

K.N.Govt Arts College for Women(A),Thanjavur-7

Department of Computer Science

Subject:Open Source Technology

Subject code: 18KP1CS02

Subject Incharge:

1.J.Shanmuga Priya.,Guest Lecturer in ComputerScience

2.N.Anuradha.,Guest Lecturer in ComputerScience.

OPEN SOURCE TECHNOLOGY-(18KP1CS02)

Course content

UNIT-I TheWeb Explained: How it works- Security –Linux: The choice of GNU Generation. Introduction-Linux Distributions –Accounts-Security-Basic Unix Shell-Owner-Groups permissions,ownership-Processes-Path and Environment-commands-Basic file system Essentials-Useful Programs.

UNIT II: Apache Web Server:Introduction-Starting,stopping, and Restarting Apache- Configuration-Securing Apache-Create the website –Apache Log Files.

UNIT III: Perl : Introduction-Perl Documentation –Perl Syntax Rules: A first perl programs- Example: -Declaring variables with use strict- variables-operators –flow control constructs- Regular expressions-functions-File I/O –Additional Perl Constructs-Making Operating System Calls.-Quick introduction to OOP.

UNIT IV: MySQL: Introduction-The SHOW DATABASE and CREATE DATABASE commands- The USE command-CREATE TABLE and SHOW TABLES command-DESCRIBE ,INSERT,SELECT,UPDATE, AND DELETE commands-Database independent Interface-Table joins-Loading and Dumping a database.

UNIT V: PHP : Introduction Embedding PHP into HTML-Configuration-Language syntax- Variables ,Data Types- Web variables -Operators- Flow control constructs- Writing PHP functions-Built –in PHP functions: Important functions-Array functions-String `functions-Other functions-PHP and MySQL functions.

Reference:

“Open Source web development with lamp using Linux,Apache,MySQL,Perl and PHP,james lee and Brent ware,Pearson education and dorling Kindersley(india)pvt.Ltd.,fourth Impression,2009.

UNIT-I

WEB EXPLAINED

The web is usually accessed through a browser .The most well known browsers are

- Mozilla/Netscape
- Internet Explorer

Some other alternative browsers are

- Galeon
- Omniweb and Opera

Text based options browsers are such as Bynr, Links,and W3g

When click a link or type a URL into the location box, the browser makes a socket connection to the server. The Browser connects the URL using ports 80 the server Operating System open for such HTTP requests.

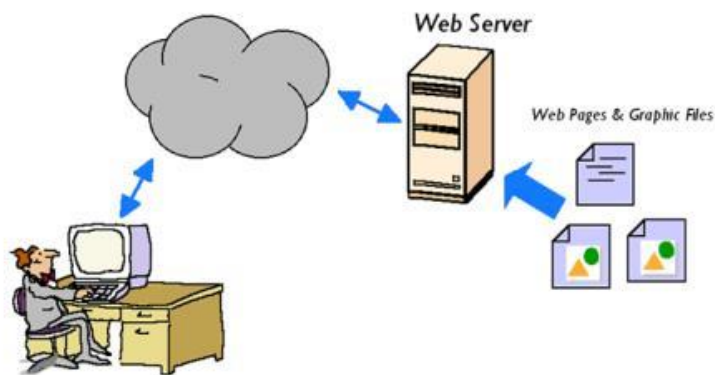
Based on the client request ,the server serves up or delivers information to the client.

The type of data server serves include

- Text
- Images
- Java applets
- Various type documents and
- PDF
- The Clients job is to receive from server a stream of text ,images and executes any java applets that are served up. And the client can send data to the server using Common Gateway Interface(CGI)
- The primary function of a web server is to store, process and deliver web pages to clients.

The Client-Server Model for the Web

1. A Web Client (usually in the form of a web browser) makes an HTTP request to a specific web server.
2. The Web Server receives the request and sends back the requested document (usually in the form of an HTML page).
3. The web client interprets the information returned by the server and displays it appropriately.

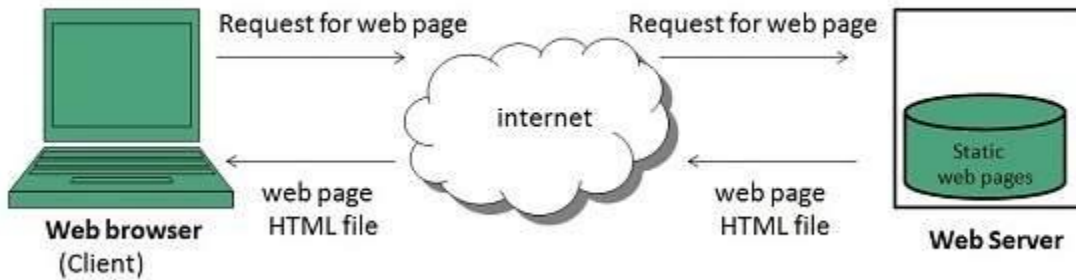


A web application's content can be of two types: dynamic or static. Dynamic content is anything that requires some type of processing to be generated (e.g. a PHP script or Java Server Page), whereas static content is something that will never or occasionally change (e.g. a JavaScript library or HTML file).

Server can process this data in two ways –Client side and Server Side

Serving up Static Data:

- ✓ The simplest thing for the server to do so too serve static data that is the same way for every client and changes only when a HTML Programmer changes the source file.
- ✓ The server just locating file on the local drive and send the content back to the client undamaged.
- ✓ This requires no server programming (the Apache Server)



Serving up Dynamic Data:

- ✓ More Complex way of generating HTML is to execute a server –side program that dynamically generates the HTML that is sent to the browser.
- ✓ The server locates the file in the local drive and execute the program and sent back to the client.

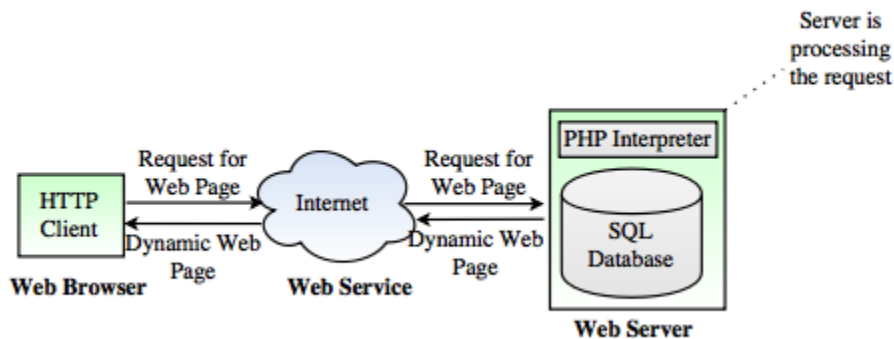


Fig. Dynamic Web Page

Serving up content with Embedded HTML:

- ✓ Flexible way to create dynamic web pages is too use embedded HTML or executable code embedded within an HTML file.
- ✓ Dynamic web pages embed with HTML provide a measure of flexibility
- ✓ If the URL is www.example.com/a.html ,web server grabs HTML file a.html
- ✓ Preprocesses it and generating HTML by execute the code within the original HTML file.

Four approaches to embedded programming:

SSI-Server Side Includes

Simple solution built into a Apache using a syntax unique to SSI

- ✓ EmbPerl: A perl module that enables an HTML to have Perl code embedded within it.
- ✓ Mason: Another Perl module like EmbPerl.
- ✓ PHP: A language itself ,Perl like syntax provide rich built-in function to perform various tasks.

Some popular technologies not Open Source:

XML: Extensible Mark Up Language- XML will replace HTML as the language used to create web pages.

Java: An Object –Oriented Programming Language used to create applets and stand alone applications. API provide methods to create GUI application.

Java Script: Java like language –Manipulates the client browser- used to Pop-up new browser windows.

Security:

It is a process not a step. Install a software , or change password,security should be foremost in mind.

Linux-The Choice of GNU Generation

- ✓ **Linux** is a community of open-source Unix like operating systems that are based on the **Linux** Kernel.
- ✓ **Linux** is a tried-and-true, open-source operating system released in 1991 for computers, but its **use** has expanded to underpin systems for cars, phones, web servers and, more recently, networking gear.
- ✓ It was initially released by Linus Torvalds on September 17, 1991. Due to android **Linux** has the largest installed base of all general-purpose operating systems.

Linux distributions:

- ✓ Linux is free and Open Source we can download and compile them all by ourselves.
- ✓ Linux distribution is an operating system that is made up of a collection of software based on Linux kernel or you can say distribution contains the Linux kernel and supporting libraries and software.

- ✓ And we can get Linux based operating system by downloading one of the Linux distributions and these distributions are available for different types of devices like embedded devices, personal computers.

Debian:

Debian GNU/Linux, is a Linux distribution composed of free and open source software, developed by the community-supported Debian Project

- Advanced package Tool- powerful and high esteemed updater
- apt-get handle package dependency

Red Hat: popular in united states

- RPM -Red Hat Package Manager similar to apt-get to keep software up to date
- Up2date provide package dependency resolution

Slackware:

- First Linux Distribution used by many hard core Linux users.
- Contain user friendly interfaces similar other distributions

SuSE : Most popular in Europe

- Popular distribution uses a variant of RPM package
- Include a sophisticated system tool YaST(Yet another Setup Tool) to make administration easier

Mandrake:

- Took Red Hat distribution ,from added easy to use installer and changed the default desktop to KDE.
- It has a reputation for being easy to use, to install and cutting-edge

Download and Install:

- Down load for free or get it from www.cheapbytes.com for a very minimal cost.
- Download the software from Red Hat or mirror under existing Linux software and burn a CD .Having a CD around is nice for reinstalling , fixing problems, having backup and other necessary tasks.

Linux Partition Sizes:

Partition	Size	Description
/boot	32MB	Where boot the file
/	512MB	Base of the OS
/usr	5GB	All the programs and documents here
/home	1GB	User files here
/var	2.25GB	All the
/tmp	256 MB	Web source and web logs here
/web	Optional	Optional place to put web source
/root	Optional	Root user space
/swap		Twice times the amount of physical memory

- ✓ After backed up and partitioned insert the Linux installation CD and reboot from it. Follow the direction included with the distribution and install everything.

Accounts:

- ✓ The installer will ask to create a root account.
- ✓ Create atleast one root account. And later create any other account for other user.
- ✓ The Red Hat installer creates several other user for specific purpose. Apache will be owned by the apache user.
- ✓ Port numbers under 1024 are owned by root and Apache runs on port 80.

Security:

- ✓ **Red Hat Product Security** provides the guidance, stability and **security** needed to confidently deploy enterprise solutions.
- ✓ Help protect customers from meaningful **security** concerns when using **Red Hat** products and services.
- ✓ Red Hat not allow us to pick the services turned on during installation. Provide firewall level of security during installation.

Basic UNIX

Linux is primarily used through a Command line Interface.

- ✓ **Command Line Interface (CLI):**
The Command Line Interface (CLI), is a non-graphical, text-based interface to the computer system, where the user types in a command and the computer then successfully executes it.
- ✓ **man**command in Linux is used to **display** the user manual of any command that we can run on the terminal. It provides a detailed view of the command.

- ✓ Most of the Linux distributions come with some sort of desktop to give Apple/Windows feel. In some respects ,because OS X and XP come up with equivalent multiple desktop that are standard Gnome and KDEWeb site development and administration work has to be done as **root**.

Shell:

Shell is a UNIX term for the interactive user interface with an operating system. The **shell** is the layer of programming that understands and executes the commands a user enters.

- bash is the default, shell when Red Hat is installed.
- tcsh , a variation on the original csh. The other also zsh and ksh
- The preference of shell are found in .tcshrc or .bashrc or .zshrc
- Changing of shell with .chsh command, which asks for a password.

A shell is a computer program that presents a command line **interface** which allows you to control your computer using commands entered with a keyboard ,instead of controlling graphical user interfaces (GUIs) with a mouse/keyboard combination.

Owner ,Groups,Permissions and Ownership

Everything in Unix is a file, and each of the file has associated with it an owner and group. ls command list directories and ownership and group information.

File ownership is an important component of Unix that provides a secure method for storing files. Every file in Unix has the following attributes –

- **Owner permissions** – The owner's permissions determine what actions the owner of the file can perform on the file.
- **Group permissions** – The group's permissions determine what actions a user, who is a member of the group that a file belongs to, can perform on the file.
- **Other (world) permissions** – The permissions for others indicate what action all other users can perform on the file.

Unix permissions command

linuxfrombeginning.wordpress.com

- ✓ Change permissions for a file in Unix. You can change file permissions with the **chmod** command.
- ✓ In Unix, file permissions, which establish who may have different types of access to a file, are specified by both access classes and access types.
- ✓ **chown**change the on ownership of a file.

- ✓ (**rw**x-----) The file's owner may read, write, and execute the file. ... (**rw-rw-rw-**) All users may read and write the (**rw-r--r--**). The owner may read and write a file, while all others may only read the file.
- ✓ While using **ls -l** command, it displays various information related to file permission as follows

```
$ls -l /home/amrood
-rwxr-xr-- 1 amrood users 1024 Nov 2 00:10 myfile
drwxr-xr--- 1 amrood users 1024 Nov 2 00:10 mydir
```

Here, the first column represents different access modes, i.e., the permission associated with a file or a directory

No Permission	---
Execute permission	--x
Write permission	-w-
Execute and write permission: 1 (execute) + 2 (write) = 3	-wx
Read permission	r--
Read and execute permission: 4 (read) + 1 (execute) = 5	r-x
Read and write permission: 4 (read) + 2 (write) = 6	rw-
All permissions: 4 (read) + 2 (write) + 1 (execute) = 7	rwX

While using **ls -l** command, it displays various information related to file permission as follows

```
$ls -l /home/amrood  
-rwxr-xr-- 1 amrood users 1024 Nov 2 00:10 myfile  
drwxr-xr--- 1 amrood users 1024 Nov 2 00:10 mydir
```

Here, the first column represents different access modes, i.e., the permission associated with a file or a directory

The permissions are broken into groups of threes, and each position in the group denotes a specific permission, in this order: read (r), write (w), execute (x) –

- The first three characters (2-4) represent the permissions for the file's owner. For example, **-rwxr-xr--** represents that the owner has read (r), write (w) and execute (x) permission.
- The second group of three characters (5-7) consists of the permissions for the group to which the file belongs. For example, **-rwxr-xr--** represents that the group has read (r) and execute (x) permission, but no write permission.
- The last group of three characters (8-10) represents the permissions for everyone else. For example, **-rwxr-xr--** represents that there is **read (r)** only permission.

Processes:

- ✓ Every object in UNIX is file, everything that runs is process. Whenever you issue a command in Unix, it creates, or starts, a new process. When you tried out the **ls** command to list the directory contents, you started a process. A process, in simple terms, is an instance of a running program.
- ✓ If a process is running in the background, you should get its Job ID using the **ps** command. After that, you can use the **kill** command to kill the process

The top Command

- ✓ The **top** command is a very useful tool for quickly showing processes sorted by various criteria.
- ✓ It is an interactive diagnostic tool that updates frequently and shows information about physical and virtual memory, CPU usage, load averages, and your busy processes.

Setting the PATH

- ✓ An important Unix concept is the environment, which is defined by environment variables.
- ✓ When you type any command on the command prompt, the shell has to locate the command before it can be executed.
- ✓ The PATH variable specifies the locations in which the shell should look for commands. Usually the Path variable is set as follows –

\$PATH=/bin:/usr/bin

Here, each of the individual entries separated by the colon character (:) are directories. If you request the shell to execute a command and it cannot find it in any of the directories given in the PATH variable.

Commands:

Sr.No.	Command & Description
1	Cat Displays File Contents
2	Cd Changes Directory to dir name
3	Chgrp Changes file group
4	Chmod Changes permissions
5	Cp Copies source file into destination
6	Popd pop to another directory on the stack
7	Locate Finds files with names matching the given string

8	Grep Searches files for regular expressions
9	Head Displays first few lines of a file
10	Ln Creates softlink on oldname
11	Ls Displays information about file type
12	Mkdir Creates a new directory dirname
13	More Displays data in paginated form
14	Mv Moves (Renames) an oldname to newname
15	Pwd Prints current working directory
16	Rm Removes (Deletes) filename
17	Rmdir Deletes an existing directory provided it is empty

Basic File System Essentials:

A file system starts from /. This is the root node which hosts the first level directories.

Usual directories that you will find in a UNIX system are

- /bin contains the main system commands
- /etc contains the system configuration
- /dev contains the system devices
- /usr contains the user files
- /tmp contains the temporary files
 - /bin - the main system commands
 - /boot - the files used to boot the machine (not existing on macOS)
 - /dev - system devices
 - /etc - system configuration files
 - /etc/ - opt user programs configuration files
 - /home - the home directories of users (/Users in macOS)
 - /lib - the system libraries (not existing on macOS)
 - /mnt- mounted filesystems
 - /opt - user programs
 - /proc - user by kernel and processes (not existing on macOS)
 - /root - the home folder of the root user (not existing on macOS)
 - /run - (not existing on macOS)
 - /sbin - system binaries user for booting the system
 - /tmp - temporary files
 - /usr - holds user software, libraries and tools
 - /usr/bin -user binaries
 - /usr - include user header files
 - /usr/lib - user libraries
 - /usr/local - used by user software to store installations, /usr/sbin system binaries

- /usr/src - contains the source code of installed packages
- /var - contains temporary files, logs and more
- /proc -The unix process directory,which acts as an internal data structures of the kernel.

Useful Programs:

List of linuxprograms :

- ✓ **Log watch,swatch**- These programs watch log files ,can send e-mails documenting daily occurrences.
- ✓ **Bk2site**- Turns a list of bookmarks into a set of web pages that can be browsed.
- ✓ **Linuxconf,webin**- Configuration programs for newbie system administrators.
- ✓ **CUPS**-A printer administration tool
- ✓ **Nessus**-A system security too that scans your system to look for security holes.
- ✓ **Privoxy**- Gets rid of banner ads in your browser, and controls cookies.
- ✓ **Gkrellm**- A system monitor that keeps track of CPU ,memory .
- ✓ **AIDS,Tripwire**: Watch System configuration files to see if a cracker is altering them.

Red Hat includes the following GUI programs:

- ✓ **Neat** - Network
- ✓ **Sndconfig** - Sound card
- ✓ **Xconfigurator** - X windows
- ✓ **Usbview** -USB
- ✓ **Mouseconfig** -Mouse
- ✓ **Kbconfig** - Keyboard
- ✓ **Printtool** – Printer
- ✓ **Linudconfig** -Syst

UNIT - II

Apache Web Server

- ✓ **Apache** is an open-source and free web server software that powers around 40% of websites around the world.
- ✓ The official name is **Apache** HTTP Server, and it's maintained and developed by the **Apache** Software Foundation. ...
- ✓ Its **job** is to establish a connection between a **server** and the browsers of website visitors (Firefox, Google Chrome, Safari, etc.) while delivering files back and forth between them (client-**server** structure).
- ✓ **Apache** is a cross-platform software, therefore it **works** on both Unix and Windows server
- ✓ Apache foundation started to support the web server project, but now extends to a multitude of other projects.
- ✓ Daemon- a computer program that are always running in the background.

Apache web server

What is Web Server and how it works?

A web server is a computer system that processes requests via HTTP, the basic network protocol used to distribute information on the World Wide Web.

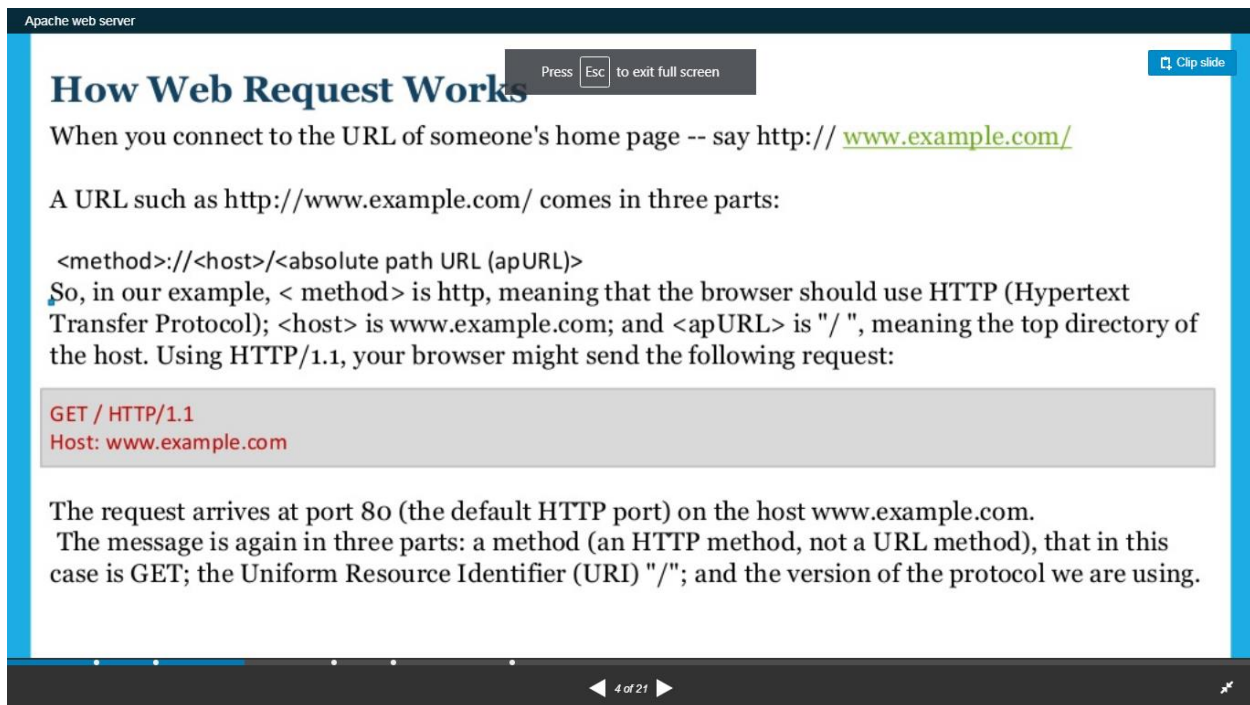
The primary function of a web server is to store, process and deliver web pages to clients. The communication between client and server takes place using the Hypertext Transfer Protocol (HTTP). Pages delivered are most frequently HTML documents, which may include images, style sheets and scripts in addition to text content.

```
graph LR; Client[Web client] -- REQUEST --> Server[WEB SERVER]; Server -- RESPONSE --> Client; subgraph Server [WEB SERVER]; direction TB; PHP; MYSQL; APACHE; end
```

3 of 21

Apache Explained:

The webserver recognizes an HTTP request by the URL of the thing requested or by file extension.



The screenshot shows a presentation slide titled "How Web Request Works" from an "Apache web server" presentation. The slide explains how a browser sends an HTTP request. It defines the components of a URL: <method>://<host>/<absolute path URL (apURL)>. An example is given: http://www.example.com/. The slide shows the resulting request: GET / HTTP/1.1 and Host: www.example.com. It also notes that the request arrives at port 80 on the host www.example.com.

Apache web server

How Web Request Works

When you connect to the URL of someone's home page -- say <http://www.example.com/>

A URL such as <http://www.example.com/> comes in three parts:

<method>://<host>/<absolute path URL (apURL)>

So, in our example, < method> is http, meaning that the browser should use HTTP (Hypertext Transfer Protocol); <host> is www.example.com; and <apURL> is "/", meaning the top directory of the host. Using HTTP/1.1, your browser might send the following request:

```
GET / HTTP/1.1
Host: www.example.com
```

The request arrives at port 80 (the default HTTP port) on the host www.example.com. The message is again in three parts: a method (an HTTP method, not a URL method), that in this case is GET; the Uniform Resource Identifier (URI) "/"; and the version of the protocol we are using.

Press Esc to exit full screen

Clip slide

4 of 21

Starting ,Stopping ,and Restarting Apache:

- ✓ If you installed Linux ,to check whether apache is running use the command

<http://localhost/>

it will the test page , If not check whether Apache is running by the command

psax|grephttpd - This command will show the running processes

#/etc/init.d/httpd start

#/etc/init.d/httpdstatus -show the running processes with PID

#/etc/init.d/httpd start –start apache

#/etc/init.d/httpd stop –stop apache service

#/etc/init.d/httpd graceful – easiest way of starting apache

- ✓ Several directives in the Red Hat httpd.conf file configure logging. The ErrorLog directive defines the path of the error log file. Use the error log to detect failures. Error logs are in **/var/log/httpd**

#/etc/init.d/httpd start

#/etc/init.d/httpd stop

Another way for restarting apache :

- ✓ **#/etc/init.d/httpd graceful**
- ✓ **# killall -USR1 httpd**- Kill all may not be installed on your system. The parameter USR1 is graceful way to reload process that allows the process and its children to exit after serving existing requests before starting again.

CONFIGURATION:

Apache was designed to be modular and can run as lean or as bloated a webserver as we want. All the directives are described at httpd.apache.org/docs/mod/directives.html but during linux installation ,the documents were placed in **/var/www/html/mod/directives.html**

Modifying the Default Configuration:

Apache's configuration file is `/etc/httpd/conf/httpd.conf`

Modify the following:

ServerAdminroot@webserver.example.com

To

ServerAdminwebmaster@example.com

Apache Logging:

Apache logs every hit to the webserver. Just as important as errors, the logs provide information about who is using your server, how much it is being used, and how well it is servicing the users. Use the transfer log to monitor activity and performance. The log includes the following

- ✓ The client(Web Surfer(IP) address
- ✓ The date
- ✓ The URI requested
- ✓ The referer –the web page the client was at when they clicked link to take them to the web page
- ✓ The user agent agent(the browser he client is using)

CustomLog:

- ✓ There are various customlogoptions, logging more or less information .
- ✓ The default logged in **/var/log/httpd/** include some of this information .
- ✓ If more information has to be needed then change the following

CustomLog /var/log/httpd/logs/acces_log commonto

CustomLog /var/log/httpd/logs/acces_log combined

- ✓ Other options include

CustomLog /var/log/httpd/logs/acces_log agent

CustomLog /var/log/httpd/logs/acces_logreferer

Securing Apache:

- ✓ Linux/Unix operating systems have the ability to multitask in a manner similar to other operating systems. However, Linux's major difference from other operating systems is its ability to have multiple users.
- ✓ Linux was designed to allow more than one user to have access to the system at the same time. In order for this multiuser design to work properly, there needs to be a method to protect users from each other. This is where permissions come in to play.
- ✓ Make sure the user and group set to

User apache
Group Apache

- ✓ If apache were cracked could some one to crack your box from root Apache account this is called **rooting your box.**
 - ✓ Create a user with **useradd**, with a locked account to run Apache
- Remove Online Manuals:**

- ✓ Apache manuals were installed in the html directory **/var/www/manual/** , which can be accessed via **file://var/www/html/manual/** or **http://localhost/manual/** or **www.your_web_site.com/manual** .
- ✓ A cracker could gain information about our machine and installation by simply hitting the directory.
- ✓ Move the manuals someplace out of the web path:
`#mv /var/www/html/manual/ usr/doc/apache-1.3.24/`

Consider Allowing Access to Local Documentation:

Red Hat defines the following by default

```
Alias /doc/ /usr/share/doc/  
<Location /doc>  
Order deny,allow  
Deny from all  
Allow from localhost, .localdomain  
Options Indexes FollowSymLinks  
</Location>
```

- ✓ This directive allows access to the local documentation in /usr/share/doc. Even though this directive allows access to these files only from localhost and .localdomain.

Don't allow Public_html Web Sites:

- ✓ The mod_usr module allows users to serve Web content without having access to the main web directory tree. For example, the user jr1 could create a directory called public_html in his home directory, which would be available at the URL <http://server/~jr1/>.
- ✓ To consider whether to allow users to create these public_html sites.

Modify the following:

UsrDirpublic_html to UsrDir disabled

.htaccess:

- ✓ We can allow access control of individual directories with the following configuration module.

```
AccessFileName .htaccess  
<Files ~".ht">  
Order allow,deny  
Deny from all  
</Files>
```

- ✓ The **AccessFileName** directive defines the name of the file apache looks at to determine whether the client can view the web page or other parts of the site.
- ✓ The **Files** directive says that files beginning with **.ht** can't be seen by anyone even if they type the filename into their browser.
- ✓ **Order** and **Deny** determine how access is controlled. The last command takes precedent. The rule here is to deny every thing except that specifically allowed.

Remove server-status and server-info:

- ✓ The following directives allow clients to find out information about our machine and server. There's no reason to give crackers any more information than necessary.

Server-status:

```
#<Location /server-status>
# SetHandler server-status
# Order deny ,allow
# Deny from all
# Allow from .your _domain.com
#</Location>
```

Server- _info:

```
#<Location /server-info>

# SetHandler server-status
# Order deny ,allow
# Deny from all
# Allow from .your _domain.com
#</Location>
```

- ✓ If we want to allow this information to be given out, change **your_domain.com** to the specific sites we want to have access.

Disallow Symbolic Links:

- ✓ Allowing symbolic links from within your webserver document tree to other directories can cause content control problems.
- ✓ To disallow symbolic links, be sure that the **Options** directives do not include **FollowSymLinks**.

Do not allow Directory Indexes:

- ✓ If we add **Indexes** to the **Options** directive, clients can access directory listings if they type in a directory with no index.html.
- ✓ This is generally a bad idea because it lets people look at the directory structure, perhaps to see files that we don't want to be served up, that is, .htaccess files, old versions, backups. Be sure **Indexes** is not part of our **Options** directive.

Don't Be a Proxy Server Unless you want to be:

If we don't want to be a proxy server, make sure the following sections are commented out:

```
#LoadModuleproxy_module modules/libproxy.soAnd
```

```
#AddModulemod_proxy.c
```

Disable CGI programs:

- ✓ CGI scripts are an excellent way to get our system cracked so disable any CGI script that were shipped with apache as follows,

```
chmod -x /var/www/cgi-bin/* remove them as
```

```
#rm -rf /var/www/cgi-bin/*
```

Reload the configuration:

After securing Apache ,reload the configuration file as follows

```
#/etc/init.d/httpd graceful
```

CREATE THE WEBSITE:

After installed and configured the basic setting we can use this as a model for our own system. There are two approaches just download entire web site are create our own examples using some favorite editors.

Downloading the Examples:

- ✓ Download all the examples ,go to www.opensoucecewebook.com/sourcecode/. Follow the instructions on how to obtain a username/password. We will need the password to download the source and view all the examples and enter the username/password. Save the tarball in a convenient place,say in **/tmp**.

Now , as **root** execute the following commands:

```
#mv /var/www/ /var/www.old
```

```
#mkdir /
```

```
#chownjrl/var/www
```

```
#tar xzv /tmp/source.tar.gz
```

```
#find .exec chownjrl {} \;
```

- ✓ First,the web site that came with the Red Hat is moved to /var/ww.old. Then , a new directory is made for the downloaded source, and this new directory is modified to be owned by jrl.
- ✓ Then the source is untarred in the new directory. And finally, all files in the new directory are changed to be owned by jrl.

- ✓ Check for our own website <http://localhost>. When the location <http://www.localhost/> is loaded in the browser, the server will locate a file named index.html in the document root.
- ✓ This default name, Index.html is configurable in the Apache configuration/index.html file.

Creating them yourself:

- ✓ First create an HTML file in `/var/www/html/index.html`. Place the following text in index.html.

```
<!Document HTML PUBLIC “ - //W3C//DTD HTML 4.0
Transitional // EN”
“http: //www.w3.org/TR/REC-html140/loose.dtd”>
<html>
<head>
</head>
<body> hello world</body>\</html>
```

- ✓ Set the permissions so that we can write to the file and rest of the world can read it.

```
$ chmoda+r /var/www/html/index.html
```

- ✓ Reload the browser with <http://localhost/> it will show the html index page (hello world) that we create.
- ✓ Creating web pages is simple as creating directories and .html files.

APACHE LOG FILES

Apache keeps detailed logs of accesses to our web site errors, and more. The apache logs are located at `/var/log/httpd/access_log` and `/var/log/httpd/error_log`. These locations are configurable in `httpd.conf`.

It's a good idea to have a log monitor program running, such **swatch** or **logwatch** to keep an eye on these files for security violations and problems.

Log entries might include IP address of the requestor, the date and time of the request, the browser used, the language(en), and some system details of the requestor.

Access Control with .htaccess:

- ✓ This is useful for restricting access to certain portions of our website, either by allowing access only from specific IP addresses or domains or by password control.
- ✓ In `.htpasswd`, look for the line `Directory /var/www/html` and
- ✓ Modify the following command **AllowOverride None** to **AllowOverrideAuthConfig**.

- ✓ This change tells the sever to change its behavior from allowing anyone to connect to allowing only those clients whose attributes match those in an authorization file to connect the files in that directory.
- ✓ We could change the name of the **.htaccess**file via this directive
AccessFileName .htaccess
- ✓ Since we use the file **.htaccess** it denies serving any file whose name begins with **.ht** , meaning that clients cant' look at our **.htaccess** file to figure out what that file is and who you allow to look at this directory.
- ✓ Now, restart the server **#etc/init.d/httpd/graceful**

To see how **.htaccess** works,create a directory for some private information

```
$ cd /var/www/html
$ mkdir private
$ chmod+rx private
$cd private
```

Create a simple index.html file as

```
<html>
<head>
<title> My private Directory </title>
</head>
<body> Congratulations your now have access to my private directory
</body></html>
```

Create a password file in the same directory and call it like .htpasswd.

```
$mkdir /var/www/misc
$chmod+rx/var/www/misc
$cd /var/www/misc
```

Create a password file: **\$htpasswd -bcprivate.passwords neo Anderson**

The option **-b** means we are supplying the password on the command line and **-c** means create the file. To add new users , leave off the **-c**.

Create the **.htaccess** file in the **/private** directory. This is not the password file but the file that points tothe password.

```
$cd /var/www/html/private
$vi .htaccess
```

The file **.htaccess** has this in it,

AuthName "My Private Area"
AuthTypeBasic
AuthUserFile/var/www/misc/private.passwords
AuthGroupFile/dev/null
require valid-user

We can such passwords and be aware of vulnerabilities caused by crackers. And monitor access these directories with our log monitoring program.

We can also do simple IP verification by putting the following in our .htaccess file

Order deny,allow
Deny from all
Allow from 192.168.1.100
Allow from 10.0.1.0
Allow from 127.0.0.1

- ✓ In this example ,the first two IP addresses are special local networks.
- ✓ Although most IP addresses are assigned by ICANN and distributed by DNS,the 192.168 and 10.0 IP subnets are not unique and are used for internal networks behind a firewall.
- ✓ The third IP address, 127.0.0.1 is a special IP address, that of localhost.Everyone's computer assigns this IP to itself.
- ✓ If we point our web browser to localhost or 127.0.0.1 it will serve up the default page of the local apache host.
- ✓ We can combine password and IP verification for additional security.

Unit –III

Perl

Perl:

Perl is a general-purpose programming language originally developed for text manipulation and now used for a wide range of tasks including system administration, web development, network programming, GUI development, and more.

Perl is a high level interpreted and dynamic programming language. Perl supports both the procedural and Object-Oriented programming. Perl is a lot similar to C syntactically and is easy for the users who have knowledge of C, C++. Since Perl is a lot similar to other widely used languages syntactically, it is easier to code and learn in Perl. Programs can be written in Perl in any of the widely used text editors like Notepad++, gedit, etc.

- Perl Originate as a Text Pocessing Language.
- Perl is a Stable ,Cross Platform Programming Language.
- Perl is a Open Source Software.Pperl works with HTML,XML,& other Markup languages.it support Unicode.

Perl Documentation:

perldoc is also the name of the Perl command that provides "access to all the documentation that comes with Perl", from the command line.

Man page:

form of software documentation usually found on a Unix or Unix-like operating system, invoked by issuing the man command. Perl documentation is sometimes available as man pages.

There are several man pages

- \$ man perl
- \$man perlop
- \$man perlsyn

Running Perl program

- To run a Perl program from the Unix command line:

```
perl file name.pl
```

- A Perl script or program consists of one or more statements. These statements are simply written in the script in a straightforward fashion. There is no need to have a `main()` function or anything of that kind.

- Perl statements end in a semi-colon:

```
Print "hello,world";
```

- Comments start with a hash symbol and run to the end of the line

```
#This is a comment
```

- Numbers don't need quotes around them:

```
Print 42;
```

- You can use parentheses for functions' arguments or omit them according to your personal . They are only required occasionally to clarify issues of precedence.

```
Print("hello,world\n");
```

```
Print "hello,world\n";
```

Perl variable :

Perl has three main variable types: scalars, arrays, and hashes.

- **Scalars**

Scalar values can be strings, integers or floating point numbers, and Perl will automatically convert between them as required. There is no need to pre-declare your variable types, but you have to declare them using the `my` keyword the first time you use them. (This is one of the requirements of `use strict`;))

scalar represents a single value:

```
My $name="camel";
```

```
My $a=42;
```

- **Arrays**

An array represents a list of values:

```
my @animals = ("camel", "llama", "owl");
```

```
my @numbers = (23, 42, 69);
```

```
my @mixed = ("camel", 42, 1.23);
```

Arrays are zero-indexed. Here's how you get at elements in an array:

```
print $animals[0];      # prints "camel"
print $animals[1];      # prints "llama"
```

The special variable `$#array` tells you the index of the last element of an array:

```
print $mixed[$#mixed];  # last element, prints 1.23
```

- **Hashes**

A hash represents a set of key/value pairs:

```
my %fruit_color = ("apple", "red", "banana", "yellow");
```

You can use whitespace and the `=>` operator to lay them out more nicely:

```
my %fruit_color = (
    apple => "red",
    banana => "yellow",
);
```

To get at hash elements:

```
$fruit_color{"apple"};    # gives "red"
```

You can get at lists of keys and values with `keys()` and `values()`.

```
my @fruits = keys %fruit_color;
my @colors = values %fruit_color;
```

Conditional and looping constructs

Perl has most of the usual conditional and looping constructs

if

```
if ( condition ) {  
    ...  
} elsif ( other condition ) {  
    ...  
} else {  
    ...  
}
```

while

```
while ( condition ) {  
    ...  
}
```

for

Exactly like C:

```
for ($i = 0; $i <= $max; $i++) {  
    ...  
}
```

The C style for loop is rarely needed in Perl since Perl provides the more friendly list scanning foreach loop.

foreach

```
foreach (@array) {  
    print "This element is $_\n";  
}  
  
print $list[$_] foreach 0 .. $max;  
  
foreach my $key (keys %hash) {  
    print "The value of $key is $hash{$key}\n";  
}
```

The foreach keyword is actually a synonym for the for keyword.

Builtin operators and functions

Perl comes with a wide selection of builtin functions. Some of the ones we've already seen include print, sort and reverse. A list of them is given at the start of [perlfunc](#) and you can easily read about any given function by using `perldoc -f functionname`.

Perl operators are documented in full in [perlop](#), but here are a few of the most common ones:

Arithmetic

```
+ addition  
  
- subtraction  
  
* multiplication
```

/ division

Numeric comparison

== equality

!= inequality

< less than

> greater than

<= less than or equal

>= greater than or equal

String comparison

eq equality

ne inequality

lt less than

gt greater than

le less than or equal

ge greater than or equal

(Why do we have separate numeric and string comparisons? Because we don't have special variable types, and Perl needs to know whether to sort numerically (where 99 is less than 100) or alphabetically (where 100 comes before 99).

Boolean logic

&& and

`||` or
`!` not

(and, or and not aren't just in the above table as descriptions of the operators. They're also supported as operators in their own right. They're more readable than the C-style operators, but have different precedence to `&&` and friends. Check [perlop](#) for more detail.)

Miscellaneous

`=` assignment
`.` string concatenation
`x` string multiplication (repeats strings)
`..` range operator (creates a list of numbers or strings)

Regular expressions

The `//` matching operator is documented in [perlop](#). It operates on `$_` by default, or can be bound to another variable using the `=~` binding operator (also documented in [perlop](#)).

More complex regular expressions

<code>.</code>	a single character
<code>\s</code>	a whitespace character (space, tab, newline, ...)
<code>\S</code>	non-whitespace character
<code>\d</code>	a digit (0-9)
<code>\D</code>	a non-digit
<code>\w</code>	a word character (a-z, A-Z, 0-9, <code>_</code>)
<code>\W</code>	a non-word character

[aeiou]	matches a single character in the given set
[^aeiou]	matches a single character outside the given set
(foo bar baz)	matches any of the alternatives specified
^	start of string
\$	end of string

Quantifiers:

Quantifiers express quantity. Quantifiers can be used to specify how many of the previous thing you want to match on, where "thing" means either a literal character, one of the metacharacters listed above, or a group of characters or metacharacters in parentheses.

These are the Perl Quantifiers.

*	zero or more of the previous thing
+	one or more of the previous thing
?	zero or one of the previous thing
{3}	matches exactly 3 of the previous thing
{3,6}	matches between 3 and 6 of the previous thing
{3,}	matches 3 or more of the previous thing

Some brief examples:

/^\d+/	string starts with one or more digits
/^\$/	nothing in the string (start and end are adjacent)

`/(\d\s){3}/` three digits, each followed by a whitespace

character (eg "3 4 5 ")

`/(a.+)/` matches a string in which every odd-numbered

letter is a (eg "abacadaf")

```
while (<>) {  
    next if /^$/;  
    print;  
}
```

Files and I/O

You can open a file for input or output using the `open()` function.

```
open(my $in, "<", "input.txt") or die "Can't open input.txt: $!";
```

```
open(my $out, ">", "output.txt") or die "Can't open output.txt: $!";
```

```
open(my $log, ">>", "my.log") or die "Can't open my.log: $!";
```

You can read from an open filehandle using the `<>` operator. In scalar context it reads a single line from the filehandle, and in list context it reads the whole file in, assigning each line to an element of the list:

```
my $line = <$in>;
```

```
my @lines = <$in>;
```

UNIT-IV

MySQL

MY SQL:

My sql is an open source (GPL)general public license,Standard Query Language(SQL) database that is fast,reliable,easy to use,and suitable for applications of any size.SQL is the ANSI-standard database query language used by most databases.

My sql can easily be integrated into perl programs by using the perl DBI.DBI is an application program interface(API)that allows perl to connect to and query a number of SQL databases.

Database:

Database is as a container for related tables.A table is a collection of rows,each row holding data for one record,each record containing chunks of information called fields.

What can you do with MySQL?

mSQL is a database, where the data is stored also we can retrieve, use data for our need. We can store data in tables, indexes can be created, we can query the data using SQL. Generally, mSQL is used to store the data from the internet, to achieve this we need to write an application. mSQL is an RDBMS, some of the features of RDBMS are constraints, triggers, stored procedures, and views.

Understanding MySQL

The most popular open-source database in the world is mSQL. It is very powerful and simple to set up and easy to use. Once we have done the setup and ready to use we can connect to it as superuser with the client.

In the shell, we need to give this command to connect with the root(superuser)

MySQL -u root -p

We can perform many different operations using mSQL like create, delete a database, insert a record all this is possible using simple commands.

How does it make working so easy?

Let us discuss how it makes working so easy.

- mSQL can support multiple storage engines whereas other systems like SQL server supports only one storage engine
- mSQL's performance is high compared to other relational database management systems.
- mSQL works on many platforms, so it is easy to deploy and use. Where MS SQL Server runs only on the windows platform.

The SHOW database CREATE database Commands:

How to get a list of all databases present?

We can get a list of all running databases in MySQL using the below query

Query: Mysql>SHOW DATABASES;

Database
My sql
Test

2 rows in set (0.00)sec

We need to create new database.check the current databases to make sure a database of that name doesn't already exist;then create the new one ,and verify the existence of the new database:

Mysql>CREATE DATABASE people;

Query ok,1 row affected(0.00 sec)

My sql>SHOW DATABASES;

Database
My sql
people
Test

3 rows in set(0.00sec).

CREATE->is a command

DATABASE->is a sub command,both are case sensitive.

USE command:

Before anything can be done with the newly created database,mysqlhas to connect to it,That's done with the USE command.

Mysql>USE people;

The CREATE TABLE command:

- Each table within the database must be defined and created .This is done with the CREATE TABLE command .
- Create a table named ageinformation to contain an individual's first name,last name, and age.MYSQL needs to know what kind of data can be stored in these fields.

Mysql>CREATE TABLE ageinformation(last name CHAR(20),first name CHAR(20),age INT

Query ok,0 rows affected (0.00sec).

- It appears that the table was created properly,but this can be checked by executing the SHOW TABLES command.If an error is made ,the table can be removed with DROP TABLE.

The SHOW TABLE command:

Mysql>SHOW TABLES;

Tables in people
Ageinformation

The DESCRIBE Command:

The DESCRIBE Command gives information about the fields in a table.The fields created earlier-lastname,first name,and age-appear to have been created correctly.

Mysql>DESCRIBE ageinformation;

field	Type	Null	key	default	Extra
Last name	Char(20)	Yes		Null	
First name	Char(20)	Yes		Null	
age	Int(11)	Yes		Null	

The INSERT command:

We need to add information to it. we do so with the INSERT command:

```
Mysql>INSERT INTO ageinformation(lastname,firstname,age)values('wall','larry',46);
```

Query ok,1 row affected(0.00sec)

The SELECT command:

SELECT selects records from the database. when this command is executed from the command line, Mysql prints all the records that match the query.

```
Mysql>SELECT * FROM age information;
```

Last name	First name	Age
wall	Larry	46

1 row in set(0.00sec)

- The * means “show values for all fields in the table”,
- From specifies the table from which to extract the information.

```
Mysql>INSERT INTO age information(last name,first name,age)values('torvalds','linus',31);
```

Query ok, 1 row affected(0.00 sec)

```
Mysql>SELECT * FROM ageinformation;
```

Last name	First name	Age
Wall	Larry	46
Torvalds	Linus	31
raymond	Eric	40

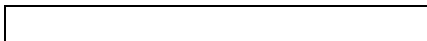
There are many ways to use the SELECT command-its very flexible.

```
Mysql>SELECT * FROM ageinformation ORDER BY lasstname;
```

Last name	First name	age
Raymond	Eric	40
Torvalds	Linus	31
Wall	Larry	46

```
Mysql>SELECT lastname FROM ageinformation ORDER BY lastname;
```

Last name
Raymond
Torvalds
Wall



The UPDATE command

To change the value in an existing record, we can update the table.

```
Mysql>UPDATE ageinformation SET age=47 where lastname='wall';
```

```
Query ok,1 row affected(0.00sec)
```

```
Rows matched:1 changed:1 warnings:0
```

```
Mysql>SELECT * FROM ageinformation;
```

Last name	First name	Age
Raymond	Eric	40
Torvalds	Linus	31
Wall	Larry	47

```
3 rows in set (0.00sec)
```

The DELETE command:

Sometimes we need to delete a record from the table. This is done with the DELETE command:

```
Mysql>DELETE FROM ageinformation WHERE lastname='raymond';
```

```
Query ok,1 row affected(0.00sec)
```

```
Mysql>SELECT * FROM ageinformation;
```

Last name	First name	Age
Torvalds	Linus	31
Wall	Larry	47

UNINT-V

PHP

PHP:

- PHP is a recursive acronym for "PHP: Hypertext Preprocessor".
- PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.

- It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.
- PHP supports a large number of major protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time.
- PHP is forgiving: PHP language tries to be as forgiving as possible.
- PHP Syntax is C-Like.

Common uses of PHP:

- PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
- PHP can handle forms, i.e. gather data from files, save data to a file, through email you can send data, return data to the user.
- You add, delete, modify elements within your database through PHP.
- Access cookies variables and set cookies.
- Using PHP, you can restrict users to access some pages of your website.
- It can encrypt data.

Characteristics of PHP:

Five important characteristics make PHP's practical nature possible –

- Simplicity
- Efficiency
- Security
- Flexibility
- Familiarity

EMBEDDING PHP INTO HTML:

- There are several ways to embed PHP code into HTML documents. one way is to put the PHP code within the tags `<?.....?>`:

```
<?echo "hello ,world!"?>
```

- In XML the `<?....>` construct is a special thing called a processing instruction. if PHP and XML are needed in the same file, one can use the alternative PHP tag.

```
<?php....?>
```

```
<?php echo "hello,world!";?>
```


- Another alternative is to use the <SCRIPT>tag
<SCRIPT LANGUAGE="php">echo "hello ,world!";</SCRIPT>
- Final one <%.....%>ASP tag
<% echo "hello,world!";%>

"Hello World" Script in PHP

```
<html>

<head>
  <title>Hello World</title>
</head>

<body>
  <?php echo "Hello, World!";?>
</body>

</html>
```

It will produce following result –

Hello, World!

Commenting PHP Code:

A *comment* is the portion of a program that exists only for the human reader and stripped out before displaying the programs result. There are two commenting formats in PHP –

Single-line comments – They are generally used for short explanations or notes relevant to the local code. Here are the examples of single line comments.

```
<?
# This is a comment, and
# This is the second line of the comment

// This is a comment too. Each style comments only
print "An example with single line comments";
?>
```

Multi-lines printing – Here are the examples to print multiple lines in a single print statement

–

```
<?
# First Example
print <<<END
This uses the "here document" syntax to output
```

```
multiple lines with $variable interpolation. Note
that the here document terminator must appear on a
line with just a semicolon no extra whitespace!
END;
```

```
# Second Example
print "This spans
multiple lines. The newlines will be
output as well";
?>
```

Multi-lines comments – They are generally used to provide pseudocode algorithms and more detailed explanations when necessary. The multiline style of commenting is the same as in C. Here are the example of multi lines comments.

```
<?
/* This is a comment with multiline
   Author : Mohammad Mohtashim
   Purpose: Multiline Comments Demo
   Subject: PHP
*/

print "An example with multi line comments";
?>
```

CONFIGURATION

- ✓ If we installed Linux then Apache is already configured for PHP . To use PHP , simply name the PHP files with the **.php** extension.
- ✓ This is contrary to that used for Emperl and Mason, where we decided against using a separate extension for those files.
- ✓ The logic was to not provide too much information about our site and to keep a clean URL look through the site through an .html extension.
- ✓ There is way to configure apache so that all .html files in a directory are processed by PHP.

A COUPLE OF QUICK EXAMPLE

Create a new directory for PHP work and cd in to it

```
$mkdir /var/www/html/php
$chmod+rx /var/www/html/php
```

```
$cd /var/www/html/php
```

- ✓ Create a PHP file named hello.php

```
<html>
<head>
<title> Hello world with php</title>
</head>
<?echo "help world" ?>
</body>
</html>
```

- ✓ To view the result type URL <http://localhost/php/hello.php> will display the php file.
- ✓ Try the built-in function `phpinfo()` as follows and it will build a page with a wealth of useful information on how PHP was built, the PHP environment by type the URL <http://localhost/php/phpinfo.php>

```
<html>
<head>
<title> PHP information</title>
</head>
<?phpinfo () ?>
</body>
</html>
```

Variables:

- A variable starts with the \$ sign, followed by the name of the variable. A variable name must start with a letter or the underscore character. A variable name cannot start with a number.
- Variables are "containers" for storing information.
- In PHP, a variable starts with the \$ sign, followed by the name of the variable:

```
<?php
$txt = "Hello world!";
$x = 5;
$y = 10.5;
?>
```

Datatypes:

Data Types defines the type of data a variable can store. PHP allows eight different types of data types. The first five are called simple data types and the last three are compound data types:

Integer : Integers hold only whole numbers including positive and negative numbers, i.e., numbers without fractional part or decimal point. They can be decimal (base 10), octal (base 8) or hexadecimal (base 16). The default base is decimal (base 10). The octal integers can be declared with leading 0 and the hexadecimal can be declared with leading 0x. The range of integers must lie between -2^{31} to 2^{31} .

Example:

```
<?php
```

```
// decimal base integers
```

```
$dec1 = 50;
```

```
$dec2 = 654;
```

```
// octal base integers
```

```
$octal1 = 07;
```

```
// hexadecimal base integers
```

```
$octal = 0x45;
```

```
$sum = $dec1 + $dec2;
```

```
echo $sum;
```

```
?>
```

Output:

704

Double: Can hold numbers containing fractional or decimal part including positive and negative numbers. By default, the variables add a minimum number of decimal places.

Example:

```
<?php
```

```
$val1 = 50.85;
```

```
$val2 = 654.26;
```

```
$sum = $val1 + $val2;
```

```
echo $sum;
```

```
?>
```

Output:

705.11

1. **String** : Hold letters or any alphabets, even numbers are included. These are written within double quotes during declaration. The strings can also be written within single quotes but it will be treated differently while printing variables. To clarify this look at the example below.

Example:

```
<?php
```

```
$name = "Krishna";  
echo "The name of the Geek is $name \n";  
echo 'The name of the geek is $name';
```

```
?>
```

Output:

The name of the Geek is Krishna

The name of the geek is \$name

2. **NULL:** These are special types of variables that can hold only one value i.e., NULL. We follow the convention of writing it in capital form, but its case sensitive.

Example:

```
<?php
```

```
$nm = NULL;  
echo $nm; // This will give no output
```

```
?>
```

3. **Boolean:** Hold only two values, either TRUE or FALSE. Successful events will return true and unsuccessful events return false. NULL type values are also treated as false in Boolean. Apart from NULL, 0 is also consider as false in boolean. If a string is empty then it is also considered as false in boolean data type.

Example:

```
<?php
```

```
if(TRUE)  
    echo "This condition is TRUE";  
if(FALSE)  
    echo "This condition is not TRUE";
```

```
?>
```

Output:

This condition is True

4. **Arrays:** Array is a compound data-type which can store multiple values of same data type. Below is an example of array of integers.

```
<?php
```

```
$intArray = array( 10, 20 , 30);  
  
echo "First Element: $intArray[0]\n";  
echo "Second Element: $intArray[1]\n";  
echo "Third Element: $intArray[2]\n";
```

```
?>
```

Output:

First element:10

Second element:20

Third element:30

5. **Objects:** Objects are defined as instances of user defined classes that can hold both values and functions. This is an advanced topic and will be discussed in details in further articles.
6. **Resources:** Resources in PHP are not an exact data type. These are basically used to store references to some function call or to external PHP resources. For example, consider a database call. This is an external resource.

Operators:

Similar in perl operators.

Flow control construct:

if statement:

```
if (expression)
    statement
```

```
<?php
```

```
$num = 31;
```

```
if ($num > 0)
    echo "\$num variable is positive\n";
```

switch statement:

The `switch` statement is a selection control flow statement. It allows the value of a variable or expression to control the flow of program execution via a multiway branch. It creates multiple branches in a simpler way than using the `if`, `elseif` statements.

The `switch` statement works with two other keywords: `case` and `break`. The `case` keyword is used to test a label against a value from the round brackets. If the label equals to the value, the statement following the case is executed. The `break` keyword is used to jump out of the `switch` statement. There is an optional `default` statement. If none of the labels equals the value, the default statement is executed.

```
<?php
```

```
$domain = 'sk';
```

```
switch ($domain) {
    case 'us':
        echo "United States\n";
```

```

break;
case 'de':
    echo "Germany\n";
break;
case 'sk':
    echo "Slovakia\n";
break;
case 'hu':
    echo "Hungary\n";
break;
default:
    echo "Unknown\n";
break;
}

```

while loop:

The while is a control flow statement that allows code to be executed repeatedly based on a given boolean condition.

This is the general form of the while loop:

```

while (expression):
    statement

```

The while loop executes the statement when the expression is evaluated to true. The statement is a simple statement terminated by a semicolon or a compound statement enclosed in curly brackets.

DO..While Loop:

The do while loop is a version of the while loop. The difference is that this version is guaranteed to run at least once.

for :

The for loop does the same thing as the while loop. Only it puts all three phases, initialization, testing and updating into one place, between the round brackets. It is mainly used when the number of iteration is know before entering the loop.

```

for ($i = 0; $i < $len; $i++)

```

foreach statement:

The foreach construct simplifies traversing over collections of data. It has no explicit counter. The foreach statement goes through the array one by one and the current value is copied to a variable defined in the construct. In PHP, we can use it to traverse over an array.

```
<?php

$planets = [ "Mercury", "Venus", "Earth", "Mars", "Jupiter",
            "Saturn", "Uranus", "Neptune" ];

foreach ($planets as $item) {
    echo "$item ";
}

echo "\n";
```

break, continue statements:

The break statement is used to terminate the loop. The continue statement is used to skip a part of the loop and continue with the next iteration of the loop.

PHP Functions:

PHP functions are similar to other programming languages. A function is a piece of code which takes one more input in the form of parameter and does some processing and returns a value.

There are two parts –

- Creating a PHP Function
- Calling a PHP Function

Syntax to declare a functions:

Function function name(variable list)

{

Statements

}

The syntax to call a function

Function name(variable list);

PHP Functions returning value:

- A function can return a value using the **return** statement in conjunction with a value or object. return stops the execution of the function and sends the value back to the calling code.
- You can return more than one value from a function using **return array(1,2,3,4)**.
- Note that **return** keyword is used to return a value from a function.

```

<html>

<head>
  <title>Writing PHP Function which returns value</title>
</head>

<body>

  <?php
  function addFunction($num1, $num2) {
    $sum = $num1 + $num2;
    return $sum;
  }
  $return_value = addFunction(10, 20);

  echo "Returned value from the function : $return_value";
?>

</body>
</html>

```

This will display following result –

Returned value from the function : 30

BUILT IN PHP FUNCTIONS

Important Functions:

- Print() -outputs a string that ends up being displayed in the browser
- Die () -outputs a string and then exists the script and used to output an error message when something unexpected happens.

ARRAY FUNCTIONS

- ✓ The array_push() function inserts one or more elements to the end of an array
- ✓ The array_pop() function deletes the last element of an array.
- ✓ The array_unshift() function inserts new elements to an array. The new array values will be inserted in the beginning of the array.
- ✓ The array_shift() function removes the first element from an array, and returns the value of the removed element.
- ✓ The sort() function sorts an indexed array in ascending order.

- ✓ Use the rsort() function to sort an indexed array in descending order.
- ✓ The asort() function sorts an associative array in ascending order, according to the value.
- ✓ Use the arsort() function to sort an associative array in descending order, according to the value.
- ✓ Use the ksort() function to sort an associative array in ascending order, according to the key.
- ✓ The **krsort()** function sorts an associative array in descending order, according to the key.
- ✓ The **count()** function returns the number of elements in an array.

```
<?php
$scars=array("Volvo","BMW","Toyota");
echo count($scars);
?
```

- ✓ The **array_keys()** function returns an array containing the keys.

Return an array containing the keys:

```
<?php
$a=array("Volvo"=>"XC90","BMW"=>"X5","Toyota"=>"Highlander");
print_r(array_keys($a));
?>
```

- ✓ The **array_values()** function returns an array containing all the values of an array. The returned array will have numeric keys, starting at 0 and increase by 1. The returned array will have numeric keys, starting at 0 and increase by 1.

STRING FUNCTIONS:

- ✓ The **printf()** function outputs a formatted string. The **arg1**, **arg2**, **++** parameters will be inserted at percent (%) signs in the main string. PHP also has the **sprintf()** function which returns string in the format specified.
- ✓ The **join()** function joins array elements into a string`

```
$array=array(2,4,6,8)
$string =join(":",$array);
//$string is now "2:4:6:8"
```

- ✓ The **substr()** function returns a part of a string.

```
$string="hello, world";  
$substring=substr($string,2,6);  
//$substring is "llo,w"
```

- ✓ The **trim()** function removes whitespace and other predefined characters from both sides of a string.

PHP and MySQL:

PHP provides built in functions to connect to and query a MySQL database. This is one of the main benefits of using PHP.

MySQL Functions :

PHP functions connect to the necessary to connect the MySQL server and execute a simple database query, and display the result.

- ✓ **Mysql_connect()**: This function returns a MySQL link identifier on success or an error message on failure.

```
<$mysql=mysql_connect("localhost","apache","lampcool")  
or die("could not connect" ?>
```

- ✓ **mysql_close()** : To close the connection in mysql database we use php function `mysql_close()` which disconnect from database. `<? Mysql_close($mysql)?>`
- ✓ **mysql_db_query(-)** Execute query: This function executes a query for the specified database. And returns a result identifier on success or returns false on error.

- `$result=mysql_db_query($db,$query);`

- ✓ **Mysql_fetch_row()** –Return the number of rows selected `<? echo mysql_num_rows($result)?>`
- ✓ **Mysql_fetch_row()**- Fetch a row as an enumerated array. Fetches one row from a result-set and returns it as an enumerated array. Each column is stored as an array element indexing starting with 0.
- ✓ **Mysql_errno()** and **mysql_error ()**- Return Mysql errors

`Mysql_errno()` returns the error number of the most recent call or 0 if there was no error. `mysql_error ()` returns the error text of the most recent function call or the empty string if there was no error.

mysql_select_db() Function

- ✓ This function returns TRUE on success, or FALSE on failure.
- ✓ This function selects a database that is used by all subsequent mysql_query() function calls.
- ✓ **mysql_select_db("books") or die ("select failed.`mysql_errno());**

mysql_query() – Query a selected database

- ✓ This function performs a query against a database and returns true if successful or on error

Mysql_fetch_array() - Fetch a row as an associative array

- ✓ This function returns the next row as an associative array with the table field names as the array keys and the table values as the array values.
- ✓ It returns the fetched row or false if there are no more rows. Each column is stored as an array element, where the field name is used as the key.

Mysql_affected_rows() – Return the number of affected rows

This function returns the number of affected rows from the last INSERT, UPDATE, or DELETE MYSQL query.

Mysql_free_result - Free the result memory

This function frees the memory used by the result. If you do not call this function, all the memory used by the result is deleted when the script finishes running. This function is needed only if you are concerned about memory as your script running.

More PHP MySQL Functions:

Mysql_change_user() - Change the logged-in user

Mysql_create_db() – Create a new database

Mysql_drop_db() - Drop database

Mysql_data_seek() - Move the result pointer

Mysql_fetch_field() – Get field from a query result

Mysql_field_name() - Get the name of the field

Mysql_field_table() - Get the name of the table the field is in

Mysql_field_type() – Get a field's type

