : $third

Nếu giá trị của biểu thức con đầu tiên là đúng(khác không), khi đó biểu thức thứ hai sẽ được định giá cho nó, và đó là kết qủa của biểu thức điều kiện. Trong trường hợp khác, biểu thức nhỏ thứ 3 sẽ được định giá, và đo cũng là giá trị của nó.
Ví dụ sau sẽ giúp ta hiểu các biểu thức tăng trước và tăng sau trong một trường hợp khá thông thường:

```
function double($i) {
   return $i*2;
}
$b = $a = 5;        /* assign the value five into the variable $a and $b */
$c = $a++;          /* post-increment, assign original value of $a
                    (5) to $c */
$e = $d = ++$b;     /* pre-increment, assign the incremented value of
                    $b (6) to $d and $e */

/* at this point, both $d and $e are equal to 6 */

$f = double($d++);  /* assign twice the value of $d before
                    the increment, 2*6 = 12 to $f */
$g = double(++$e);  /* assign twice the value of $e after
                    the increment, 2*7 = 14 to $g */
$h = $g += 10;      /* first, $g is incremented by 10 and ends with the
                    value of 24. the value of the assignment (24) is
                    then assigned into $h, and $h ends with the value
                    of 24 as well. */
```

Trong phần đầu của chương chúng ta đã nói rằng chúng ta sẽ thảo luận về các loại câu lệnh khác nhau, và như đã hứa, các biểu thức có thể là các câu lệnh. Tuy vậy, không phải mỗi biểu thức là một câu lệnh. Trong trường hợp này, một câu lệnh có dạng là một biểu thức, điều này có nghĩa là, một biểu thức được theo sau bởi dấu chấm phẩy. Trong '$b=$a=5;', $a =5 là một biểu thức đúng, nhưng chính bản thân nó không phải là một câu lệnh. Tuy nhiên '$b=$a=5;' là một câu lệnh đúng.
Điều cuối cùng đáng để nói đến là giá trị thực sự của các biểu thức. Trong rất nhiều sự kiện, chủ yếu trong việc thực hiện các điều kiện và các vòng lặp, ta không thích thú gì trong giá trị cụ thể của biểu thức, nhưng chỉ chú ý đến việc giá trị của chúng là TRUE hay FALSE (PHP không cung cấp một kiểu logic). Giá trị thực sự của các biểu thức trong PHP được tính với cách khá giống với cách trong Perl. Bất kỳ giá trị khác không đều là TRUE, các giá trị bằng không là FALSE. Hãy thận trọng rằng các giá trị âm là khác không và vì vậy nó có giá trị TRUE. Chuỗi trống và chuỗi "0" là FALSE; Các chuỗi còn lại khác là TRUE. Với các giá trị không vô hướng (các mảng và các đối tượng) - Nếu giá trị không chứa các phần tử nó được xem như FALSE, các trường hợp khác là TRUE.
PHP cung cấp một sự bổ sung đầy đủ và mạnh mẽ của các biểu thức, và điều này được chứng minh trong tài liệu PHP Manual này.

V_CÁC TOÁN TỬ

1_Các toán tử số học

Ví dụ	Tên	Kết quả
$a + $b	Phép cộng	Tổng $a và $b.
$a - $b	Phép trừ	Hiệu $a và $b.
$a * $b	Phép nhân	Tích số $a và $b.
$a / $b	Phép chia	Thương số $a và $b.
$a % $b	Phép chia dư	Số dư của $a chia cho $b.

Toán tử chia ("/") trả lại một giá trị số nguyên (Kết quả của một phép chia số nguyên) nếu hai toán tử là các số nguyên(hoặc các chuỗi được chuyển đổi thành các số nguyên) và thương số là một số nguyên. Nếu một toán hạng là một giá trị số thực động, hoặc các kết quả phép toán không phải là một số nguyên, giá trị số thực động được trả về.

2_Các toán tử gán

Toán tử gán cơ bản là "=".Ý nghĩ trước tiên của ta có thể cho rằng điều này là " bằng với". Không phải như vậy. Nó thật sự là toán tử phía bên trái lấy giá trị của biểu thức bên phải.

Giá trị của một biểu thức gán là một giá trị được gán. Điều đó có nghĩa là, giá trị của "$a=3" là 3. Điều này cho phép ta thực hiện một số việc phức tạp:

$a = ($b = 4) + 5; // $a is equal to 9 now, and $b has been set to 4

Trong phép cộng với toán tử gán, có sự kết hợp các toán tử cho tất cả các số học nhị phân và các toán tử chuỗi cho phép ta dùng một giá trị trong một biểu thức và tiếp theo thiết lập giá trị của nó tới kết quả của biểu thức đó. Ví dụ:

```
$a = 3;
$a += 5; // sets $a to 8, as if we had said: $a = $a + 5;
$b = "Hello ";

$b .= "There!"; // sets $b to "Hello There!", just like $b = $b . "There!";
```

Chú ý rằng phép gán sao chép biến thông thường tới một biến mới(Phép gán bởi giá trị), bởi vậy thay đổi tới một biến sẽ không ảnh hưởng đến các biến còn lại. Điều này có thể cũng thích hợp nếu ta cần sao chép một thứ gì đó giống như một mảng lớn trong một vòng lặp vô hạn. PHP 4 hỗ trợ phép gán bằng tham chiếu, sử dụng $biến = &$ biến khác; Nhưng cú pháp này không cho phép trong PHP3.'Ph ép gán bằng tham chiếu' có nghĩa các biến cùng chỉ vào một giá trị giống nhau, và không có một sự sao chép từ bất kỳ đâu.

3_Các toán tử mức độ bit

Example	Name	Result
$a & $b	And	Bits that are set in both $a and $b are set.
$a | $b	Or	Các bit được lập trong trường hợp $a hoặc $b được lập.
$a ^ $b	Xor	Các bit được lập khi $a hoặc $b được lập nhưng cả hai không đồng thời lập.
~ $a	Not	Các bit được lập nếu $a không được lập và ngược lại.
$a << $b	Shift left	Dịch các bit của $a sang trái $b vị trí(mỗi bước dịch tương đương việc nhân với 2).
$a >> $b	Shift right	Dịch các bit của $a sang phải $b vị trí (mỗi bước dịch tương đương với việc chia cho 2).

4_Các toán tử so sánh

Các toán tử so sánh, như tên của chúng ngụ ý, cho phép ta so sánh hai giá trị

Ví dụ	Tên	Kết quả
$a == $b	Bằng	Đúng nếu $a bằng với $b.
$a === $b	Đồng nhất	Đúng nếu $a bằng với $b , nhưng chúng không cùng kiểu (Chỉ có ở PHP4).
$a != $b	Khác	Đúng nếu $a khác $b
$a !== $b	Không đồng nhất	Đúng nếu $a khác $b hoặc chúng không cùng kiểu
$a < $b	Nhỏ hơn	Đúng nếu $a hoàn toàn nhỏ hơn $b
$a > $b	Lớn hơn	Đúng nếu $a hoàn toàn lớn hơn $b
$a <= $b	Nhỏ hơn hoặc bằng	Đúng nếu $a nhỏ hơn hoặc bằng $b
$a >= $b	Lớn hơn hoặc bằng	Đúng nếu $a lớn hơn hoặc bằng $b

Một toán tử điều kiện khác là toán tử ”?” nó thao tác giống như trong C và rất nhiều ngôn ngữ khác.

(expr1) ? (expr2) : (expr3);

Biểu thức này ước lượng với expr2 nếu expr1 ước lượng là đúng, và expr3 nếu expr1 ước lượng là sai.

5_Các toán tử điều khiển lỗi

PHP hỗ trợ một toán tử điều khiển lỗi: dấu at (@). Khi đã chú ý đến một biểu thức trong PHP, bất kỳ thông báo lỗi nào mà có thể được tạo ra bởi biểu thức đó sẽ bị từ chối.

Nếu chức năng track_errors được kích hoạt, bất kỳ một lỗi nào được phát sinh bởi biểu thức sẽ được ghi lại trong một biến cục bộ $php_errormsg. Biến này sẽ đợc ghi đè sau mỗi lần bị lỗi, bởi vậy hãy kiểm tra một cách sớm nếu ta muốn sử dụng nó.

```
<?php
/* Intentional SQL error (extra quote): */
$res = @mysql_query ("select name, code from 'namelist") or
   die ("Query failed: error was '$php_errormsg'");

?>
```

Cảnh báo
Tiền tố toán tử điều khiển lỗi "@"hiện tại sẽ vẫn không có thể báo cáo lỗi cho các lỗi nguy cấp làm ngưng sự thực thi script. Nói cách khác, điều này có nghĩa là nếu ta dùng "@"để tiêu diệt các lỗi từ một hàm nào đó và nó cũng không sẵn có hoặc đã bị vô hiệu hoá, script sẽ bị "chết" mà không có bất kỳ một hiển thị là tại sao.

6_Các toán tử thực thi

PHP cung cấp một toán tử thực thi: Hai dấu nháy kép phía dưới ký tự ~ (``). Chú ý rằng không phải là một dấu nháy đơn !. PHP sẽ cố gắng để thực hiện các nội dung của hai dấu nháy như là một lệnh shell; Kết qủa đầu ra sẽ được trả về(nó không chỉ đơn giản là đưa ra một kết quả cho đầu ra, nó có thể được gán tới một biến).

```
$output = `ls -al`;

echo "<pre>$output</pre>";
```

Chú ý: Toán tử nháy đơn sẽ mất tác dụng khi safe_mode được thiết lập.

7_Các toán tử tăng / giảm

PHP cung cấp kiểu cách của ngôn ngữ C là các toán tử phía trước và phía sau để tăng hay giảm giá trị của một biến.

Các toán tử tăng/giảm

Ví dụ	Tên	Kết quả
++$a	Tăng trước	Tăng $a lên 1, tiếp đến trả lại giá trị cho $a.
$a++	Tăng sau	Trả giá trị cho $a, sau đó mới tăng $a lên 1 .
--$a	Giảm trước	Giảm $a một đơn vị, sau đó mới trả lại giá trị.
$a--	Giảm sau	Trả giá trị cho $a, sau đó mới giảm giá trị của $a.

Sau đây là một script đơn giản:

```
<?php
echo "<h3>Postincrement</h3>";
$a = 5;
echo "Should be 5: " . $a++ . "<br>\n";
echo "Should be 6: " . $a . "<br>\n";

echo "<h3>Preincrement</h3>";
$a = 5;
echo "Should be 6: " . ++$a . "<br>\n";
echo "Should be 6: " . $a . "<br>\n";

echo "<h3>Postdecrement</h3>";
$a = 5;
echo "Should be 5: " . $a-- . "<br>\n";
echo "Should be 4: " . $a . "<br>\n";

echo "<h3>Predecrement</h3>";
$a = 5;
echo "Should be 4: " . --$a . "<br>\n";
echo "Should be 4: " . $a . "<br>\n";
?>
```

8_Các toán tử logic

Các toán tử logic:

Ví dụ	Tên	Kết quả
$a and $b	And	Đúng nếu cả $a và $b đều đúng.
$a or $b	Or	Đúng nếu $a hoặc $b đúng.
$a xor $b	Xor	Đúng nếu chỉ có $a hoặc chỉ có $b đúng.
! $a	Not	Đúng nếu $a sai.
$a && $b	And	Đúng nếu cả $a và $b đều đúng.
$a || $b	Or	Đúng nếu $a hoặc $b đúng.

9_ Quyền ưu tiên của các toán tử

Quyền ưu tiên của một toán tử chỉ ra băng cách nào ràng buộc hai biểu thức với nhau. Ví dụ, trong biểu thức 1+5*3, câu trả lời là 16 và không phải là 18 bởi vì toán tử nhân (“ * ”) có một quyền ưu tiên cao hơn đối với toán tử cộng

(“ + “). Các dấu ngoặc đơn có thể được thực hiện để chỉ ra quyền ưu tiên nếu thấy cần thiết. Ví dụ (1+5)*3 sẽ cho giá trị 18.

Bảng sau chỉ ra quyền ưu tiên của các toán tử với các toán tử có quyền ưu tiên thấp nhất được liệt kê trước.

Quyền ưu tiên các toán tử

Kết hợp	Các toán tử
Trái	,
Trái	or
Trái	xor
Trái	and
Phải	print
Trái	= += -= *= /= .= %= &= |= ^= ~= <<= >>=
Trái	? :
Trái	||
Trái	&&
Trái	|
Trái	^
Trái	&
Không kết hợp	== != === !==
Không kết hợp	< <= > >=
Trái	<< >>
Trái	+ - .
Trái	* / %
Phải	! ~ ++ -- (int) (double) (string) (array) (object) @
Phải	[
Không kết hợp	new

10_Các toán tử chuỗi

Có hai toán tử chuỗi. Toán tử đầu tiên là toán tử ghép (‘ . ‘), nó trả về sự ghép các đối số của bên trái và bên phải của nó. Toán tử thứ hai là toán tử ghép gán (‘ .= ‘), cho phép nối thêm đối số phía bên phải vào đối số phía bên trái của nó.

```
$a = "Hello ";
$b = $a . "World!"; // now $b contains "Hello World!"

$a = "Hello ";

$a .= "World!";     // now $a contains "Hello World!"
```

VII_CÁC CẤU TRÚC ĐIỀU KHIỂN

1_Cấu trúc If

Cấu trúc if là một trong các đặc điểm quan trọng nhất trong rất nhiều ngôn ngữ, PHP cũng có cấu trúc này. Nó cho phép thực thi có điều kiện các đoạn mã. PHP đề cao một cấu trúc if tương tự như C.

```
if (expr)
   statement
```

Như đã đề cập trong phần các biểu thức , expr được định giá tới chính giá trị thực sự của nó. Nếu expr định giá là TRUE, PHP sẽ thực hiện câu lệnh, và nếu nó định giá là FALSE - PHP sẽ bỏ qua nó.

Ví dụ sau sẽ hiển thị *a is big than b* nếu $a lớn hơn $b:

```
if ($a > $b)
   print "a is bigger than b";
```

Thông thường ta muốn có nhiều hơn một câu lệnh để được thực hiện một cách có điều kiện. Dĩ nhiên, không cần bao bọc mỗi câu lệnh với một mệnh đề if. Thay vào đó, ta có thể nhóm một vài câu lệnh vào trong một nhóm câu lệnh . Ví dụ, đoạn mã sau sẽ hiển thị *a is bigger than b* nếu $a lớn hơn $b, và sẽ gán giá trị của biến $a sang cho biến $b:

```
if ($a > $b) {
   print "a is bigger than b";
   $b = $a;

}
```

Các câu lệnh if có thể được lồng vào nhau sẽ cung cấp cho ta một cách toàn diện và linh hoạt cho việc thực hiện điều kiện nhiều phần khác nhau trong chương trình của ta.

2_Cấu trúc Else

Thông thường ta muốn thực hiện một câu lệnh nếu gặp phải một điều kiện chắc chắn nào đó,và một câu lệnh khác nếu điều kiện này là sai. Điều này cần dùng đến *else* . *else* mở rộng một câu lệnh *if* để thực hiện một câu lệnh trong trường hợp biểu thức trong câu lệnh if có giá trị là FALSE. Ví dụ, đoạn mã sau sẽ hiển thị *a is bigger than b* nếu $a lớn hơn $b và *a is not bigger than b* trong trường hợp ngược lại.

```
if ($a > $b) {
   print "a is bigger than b";
} else {
   print "a is NOT bigger than b";

}
```

Câu lệnh *else* chỉ được thực hiện trong trường hợp biểu thức trong câu lệnh if nhận giá trị FALSE.

3_Cấu trúc Elseif

Elseif, như tên của nó đã gợi ý, là một sự kết nối của if và else. Giống như else, nó mở rộng một câu lệnh if để thực hiện một câu lệnh khác trong trường hợp biểu thức if thông thường nhận giá trị FALSE. Tuy nhiên, không giống như else, nó sẽ chỉ thực hiện biểu thức loại trừ lẫn nhau nếu biểu thức điều kiện elseif nhận giá trị TRUE. Ví dụ, đoạn mã sau đây sẽ hiển thị *a is bigger than b, a equal to b* hoặc *a is smaller than b*:

```
if ($a > $b) {
   print "a is bigger than b";
} elseif ($a == $b) {
   print "a is equal to b";
} else {
   print "a is smaller than b";

}
```

Có thể có vài elseif trong cùng một câu lệnh if. Biểu thức elseif đầu tiên (nếu có) nhận giá trị TRUE sẽ được thực hiện. Trong PHP, ta cũng có thể viết 'else if (trong hai từ) và cũng cho giá trị tương đương như khi ta viết gộp trong một từ 'elseif'. Ý nghĩa cú pháp chỉ khác nhau một chút (nếu ta quen với C, điều này là tương tự) nhưng dòng trên cũng có cùng giá trị chính xác.

Câu lệnh elseif chỉ được thực hiện nếu biểu thức if trước đó và bất kỳ các biểu thức elseif nhận giá trị FALSE, và biểu thức elseif hiện tại nhận giá trị TRUE.

4_Cú pháp lựa chọn cho các cấu trúc điều khiển

PHP cũng cung cấp một cú pháp lựa chọn cho một vài cấu trúc điều khiển của nó; có tên là , if, while,for, foreach, và switch. Trong mỗi trường hợp, khuôn dạng cơ bản của cú pháp lựa chọn là để thay đổi việc mở ngoặc cho một dấu hai chấm (:) và việc đóng ngoặc cho endif;, endwhile;,endfor;, endeachfor;, hoặc endswitch .

```
<?php if ($a == 5): ?>
 A is equal to 5

 <?php endif; ?>
```

Trong ví dụ trên, khối HTML "A=5" được đặt trong một câu lệnh if được viết trong cú pháp lựa chọn. Khối HTML sẽ chỉ được hiển thị nếu $a =5.

Cú pháp lựa chọn được áp dụng cho cả else và elseif. Sau đây là một cấu trúc if với elseif và else trong định dạng lựa chọn:

```
if ($a == 5):
   print "a equals 5";
   print "...";
elseif ($a == 6):
   print "a equals 6";
   print "!!!";
```

```
else:
   print "a is neither 5 nor 6";
```

endif;

5_Cấu trúc while

Các vòng lặp while là một dạng lặp đơn giản nhất trong PHP. Chúng giống hệt trong C. Dạng cơ bản của câu lệnh while là:

while (expr) statement

Ý nghĩa của câu lệnh while thật đơn giản. Nó nói cho PHP thực hiện các câu lệnh lặp đi lặp lại lồng nhau, khi biểu thức while nhận giá trị TRUE. Giá trị của của biểu thức được kiểm tra mỗi lần tại đầu vòng lặp, bởi vậy nếu giá trị này thay đổi trong quá trình thực hiện các câu lệnh, quá trình thực hiện sẽ không dừng cho đến khi kết thúc sự lặp lại (mỗi lần PHP chạy các câu lệnh trong vòng lặp là một sự lặp lại). Thỉnh thoảng nếu biểu thức while nhận giá trị FALSE từ khi bắt đầu, các câu lệnh bên trong sẽ không được thực hiện.

Giống như câu lệnh if, ta có thể nhóm nhiều câu lệnh trong cùng một vòng lặp while bằng cách bao nhóm các câu lệnh trong các dấu móc, hoặc bằng cú pháp lựa chọn:

while (expr): statement ... endwhile;

Các ví dụ sau là y hệt nhau, và đều in các số từ 1 đến 10:

```
/* example 1 */

$i = 1;
while ($i <= 10) {
   print $i++;  /* the printed value would be
                   $i before the increment
                   (post-increment) */
}

/* example 2 */

$i = 1;
while ($i <= 10):
   print $i;
   $i++;
endwhile;
```

6_do .. while

Các vòng lặp do .. while cũng rất giống với các vòng lặp while, trừ việc kiểm tra điều kiện đúng tại lúc kết thúc mỗi vòng lặp thay vì ngay ban đầu. Sự khác biệt chính với các vòng lặp while thông thường là sự lặp lại đầu tiên của một vòng lặp do .. while được bảo đảm thực hiện (điều kiện đúng chỉ được chỉ ra tại cuối vòng lặp), trong khi nó có thể sẽ không được thực hiện trong vòng lặp while nếu ngay từ đầu điều kiện kiểm tra là FALSE.
Chỉ có một cú pháp cho các vòng lặp do .. while:

```
$i = 0;
do {
```

```
   print $i;
} while ($i>0);
```
Vòng lặp trên sẽ chỉ chạy một lần chính xác, sau lần lặp đầu tiên, khi biểu thức điều kiện được kiểm tra, nó xác định giá trị FALSE ($i không lớn hơn 0) và sự thực hiện vòng lặp được kết thúc. Những người dùng C cao cấp có thể quen với một cách sử dụng khác của vòng lặp do .. while, để cho phép dừng sự thực hiện tại giữa khối mã lệnh, bằng cách gói gọn chúng với do .. while(0), và sử dụng cú pháp break. Đoạn mã dưới đây mô tả điều này:
```
do {
   if ($i < 5) {
      print "i is not big enough";
      break;
   }
   $i *= $factor;
   if ($i < $minimum_limit) {
      break;
   }
   print "i is ok";

    ...process i...

} while(0);
```
Đừng lo lắng nếu ta không hiểu , ta có thể viết mã các script và cả các script mạnh mẽ mà không dùng đặc điểm này.

7_Cấu trúc For

Các vòng lặp for là các vòng lặp phức tạp nhất trong PHP. Chúng cũng giống với các vòng lặp for trong C.
Cú pháp của một vòng lặp for là:

```
for (expr1; expr2; expr3) statement
```

Biểu thức thứ nhất (expr1) được định giá (được thực hiện) một lần vô điều kiện khi bắt đầu vòng lặp.
Trong lúc bắt đầu của mỗi lần lặp, expr2 được định giá. Nếu nó định giá là TRUE, vòng lặp được tiếp tục và các câu lệnh bên trong vòng lặp sẽ được thực hiện. Nếu nó định giá là FALSE, sự thực thi của vòng lặp sẽ kết thúc.
Tại cuối mỗi lần lặp, expr3 được định giá (được thực hiện).
Mỗi một biểu thức có thể trống rỗng. expr2 đang trống có nghĩa là vòng lặp sẽ được chạy vô hạn định(PHP hoàn toàn xem nó là TRUE, giống như C). Điều này có thể không vô dụng như ta nghĩ, trước đây thông thường ta muốn kết thúc vòng lặp bằng cách dùng câu lệnh điều kiện break thay vì sử dụng biểu thức đúng đắn for
Hãy xem xét các ví dụ sau. Tất cả chúng đều hiển thị các số từ 1 đến 10:
```
/* example 1 */

for ($i = 1; $i <= 10; $i++) {
   print $i;
}

/* example 2 */

for ($i = 1;;$i++) {
   if ($i > 10) {
      break;
   }
```

```
   print $i;
}

/* example 3 */

$i = 1;
for (;;) {
   if ($i > 10) {
      break;
   }
   print $i;
   $i++;
}

/* example 4 */

for ($i = 1; $i <= 10; print $i, $i++) ;
```

Dĩ nhiên, ví dụ đầu tiên xuất hiện một cách tốt đẹp nhất, nhưng ta có thể thấy rằng sử dụng các biểu thức trống trong các vòng lặp for là dễ sử dụng trong nhiều trường hợp.
PHP cũng cung cấp cú pháp hai chấm đan xen nhau cho các vòng lặp for:

```
for (expr1; expr2; expr3): statement; ...; endfor;
```

Các ngôn ngữ khác có một câu lệnh foreach để duyệt một danh sách hoặc một mớ lộn xộn. PHP3 không có cấu trúc này, PHP4 thì có. Trong PHP3, ta có thể nối while với các hàm list() và each() để lấy ra kết qủa tương tự.

8_Cấu trúc Foreach

PHP4 (không phải PHP3) bao gồm một cấu trúc foreach, khá giống với perl và một số ngôn ngữ khác. Cấu trúc này đơn giản tạo ra một cách dễ dàng để duyệt qua các mảng. Có hai cú pháp; cú pháp thứ hai là thứ yếu nhưng là sự mở rộng một cách hữu ích của cấu trúc thứ nhất:

```
foreach(array_expression as $value) statement
foreach(array_expression as $key => $value) statement
```

Dạng đầu tiên của các vòng lặp trên mảng được đưa ra bởi array_expression. Trên mỗi vòng lặp, giá trị của các phần tử hiện tại được gán cho $value và con trỏ mảng cục bộ được tăng lên một(bởi vậy trong vòng lặp sau, ta sẽ thấy phần tử tiếp theo).
Dạng thứ hai cũng thực hiện việc tương tự, trừ khoá của phần tử hiện tại sẽ được gán tới biến $key trên mỗi vòng lặp.
Chú ý: Khi foreach lần đầu tiên bắt đầu thực hiện, con trỏ cục bộ tự động được điều chỉnh lại tới phần tử đầu tiên của mảng . Điều này có nghĩa là ta không cần gọi hàm reset() trước mỗi vòng lặp foreach.
Chú ý: chú ý rằng foreach thao tác trên một bản sao chép của mảng được chỉ ra, không phải là chính mảng đó. Do vậy, con trỏ mảng không bị thay đổi giống như mỗi cấu trúc.
Ta phải chú ý rằng các ví dụ sau có cùng một chức năng như nhau:

```
reset ($arr);
while (list(, $value) = each ($arr)) {
   echo "Value: $value<br>\n";
}

foreach ($arr as $value) {
   echo "Value: $value<br>\n";
}
```

Sau đây cũng có chức năng tương tự:

```
reset ($arr);
while (list($key, $value) = each ($arr)) {
   echo "Key: $key; Value: $value<br>\n";
}

foreach ($arr as $key => $value) {
   echo "Key: $key; Value: $value<br>\n";
}
```

Thêm một vài ví dụ để mô tả cách sử dụng:

```
/* foreach example 1: value only */

$a = array (1, 2, 3, 17);

foreach ($a as $v) {
  print "Current value of \$a: $v.\n";
}

/* foreach example 2: value (with key printed for illustration) */

$a = array (1, 2, 3, 17);

$i = 0; /* for illustrative purposes only */

foreach($a as $v) {
   print "\$a[$i] => $v.\n";
}

/* foreach example 3: key and value */

$a = array (
   "one" => 1,
   "two" => 2,
   "three" => 3,
   "seventeen" => 17
);

foreach($a as $k => $v) {
   print "\$a[$k] => $v.\n";
}
```

9_Cấu trúc break

break kết thúc sự thực hiện của các cấu trúc for, while hoặc switch hiện tại.
break chấp nhận một số lượng đối số tuỳ chọn chỉ ra rằng bằng cách nào rất nhiều các cấu trúc khép kín được đặt trong đó bị bẻ gãy

```
$arr = array ('one', 'two', 'three', 'four', 'stop', 'five');
while (list (, $val) = each ($arr)) {
   if ($val == 'stop') {
      break;    /* You could also write 'break 1;' here. */
   }
   echo "$val<br>\n";
}
```

```
/* Using the optional argument. */

$i = 0;
while (++$i) {
   switch ($i) {
   case 5:
      echo "At 5<br>\n";
      break 1;  /* Exit only the switch. */
   case 10:
      echo "At 10; quitting<br>\n";
      break 2;  /* Exit the switch and the while. */
   default:
      break;
   }
}
```

10_Cấu trúc continue

continue được dùng trong các cấu trúc vòng lặp để bỏ qua phần cuối của vòng lặp hiện tại và tiếp tục thực hiện tại phần đầu của vòng lặp tiếp theo.
continue chấp nhận một số lượng các đối số tuỳ chọn mà chỉ cho biết có bao nhiêu mức của các vòng lặp khép kín sẽ được bỏ qua cho đến cuối.

```
while (list ($key, $value) = each ($arr)) {
   if (!($key % 2)) { // skip odd members
      continue;
   }
   do_something_odd ($value);
}

$i = 0;
while ($i++ < 5) {
   echo "Outer<br>\n";
   while (1) {
      echo "  Middle<br>\n";
      while (1) {
         echo "  Inner<br>\n";
         continue 3;
      }
      echo "This never gets output.<br>\n";
   }
   echo "Neither does this.<br>\n";
}
```

11_Cấu trúc switch

Câu lệnh switch cũng giống tương tự như chuỗi các câu lệnh if trong cùng một biểu thức. Trong rất nhiều dịp, ta cần phải so sánh cùng biến (hoặc biểu thức) với rất nhiều giá trị khác nhau, và thực hiện một phần khác nhau của mã nguồn dựa trên giá trị nào mà nó bằng. Điều này là chính xác điều gì câu lệnh switch dùng cho.
Hai ví dụ dưới đây là hai cách khác nhau để viết cùng một thứ, một sử dụng một chuỗi các câu lệnh if, và một dùng câu lệnh switch:

```
if ($i == 0) {
   print "i equals 0";
}
if ($i == 1) {
```

```
   print "i equals 1";
}
if ($i == 2) {
   print "i equals 2";
}

switch ($i) {
   case 0:
      print "i equals 0";
      break;
   case 1:
      print "i equals 1";
      break;
   case 2:
      print "i equals 2";
      break;
}
```

Điều quan trọng để hiểu bằng cách nào câu lệnh switch được thực hiện theo trình tự để tránh lỗi. Câu lệnh switch thực hiện từng dòng một (thực ra là từng câu lệnh một). Trong lúc bắt đầu, không đoạn mã nào được thực hiện.Chỉ khi một câu lệnh case được tìm thấy với giá trị thích hợp với giá trị của biểu thức switch làm cho PHP bắt đầu thực hiện các câu lệnh. PHP tiếp tục thực hiện các câu lệnh cho đến khi kết thúc khối câu lệnh switch, hoặc lần đầu tiên nó gặp câu lệnh break. Nếu ta không viết một câu lệnh break tại cuối danh sách các câu lệnh case, PHP sẽ tiếp tục thực hiện các câu lệnh của case dưới đó. Ví dụ:

```
switch ($i) {
   case 0:
      print "i equals 0";
   case 1:
      print "i equals 1";
   case 2:
      print "i equals 2";
}
```

Ở đây, nếu $i bằng 0, PHP sẽ thực hiện tất cả các câu lệnh print. Nếu $i bằng 1, PHP sẽ thực hiện 2 câu lệnh print cuối, và nếu $i chỉ bằng 2, khi đó chỉ có ‘ i equals 2 ‘ sẽ được hiển thị. Bởi vậy, điều quan trọng là không được quên các câu lệnh break.
Trong một câu lệnh switch, điều kiện chỉ được xác định một lần và kết quả được so sánh cho mỗi câu lệnh case. Trong một câu lệnh elseif, điều kiện được xác định lại. Nếu điều kiện của ta phức tạp hơn một điều kiện so sánh đơn giản và/hoặc trong một vòng lặp kín, một switch có thể nhanh hơn.
Danh sách câu lệnh cho một case cũng có thể được bỏ trống, nó đơn giản cho qua điều khiển trong danh sách câu lệnh cho case tiếp theo.

```
switch ($i) {
   case 0:
   case 1:
   case 2:
      print "i is less than 3 but not negative";
      break;
   case 3:
      print "i is 3";
}
```

Một case đặc biệt là một case mặc định. Case này thích hợp với bất kỳ trường hợp nào mà không phù hợp với các case khác, và sẽ là câu lệnh case cuối cùng, ví dụ:

```
switch ($i) {
   case 0:
      print "i equals 0";
      break;
   case 1:
      print "i equals 1";
      break;
   case 2:
      print "i equals 2";
      break;
   default:
      print "i is not equal to 0, 1 or 2";
}
```

Biểu thức case có thể là bất kỳ biểu thức nào mà xác định tới một kiểu đơn giản, đó là, kiểu số nguyên hoặc các số thực động và các chuỗi. Các mảng hoặc các đối tượng không thể được dùng ở đây nếu chúng không được tham chiếu đến một kiểu đơn giản.
Cú pháp lựa chọn cho các cấu trúc điều khiển đọc hỗ trợ với các switch.

```
switch ($i):
   case 0:
      print "i equals 0";
      break;
   case 1:
      print "i equals 1";
      break;
   case 2:
      print "i equals 2";
      break;
   default:
      print "i is not equal to 0, 1 or 2";
endswitch;
```

12_require()

Câu lệnh require() thay thế chính bản thân nó với file được chỉ định, khá giống với C trong các công việc tiền định nghĩa #include.
Nếu "URL fopen wrappers" được thiết lập trong PHP (Chúng trong cấu hình mặc định), ta có thể chỉ ra file được require() dùng một URL thay vì một đường dẫn cục bộ.
Điều quan trọng cần chú ý về việc bằng cách nào điều này thực hiện là khi một file được include() hoặc require(), bỏ qua chế độ cú pháp PHP và chuyển sang chế độ HTML tại lúc bắt đầu của file đích, và bắt đầu lại chế độ PHP một lần nữa khi kết thúc. Nguyên nhân do chính lý do này là, bất kỳ mã nào bên trong file nguồn mà sẽ được thực hiện như mã PHP phải được đóng kín trong các tag bắt đầu và kết thúc chính xác của PHP.
Require() không phải là một hàm thực sự trong PHP; đúng hơn, nó là một cấu trúc ngôn ngữ. Nó là một chủ đề của một số nguyên tắc khác nhau hơn là các hàm. Trong trường hợp cá biệt, require() không đưa ra bất kỳ cấu trúc điều khiển chứa nào. Trong trường hợp khác, nó không trả về bất kỳ một giá trị nào;cố gắng đọc một giá trị trả về từ một lời gọi require()đưa đến một cú pháp lỗi.
Không giống include(), require() sẽ luôn luôn đọc trong file đối tượng, ngay cả nếu nếu dòng nó không bao giờ thực hiện. Nếu ta muốn bao gồm một file có điều kiện, sử dụng include(). Câu lệnh điều kiện sẽ không ảnh hưởng đến require(). Tuy nhiên, nếu dòng trên đó require() xảy ra không được thực hiện, cũng sẽ không có mã nào trong file đối tượng sẽ được thực hiện.

Thông thường, các cấu trúc lặp không ảnh hưởng đến tình trạng của require(). Măc dù mã có chứa trong file đối tượng vẫn còn lệ thuộc vào vòng lặp, bản thân require() chỉ xảy ra một lần. Điều này có nghĩa là ta không thể đặt một câu lệnh require() bên trong một cấu trúc lặp và mong chờ nó bao gồm các nội dung của một file khác trong mỗi vòng lặp. Để làm điều đó, sử dụng một câu lệnh include()

```
require ('header.inc');
```

Khi một file được require(), mã nó chứa thừa kế phạm vi biến của dòng trên đó require(0 xảy ra. Bất kỳ các biến có sẵn tại dòng đó trong file gọi sẽ có hiệu lực trong file được gọi. Nếu require() xảy ra bên trong một hàm trong một file gọi, khi đó tất cả mã chứa trong file được gọi sẽ được chạy mặc dù nó đã được định nghĩa bên trong hàm đó.

Ngoài các cấu trúc hay dùng như trên, ta còn có một số cấu trúc khác có thể tham khao thêm trong PHP manual như: include(), require_once() , include_once().

VIII_CÁC HÀM

1_Các hàm được định nghĩa bởi người sử dụng

Một hàm có thể được định nghĩa bằng cách sử dụng cú pháp như sau:

```
function foo ($arg_1, $arg_2, ..., $arg_n) {
   echo "Example function.\n";
   return $retval;
}
```

Bất kỳ mã PHP đúng nào cũng có thể xảy ra bên trong một hàm, ngay cả các hàm khác và các định nghĩa lớp.

2_Các đối số của hàm

Thông tin có thể được chuyển tới các hàm thông qua danh sách đối số, đó là một danh sách được phân cách bởi dấu phẩy của các biến và/hoặc các hằng số.
PHP hỗ trợ truyền các đối số bởi giá trị (măc định), truyền qua tham chiếu, và các giá trị đối số mặc định. Một kết quả thông thường có thể đạt được trong PHP 3 bằng cách thông qua một mảng các đối số tới một hàm:

```
function takes_array($input) {
   echo "$input[0] + $input[1] = ", $input[0]+$input[1];
}
```

2.1_Tạo các đối số bằng cách truyền bằng tham chiếu

Bởi mặc định, các đối số của hàm được truyền bằng tham trị (bởi vậy nếu ta thay đổi giá trị của đối số trong một hàm, nó sẽ không thay đổi giá trị của nó bên ngoài hàm). Nếu ta muốn cho phép một hàm thay đổi các đối số của nó, ta phải truyền nó bằng tham chiếu.
Nếu ta muốn một đối số đến một hàm để luôn luôn được truyền bằng tham chiếu, ta có thể thêm một dấu và (&) phía trước tên của đối số trong định nghĩa hàm:

```
function add_some_extra(&$string) {
   $string .= 'and something extra.';
}
$str = 'This is a string, ';
add_some_extra($str);
echo $str;    // outputs 'This is a string, and something extra.'
```

Nếu ta muốn truyền một biến bằng tham chiếu tới một hàm mà không thực hiện điều này bằng mặc định, ta có thể điền một dấu và (&) phía trước tên của đối số trong lời gọi hàm:

```
function foo ($bar) {
   $bar .= ' and something extra.';
}
$str = 'This is a string, ';
foo ($str);
echo $str;    // outputs 'This is a string, '
foo (&$str);
echo $str;    // outputs 'This is a string, and something extra.'
```

2.2_Các giá trị đối số mặc định

Một hàm có thể định nghĩa các giá trị mặc định theo kiểu của C++ cho các đối số vô hướng như sau:

```
function makecoffee ($type = "cappucino") {
   return "Making a cup of $type.\n";
}
echo makecoffee ();
echo makecoffee ("espresso");
```

Đầu ra cho đoạn mã trên là:

```
Making a cup of cappucino.
Making a cup of espresso.
```

Giá trị mặc định phải là một biểu thức hằng số, không phải là một biến hoặc một thành phần lớp.

Chú ý rằng khi sử dụng các đối số mặc định, bất kỳ mặc định nào sẽ được đặt về phía bên phải của các đối số không mặc định; theo cách khác, những thứ sẽ không làm việc như mong chờ. Hãy xem xét đoạn mã nhỏ sau:

```
function makeyogurt ($type = "acidophilus", $flavour) {
   return "Making a bowl of $type $flavour.\n";
}

echo makeyogurt ("raspberry");   // won't work as expected
```

Đầu ra cho ví dụ trên là:

```
Warning: Missing argument 2 in call to makeyogurt() in
/usr/local/etc/httpd/htdocs/php3test/functest.html on line 41
Making a bowl of raspberry .
```

Bây giờ hãy so sánh với đoạn này:

```
function makeyogurt ($flavour, $type = "acidophilus") {
   return "Making a bowl of $type $flavour.\n";
}

echo makeyogurt ("raspberry");   // works as expected
```

Đầu ra cho ví dụ này là:

```
Making a bowl of acidophilus raspberry.
```

2.3_Các danh sách đối số chiều dài biến

PHP4 cung cấp các danh sách đối số chiều dài biến trong các hàm được định nghĩa bởi người sử dụng . Điều này thực sự khá dễ dàng, dùng các hàm func_num_args(),

func_get_arg() và hàm func_get_args().

Không yêu cầu cú pháp đặc biệt nào, và các danh sách đối số sẽ vẫn còn được cung cấp một cách rõ ràng với các định nghĩa hàm và sẽ trở nên thông thường.

3_Các giá trị trả về

Các giá trị được trả về bằng cách dùng câu lệnh trả về tuỳ chọn. Bất kỳ kiểu nào cũng có thể được trả về, bao gồm cả các danh sách và các đối tượng.

```
function square ($num) {
   return $num * $num;
}
echo square (4);   // outputs '16'.
```

Ta không thể trả về nhiều giá trị từ một hàm, nhưng các kết quả thông thường có thể đạt được bằng cách trả lại một danh sách.

```
function small_numbers() {
   return array (0, 1, 2);
}
list ($zero, $one, $two) = small_numbers();
```

Để trả về một tham chiếu từ một hàm, ta phải dùng toán tử tham chiếu & trong cả khai báo hàm và khi gán giá trị trả về cho một biến:

```
function &returns_reference() {
   return $someref;
}

$newref =&returns_reference();
```

4_ Old_function

Câu lệnh old_function cho phép ta khai báo một hàm sử dụng một cú pháp giống hệt PHP/FI2 (không thể ta phải thay thế 'function' bằng 'old_function').
Đây là một đặc điểm không được tán thành, và chỉ được sử dụng bởi bộ chuyển đổi PHP/FI2 sang PHP3.

Cảnh báo

Các hàm được khai báo như old_function và không thể được gọi từ mã cục bộ của PHP. Chính vì vậy mà ta không thể dùng chúng trong các hàm như **usort()**, **array_walk()**, **register_shutdown_function()**.

5_Các hàm biến thiên

PHP hỗ trợ khái niệm của các hàm biến thiên. Điều này có nghĩa là nếu một tên biến có các dấu ngoặc đơn được thêm vào nó, PHP sẽ tìm kiếm một hàm có cùng tên như thể cho dù biến định giá tới, và sẽ cố gắng để thực hiện chúng. Nói một cách khác, điều này có thể được dùng để thực hiện gọi trở lại, các bảng chức năng,và ra khỏi.
Ví dụ về hàm biến thiên:

```
<?php
function foo() {
   echo "In foo()<br>\n";
}

function bar( $arg = '' ) {
   echo "In bar(); argument was '$arg'.<br>\n";
}

$func = 'foo';
$func();
$func = 'bar';
$func( 'test' );
?>
```

IX_ CÁC LỚP VÀ CÁC ĐỐI TƯỢNG

1_Lớp

Một lớp là một tập hợp của các biến và các hàm làm việc trong các biến này. Một lớp được định nghĩa sử dụng cú pháp sau:

```
<?php
class Cart {
   var $items;  // Items in our shopping cart

   // Add $num articles of $artnr to the cart

   function add_item ($artnr, $num) {
      $this->items[$artnr] += $num;
   }

   // Take $num articles of $artnr out of the cart

   function remove_item ($artnr, $num) {
      if ($this->items[$artnr] > $num) {
         $this->items[$artnr] -= $num;
         return true;
      } else {
         return false;
      }
   }
}
?>
```

Điều này định nghĩa một lớp có tên là Cart gồm có một mảng kết hợp các thực phẩm trong cái xe chở và hai hàm để thêm và loại bỏ các phần tử từ cái xe chở này.
Chú ý: Trong PHP4, chỉ có các khởi tạo hằng cho các biến var là được phép. Dùng các hàm khởi tạo cho các khởi tạo không là hằng số.

```
/* None of these will work in PHP 4. */
class Cart {
   var $todays_date = date("Y-m-d");
   var $name = $firstname;
   var $owner = 'Fred ' . 'Jones';
}

/* This is how it should be done. */
class Cart {
   var $todays_date;
   var $name;
   var $owner;

   function Cart() {
      $this->todays_date = date("Y-m-d");
      $this->name = $GLOBALS['firstname'];
      /* etc. . . */
   }
}
```

Các lớp là các kiểu , do vậy, chúng là bản kế hoạch cho các biến thật sự. Ta phải tạo một biến có kiểu mong muốn với toán tử new.

```
$cart = new Cart;
$cart->add_item("10", 1);
```

Điều này tạo một đối tượng $cart của lớp Cart. Hàm add_item() của đối tượng đó đang được gọi để thêm một phần tử của thực phẩm số 10 vào xe chở.
Các lớp có thể được mở rộng tới các lớp khác. Lớp được thừa hưởng hoặc được mở rộng có tất cả các biến và các hàm của lớp cơ sở và những gì bạn thêm vào định nghĩa được mở rộng. Điều này được thực hiện bằng cách dùng từ khoá extend. Đa thừa kế không được hỗ trợ.

```
class Named_Cart extends Cart {
   var $owner;

   function set_owner ($name) {
      $this->owner = $name;
   }
}
```

Điều này định nghĩa một lớp Named_Cart mà có tất cả các biến và các hàm của Cart cộng với một biến thêm $owner và một hàm thêm set_owner(). Ta tạo một xe chở được chỉ định theo cách thông thường và bây giờ có thể thiết lập và lấy chủ của xe chở. Ta cũng có thể sử dụng các hàm xe chở thông thường trên các xe chở được chỉ định:

```
$ncart = new Named_Cart;    // Create a named cart
$ncart->set_owner ("kris"); // Name that cart
print $ncart->owner;        // print the cart owners name
$ncart->add_item ("10", 1); // (inherited functionality from cart)
```

Bên trong các hàm của một lớp biến $this có ý nghĩa là đối tượng này. Ta phải dùng $this->something để truy nhập đến bất kỳ biến nào hoặc hàm nào được đặt tên something trong đối tượng hiện thời. Cả bên trong lẫn bên ngoài của đối tượng ta không cần một dấu $ khi truy nhập vào các thuộc tính của một đối tượng.

```
$ncart->owner  = "chris"; // no $

$ncart->$owner = "chris";
// this is invalid because $ncart->$owner = $ncart->""

$myvar = 'owner';
$ncart->$myvar = "chris";
// this is valid because $ncart->$myvar = $ncart->owner
```

Các hàm khởi tạolà các hàm trong một lớp và được gọi một cách tự động khi ta tạo một thể hiện mới của một lớp. Một hàm trở thành một hàm khởi tạo khi nó có cùng tên với tên của lớp.

```
class Auto_Cart extends Cart {
   function Auto_Cart () {
      $this->add_item ("10", 1);
   }
}
```

Điều này định nghĩa một lớp Auto_Cart đó là một Cart thêm một hàm khởi tạo mà khởi tạo một xe chở với một phần tử thực phẩm số “10” mỗi lần một Auto_Cart được tạo với từ “new”. Các hàm khởi tạo cũng có thể đưa ra các đối số và các đối số này có thể được tuỳ chọn, nó tạo cho chúng sự thuận tiện nhiều hơn.

```
class Constructor_Cart extends Cart {
   function Constructor_Cart ($item = "10", $num = 1) {
      $this->add_item ($item, $num);
   }
}

// Shop the same old boring stuff.

$default_cart   = new Constructor_Cart;

// Shop for real...

$different_cart = new Constructor_Cart ("20", 17);
```

Chú thích

Đối với các lớp được thừa kế, hàm khởi tạo của lớp cha không được gọi một cách tự động khi hàm khởi tạo của lớp được thừa kế được gọi.

2_Các tham chiếu bên trong hàm khởi tạo

Việc tạo ra các tham chiếu bên trong hàm khởi tạo có thể dẫn đến các kết quả khó hiểu.

```
class foo {
   function foo($name) {
      // create a reference inside the global array $globalref
      global $globalref;
                        $globalref[] = &$this;
      // set name to passed value
      $this->setName($name);
                        // and put it out
      $this->echoName();
   }
function echoName() {
      echo "<br>",$this->Name;
   }

            function setName($name)          {
                        $this->Name = $name;
   }
}
```

Chúng ta hãy cùng kiểm tra nếu có một sự khác biệt giữa $bar1 mà đã được tạo bằng cách dùng toán tử = sao chép và $bar2 mà được tạo bằng cách dùng toán tử tham chiếu =& ...

```
$bar1 = new foo('set in constructor');
   $bar1->echoName();
   $globalref[0]->echoName();

   /* output:
   set in constructor
   set in constructor
   set in constructor */

   $bar2 =& new foo('set in constructor');
   $bar2->echoName();
   $globalref[1]->echoName();
```

```
/* output:
set in constructor
set in constructor
 set in constructor */
```

Dường như không có sự khác biệt, nhưng trong thực tế có một điểm rất khác biệt: $bar1 và $globalref[0] đã không được tham chiếu, chúng không cùng chung biến. Điều này giải thích tại sao "new" mặc định không trả về một tham chiếu, thay vì đó nó trả về một sao chép.
Chú ý: Không có sự thực hiện nào bị mất mát (từ khi PHP4 và trên nữa sử dụng cả tham chiếu) trả lại các bản sao chép thay vì các tham chiếu. Trái ngược lại, tốt hơn hết là làm việc một cách đơn giản với các bản sao chép thay vì các tham chiếu, bởi vì việc tạo ra các tham chiếu đòi hỏi các tốn thời gian trong khi việc tạo ra các bản sao chép dường như không mất thời gian (nếu không phải không số nào trong chúng là một mảng lớn hoặc đối tượng và một trong số chúng đã thay đổi và các số khác còn lại sau đó, tiếp đến nó sẽ chỉ ra cách dùng các tham chiếu để thay đổi chúng một cách đồng thời).
Để chứng tỏ điều gì được viết ở trên chúng ta hãy xem đoạn mã dưới đây:

```
// now we will change the name. what do you expect?
      // you could expect that both $bar and $globalref[0] change their names...
      $bar1->setName('set from outside');

      // as mentioned before this is not the case.
      $bar1->echoName();
      $globalref[0]->echoName();

      /* output:
      set on object creation
      set from outside */

      // let us see what is different with $bar2 and $globalref[1]
      $bar2->setName('set from outside');

      // luckily they are not only equyl, they are thesame variable
      // thus $bar2->Name and $globalref[1]->Name are the same too
      $bar2->echoName();
      $globalref[1]->echoName();

      /* output:
      set from outside
       set from outside */
```

Một ví dụ cuối cùng, hãy cố gắng để hiểu nó:

```
class a {
   function a($i) {
      $this->value = $i;
      // try to figure out why we do not need a reference here
      $this->b = new b($this);
   }

   function createRef() {
      $this->c = new b($this);
   }

   function echoValue() {
      echo "<br>","class ",get_class($this),': ',$this->value;
      }
```

```
}


class b  {

   function b(&$a) {
      $this->a = &$a;
   }

  function echoValue() {
      echo "<br>","class ",get_class($this),': ',$this->a->value;
      }

}

// try to undestand why using a simple copy here would yield
// in an undesired result in the *-marked line
$a =& new a(10);
$a->createRef();

$a->echoValue();
$a->b->echoValue();
$a->c->echoValue();

$a->value = 11;

$a->echoValue();
$a->b->echoValue(); // *
$a->c->echoValue();

/*
output:
class a: 10
class b: 10
class b: 10
class a: 11
class b: 11
class b: 11
*/
```