8KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
(Established Under Section 3 of UGC Act, 1956)
Coimbatore-21

SYLLABUS

DEPARTMENT OF CS, CA & IT

SUBJECT NAME: OPEN SOURCE TECHNOLOGIES **SUBJECT CODE: 17CAU504B**

SEMESTER: V **CLASS: III BCA**

Scope

This course teaches issues associated with Open Source technologies, characteristics of key Open Source, the nature of Open Source communities and development processes, and the implications for businesses interested in making use of Open Source technologies.

Objectives

The students would

- come to know the importance of Open source in the development of software
- come to know the role of open source in development and distribution of higher quality, better reliability, more flexibility, and lower cost software.
- learn the open source ideals in order to apply those principles

Unit-I

Why open source, what is Open source, open source principles, open standards requirements for software, open source successes, free software, some example of free software, free software license provider, free software Vs Open source software, Public Domain , FOSS DOES not Mean any cost, proprietary Vs Open Source Licensing Model.
Principles and Open Source Methodology: History, open source initiatives, open standards principles-methodologies, philosophy, software freedom, open source software, development, Licenses, copyright.

Unit-II

Open source projects: Starting and maintaining an open source project, open source hardware- open source design-open source teaching (OST).Open Source Ethics: Open Source Vs Closed Source-

Open source Government-The ethics of open source-social and financial impacts of open source technology-shared software, shared source.

Unit-III

Apache Web Server: Introduction-Starting, Stopping, and Restarting Apache-Configuration-Securing Apache Create the Web Site-Apache Log Files

Unit-IV

MySQL: Introduction-Tutorial-Database Independent Interface-Table Joins-Loading and Dumping a Database

Unit-V

Perl: Introduction-Perl Documentation-Perl Syntax Rules-A Quick Introduction To Object-Oriented Programming-What We Didn't Talk About

Suggested Readings

1. Andrew M. St. Laurent, (2004). *Understanding Open Source and Free Software Licensing*, O'Reilly Media.
2. Dan Woods, Gautam Guliani, (2005). Open Source for the Enterprise, O'Reilly Media.
3. Fadi P. Deek and James A. M. McHugh,(2007). *Open Source Technology and Policy*, Cambridge University Press.
4. James Lee, Brent Ware,2002. *Open Source Web Development with LAMP: Using Linux, Apache, MySQL, Perl, and PHP*, , Publisher: Addison Wesley Date
5. Kailashvadera, Bhavyesh Gandhi, (2009). *Open source Technologies, Lakshmi Publications*,(1st ed.).
6. Nick Wells,(2012). *The Complete Guide to Linux System Administration*, Delmar Cengage Learning.

KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
(Established Under Section 3 of UGC Act, 1956)
Coimbatore-21

SYLLABUS

DEPARTMENT OF CS, CA & IT

STAFF NAME: K. GEETHA

SUBJECT NAME: OPEN SOURCE TECHNOLOGIES **SUBJECT CODE: 17CAU504B**

SEMESTER: V **CLASS: III BCA**

S.No.	Lecture Duration (Period)	Topics to be Covered	Support Materials
		Unit – I	
1.	1	Introduction to Open Source, Why Open Source, Open Source Principles	T1: 1-8,W1
2.	1	Open Standards Requirements for Software, Open Source Successes, Free Software	T1: 8-12, W2
3.	1	Free Software License Provider, Free Software Vs Open Source Software	T1: 12-15 R1: 35-49
4.	1	Public Domain, Proprietary Vs Open Source Licensing Model	T1: 15- 20
5.	1	History of Open Source, Open Source Initiative, Open Standard Principles	T1: 22-28, W1
6.	1	Methodologies, Philosophy, Software Freedom	T1: 28-36, R4: 164-171
7.	1	Open Source Software Development	T1: 36-42
8.	1	Licenses, Copyright	T1: 42-52

9.	1	Recapitulation and Discussion of important questions	
		Total No. of Hours Planned for Unit-I	**9**
		Unit – II	
1.	1	Starting and Maintaining an Open Source Project, Open Source Hardware	T1: 129-132
2.	1	Open Source Design, An Introduction to Open Source Hardware Development	T1: 132-136, W1
3.	1	Open Source Hardware Roadmap, Open Source Hardware Organizations	T1: 136-139
4.	1	Open Source Teaching, Learning Objects, Driving Forces	T1: 139-145, W2
5.	1	Open Source Vs Close Source, Commercialization, End User Support, Innovation	T1: 147-154
6.	1	Compatibility and Interoperability, Open Source Government	T1: 154-157
7.	1	The Ethics of Open Source, Social and Financial Impacts of Technology on Government	T1:157-165, R1:148-159
8.	1	Shared Software, Shared Source	T1: 165-169
9.	1	Recapitulation and Discussion of important questions	
		Total No. of Hours Planned for Unit-II	**9**
		Unit – III	
1.	1	Introduction to Apache Web Server	T2: 33-35, W4
2.	1	Starting an Apache Server	T2: 36-37, W5
3.	1	Stopping & Restarting Apache Server, Configuration	T2:37-39, W5
4.	1	Modifying the default configuration, Securing Apache	T2: 39-43
5.	1	Remove Server- Status & Server – info	T2: 43-45, W3
6.	1	Creating the Website, Downloading the Examples	T2: 46-47

7.	1	Creating them yourself	T2: 48-49, W5
8.	1 1	Apache Log Files, Access Control with .htAccess	T2: 50-55
9.	1	Recapitulation and Discussion of important questions	
		Total No. of Hours Planned for Unit-III	**9**
		Unit – IV	
1.	1	Introduction to MySQL, Tutorial – MySQL	T2: 109-110, W6
2.	1	SHOW Databases, CREATE Databases, USE Command, The CREATE TABLE and SHOW TABLES Commands	T2: 111-114
3.	1	The SELECT , INSERT and DESCRIBE Commands	T2: 114-117
4.	1	The UPDATE and The DELETE Commands	T2: 117-119, W7
5.	1	Some Administrative Details, Summary of MySQL Commands	T2: 119-121, W6
6.	1	Database Independent Interface	T2: 122-127
7.	1	Table Joins	T2: 127-129
8.	1	Loading and Dumping a Database	T2: 129-130, W7
9.	1	Recapitulation and Discussion of important questions	
		Total No. of Hours Planned for Unit-IV	**9**
		Unit – V	
1.	1	Introduction to Perl, Perl Documentation	T2: 57-59,W8
2.	1	Perl Syntax Rules, Declaring Variables with use Strict	T2: 59-63, W9

3.	1	Scalar, Array and Hash Variables	T2: 63-71
4.	1	Operators	T2: 71-76
5.	1	Flow Control Constructs	T2: 77-81, W9
6.	1	Regular Expressions, Functions	T2: 81-92
7.	1	File I/O, Additional Perl Constructs	T2: 93-103
8.	1	Making Operating System Calls, A Quick Introduction to Object Oriented Programming, What we didn't talk about	T2: 103-107
9.	1	Recapitulation and Discussion of important questions	
10.	1	Discussion of previous ESE question papers	
11.	1	Discussion of previous ESE question papers	
12.	1	Discussion of previous ESE question papers	
Total No. of Hours Planned for Unit-V			**12**
Total No. of Hours Planned for this Syllabus			**48**

SUPPORT MATERIALS

T1: Kailashvadera, Bhavyesh Gandhi, (2009). Open Source Technologies, Lakshmi Publications,(1st ed.).

T2: James Lee, Brent Ware,2002. Open Source Web Development with LAMP: Using Linux, Apache, MySQL, Perl, and PHP, , Publisher: Addison Wesley Date

R1: Andrew M. St. Laurent, Understanding O p e n Source and Free Software Licensing, O'Reilly Media.

WEBSITES:

W1: https://opensource.com

W2: https://opensource.org

W3: https://www.apache.org

W4: https://www.wordpress.org

W5: https://httpd.apache.org

W6: www.mysql.com

W7: www.mytutorials.com

W8: www.perldoc.com

W9: www.perl.com

Unit-I

Open Source

Why Open Source:

With the many business and government organizations that now use open source software such as Linux, it's becoming increasingly clear that price is not the only advantage such software holds. If it were, companies that adopted it during the Great Recession would surely have switched back to the expensive proprietary stuff as soon as conditions began to ease, and that's clearly not the case. Rather, free and open source software (FOSS) holds numerous other compelling advantages for businesses, some of them even more valuable than the software's low price.

What Is Open Source?

Open-source software (OSS) is computer software with its source code made available with a license in which the copyright holder provides the rights to study, change, and distribute the software to anyone and for any purpose.

Open Source or Open Source Software is different from proprietary software. In open source, the source code used in software is made available to anyone to examine, evaluate and adapt it.

Eg: Linux, Apache, BSD, Wikipedia, Mozilla

Open Source assured the following rights:

- The right to make copy of programs and distribute those copies
- The right to access the software's source code
- The right to make improvements to the program

The promises of open source are

- Better Quality
- Higher Reliability
- More Flexible
- Lower Cost
- Transparent

Open Source Principles:

The following are the principles of open source software. The distribution terms of open-source software must comply with the following criteria:

1. Free Redistribution

The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.

Rationale:

By constraining the license to require free redistribution, we eliminate the temptation for licensors to throw away many long-term gains to make short-term gains. If we didn't do this, there would be lots of pressure for cooperators to defect.

2. Source Code

The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost preferably, downloading via the Internet without charge.

Rationale:

We require access to un-obfuscated source code because you can't evolve programs without modifying them. Since our purpose is to make evolution easy, we require that modification be made easy.

3. Derived Works

The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

Rationale:

The mere ability to read source isn't enough to support independent peer review and rapid evolutionary selection. For rapid evolution to happen, people need to be able to experiment with and redistribute modifications.

4. Integrity of The Author's Source Code

The license may restrict source-code from being distributed in modified form *only* if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.

Rationale:

Encouraging lots of improvement is a good thing, but users have a right to know who is responsible for the software they are using. Authors and maintainers have reciprocal right to know what they're being asked to support and protect their reputations.

5. No Discrimination Against Persons or Groups

The license must not discriminate against any person or group of persons.

Rationale:

In order to get the maximum benefit from the process, the maximum diversity of persons and groups should be equally eligible to contribute to open sources. Therefore we forbid any open-source license from locking anybody out of the process.

6. No Discrimination Against Fields of Endeavor

The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

Rationale:

The major intention of this clause is to prohibit license traps that prevent open source from being used commercially. We want commercial users to join our community, not feel excluded from it.

7. Distribution of License

The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

Rationale:

This clause is intended to forbid closing up software by indirect means such as requiring a non-disclosure agreement.

8. License Must Not Be Specific to a Product

The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.

Rationale:

This clause forecloses yet another class of license traps.

9. License Must Not Restrict Other Software

The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.

Rationale:

Distributors of open-source software have the right to make their own choices about their own software.

10. License Must Be Technology-Neutral

No provision of the license may be predicated on any individual technology or style of interface.

Rationale:

This provision is aimed specifically at licenses which require an explicit gesture of assent in order to establish a contract between licensor and licensee. Provisions mandating so-called "click-wrap" may conflict with important methods of software distribution such as FTP download. CD-ROM anthologies, and web mirroring; such provisions may also hinder code re-use.

Open Standards Requirement for Software

An "open standard" must not prohibit conforming implementations in open source software.

The Criteria

To comply with the Open Standards Requirement, an "open standard" must satisfy the following criteria. If an "open standard" does not meet these criteria, it will be discriminating against open source developers.

1. **No Intentional Secrets:** The standard MUST NOT withhold any detail necessary for interoperable implementation. As flaws are inevitable, the standard MUST define a process for fixing flaws identified during implementation and interoperability testing and to incorporate said changes into a revised version or superseding version of the standard to be released under terms that do not violate the OSR.
2. **Availability:** The standard MUST be freely and publicly available (e.g., from a stable web site) under royalty-free terms at reasonable and non-discriminatory cost.
3. **Patents:** All patents essential to implementation of the standard MUST:
    1. be licensed under royalty-free terms for unrestricted use, or
    2. be covered by a promise of non-assertion when practiced by open source software

No Agreements: There MUST NOT be any requirement for execution of a license agreement, NDA, grant, click-through, or any other form of paperwork to deploy conforming implementations of the standard.

No OSR-Incompatible Dependencies: Implementation of the standard MUST NOT require any other technology that fails to meet the criteria of this Requirement.

Open Source Successes:

1. More on server than client side

- More users of servers
- Server operating systems
    - Linux

- Certain Server software

    - Web
    - Mail
    - DNS

- So far less so with other server software

- Database (MySQL rising, less easy to measure)

2. Software for technical users.

Free Software: What is Free Software?

Free software is software that can be freely used, modified, and redistributed with only one restriction: any redistributed version of the software must be distributed with the original terms of free use, modification, and distribution

“Free software” means software that respects users' freedom and community. It means that the users have the freedom to run, copy, distribute, study, change and improve the software. Thus, “free software” is a matter of liberty, not price. To understand the concept, you should think of “free” as in “free speech,” not as in “free beer”. Free software is sometimes called as “libre software” to show we do not mean it is gratis.

We campaign for these freedoms because everyone deserves them. With these freedoms, the users (both individually and collectively) control the program and what it does for them. When users don't control the program, we call it a “non free” or “proprietary” program..

A program is free software if the program's users have the four essential freedoms:

- Freedom 0: The freedom to run the program as you wishes, for any purpose.
- Freedom 1: The freedom to study how the program works, and change it so it does your computing as you wish. Access to the source code is a precondition for this.
- Freedom 2: The freedom to redistribute copies so you can help your neighbor.
- Freedom 3: The freedom to distribute copies of your modified versions to others By doing this you can give the whole community a chance to benefit from your changes. Access to the source code is a precondition for this.

Some Examples of Free Software:

Operating systems and Desktop environments

Linux - Operating system kernel

Ubuntu - Linux distribution with full compliment of software for everyday use. See also **Kubuntu** that runs the KDE desktop which some people find this easier if they previously

used MS Windows. Ubuntu is an easy to use operating system based on and made possible by the **Debian** project.

Google Chrome OS - Lightweight operating system based around the web browser

Android smart-phone operating system - by Google / Open Handset Alliance

Symbian smart-phone operating system - by Nokia

MeeGo smart-phone operating system - joint venture between Intel and Nokia

BSD - Operating system

Darwin - The core of Apple's Mac OS X operating system

GNOME desktop environment for Linux (and Unix)

KDE desktop environment for Linux (and Unix)

Xfce lightweight desktop environment for Linux (and Unix)

Graphics and multimedia

GIMP - Bitmap graphics editor, similar to Adobe Photoshop

Inkscape - Vector graphics editor

VNC - cross-platform remote desktop utility. For example it enables a Mac to control a Windows or a Linux PC (or vice-versa)

Blender - Advanced 3D modelling and rendering application. Gallery

Art of Illusion - Java-based 3D modelling and rendering application with an intuitive user interface. Gallery

VideoLAN - cross-platform media player capable of playing most video filetypes

Miro video player - cross-platform media player that can automatically download videos from RSS-based "channels" and play videos sequentially from a playlist

Songbird - similar to iTunes with built-in browser. Screencast

Audacity - digital audio editor

Ardour - digital audio workstation

Scribus - Desktop Publishing (DTP)

F-Spot - Photo manager

Office software

OpenOffice.org - office productivity software. Comparable to Microsoft Office. As well as having using an open file format it can read and write Microsoft Office files

NeoOffice - Mac OS X version of OpenOffice

KOffice - office suite for KDE desktop (Unix / Linux)

AbiWord - word processor

Evolution - personal information manager

PDFCreator - creates PDFs from any Windows program. Use it like a printer (Windows only)

Internet related software

Apache webserver - web server

BitTorrent - distributed file-sharing

Mozilla Foundation

Mozilla Firefox - web browser

Mozilla Thunderbird - mail client

WebKit - HTML rendering engine developed by KDE, Apple and Google. Core of Safari, Google Chrome and Konqueror browsers

Google Chrome - Google's web browser

Sendmail - email transfer agent

Pidgin (software) - multiple-protocol instant messaging

Diaspora - distributed social software

Content management systems

Joomla! - CMS

Alfresco - CMS

MediaWiki - wiki content management software. AdCiv.org is powered by MediaWiki

Drupal - modular content management framework and blogging engine

Plone - python-based CMS

Other

Celestia 3D space simulation software. For a tour of its capabilities, select 'Run demo' from the 'Help' menu. Gallery and addons

NASA worldwind - virtual Earth / Moon / Mars software

FlightGear - flight simulator

Second life - virtual world viewer (as in a virtual reality, rather than a virtual Earth model)

FreeMind - mind mapping software

Stellarium - planetarium software for accurate representations of the night's sky from any location or time

K3b - CD and DVD authoring application for Linux

phpMyAdmin - web front-end for MySQL database management system

Vega Strike - Space flight simulator and trading game

Wine - a compatibility layer for computers running Linux that enables them to run many applications that were originally written for MS Windows

Freemat - environment for rapid engineering and scientific prototyping and data processing

Programming related

Eclipse - software framework and Java IDE

GTK - Popular widget toolkit for the X Window System, for creating graphical user interfaces

QT toolkit - cross-platform application development framework

KDevelop - Programming IDE for Linux / Unix

PHP - server-side programming language

PERL - Dynamic programming language

Python - versatile, clean and powerful programming language used for cross-platform desktop applications, server-side scripting for websites, and scripting within java and .net environments

GNU Compiler Collection (GCC)

MySQL - Database management system

Java - Programming language

Mono - Free and open-source crossplatform .NET implementation

OpenCV - Real-time computer vision function library

Free Software License Provider:

The GNU General Public License:

The **GNU General Public License** (**GNU GPL** or **GPL**) is a widely used free software license, which guarantees end users (individuals, organizations, companies) the freedoms to run, study, share (copy), and modify the software. Linux Kernel and GNU Compiler collection (GCC) released under this license

The GNU Lesser General Public License:

The GNU Lesser General Public License (LGPL) is a free software license published by the Free Software Foundation (FSF). The license allows developers and companies to use and integrate software released under the LGPL into their own software. The license only requires software under the LGPL be modifiable by end users via source code availability.

BSD licenses:

BSD licenses are a family of permissive free software licenses, imposing minimal restrictions on the redistribution of covered software. This is in contrast to copyleft licenses. BSD licenses are sometimes referred as BSD- Style. The original was used for Berkeley Software Distribution (BSD), The Unix- like operating system for which the license was named.

Apache License :

The Apache License (ASL) is a permissive free software license written by the Apache Software Foundation (ASF). Like other free software licenses, the license allows the user of the software the freedom to use the software for any purpose, to distribute it, to modify it, and to distribute modified versions of the software. This license is incompatible with GPL.

The "as-is" Release Model:

This means that the release code is simply placed in the public domain with no restriction. There are actually a few dedicated purpose libraries released under this license, which do something very specific and very efficiently.

Free Software Vs Open Source Software

Some of the differences in the terms Free Software and Open Source Software.

Open Source Software:

For software to be considered "Open Source" it must meet ten conditions. Of these ten conditions, it's the first three that are really at the core of Open Source and differentiates it from other software. Those three conditions are;

- **Free Redistribution: the software can be freely given away or sold.**
- **Source Code: the source code must either be included or freely obtainable.**
- **Derived Works: redistribution of modifications must be allowed.**

The other conditions are:

- Integrity of The Author's Source Code: licenses may require that modifications are redistributed only as patches.
- No Discrimination Against Persons or Groups: no one can be locked out.
- No Discrimination Against Fields of Endeavor: commercial users cannot be excluded.
- Distribution of License: The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.
- License Must Not Be Specific to a Product: the program cannot be licensed only as part of a larger distribution.
- License Must Not Restrict Other Software: the license cannot insist that any other software it is distributed with must also be open source.
- License Must Be Technology-Neutral: no click-wrap licenses or other medium-specific ways of accepting the license must be required.

Free Software :

Unlike the Open Source term, Free Software only has 4 "Freedoms" with its definition and are numbered 0-3:

- The freedom to run the program for any purpose (Freedom 0)
- The freedom to study how the program works and adapt it to your needs (Freedom 1)
- The freedom of redistribution of software (Freedom 2)
- The freedom to improve the program and release your improvements to the public to benefit the while community. (Freedom 3)

Although not explicitly outlined as a freedom, access to source code is implied with Freedoms 1 and 3. You need to have the source code in order to study or modify it.

Public Domain:

Public domain is a designation for content that is not protected by any copyright law or other restriction and may be freely copied, shared, altered and republished by anyone. The designation means, essentially, that the content belongs to the community at large. Copyright restrictions vary among types of content and different countries.

Content in the public domain is usually there because either the copyright or restriction has expired or because it was never protected, often because the content owner has purposely placed it in the public domain. Early silent films, for example, are usually in the public domain because their copyrights have expired. Musicians sometimes release works directly into the public domain so that it may be freely accessed.

Music, films, literature, images, software, patents and mathematical formulas are among the wide variety of intellectual property that may be in the public domain.

FOSS DOES NOT Mean any Cost:

This is a common misunderstanding about FOSS, in no small part because nearly all FOSS programs are available free of charge. For example when the text editor Emacs was first released Richard Stallman charged from time to time to get copies. Developers have the choice to charge under most FOSS licenses, although they rarely choose to. The only requirement to be a truly FOSS project is that the publisher provides the source code with the program, and to allow the user to edit that code.

On top of the initial cost of purchasing software, there are other ongoing costs associated with all software. This can come in the form of support agreements, the cost of customization, training costs support personnel and other sources. This is true of both traditional commercial software and FOSS programs. There is a large and active debate about which type of software is more expensive over the long run for large corporations, for individual users there is little to no question that FOSS is cheaper by far.

proprietary vs open source licensing model:

Proprietary	Open Source
Licensor distributes object code only; Source code is kept safe.	Licensor distributes source code
Modifications are prohibited	Modifications are permitted
All upgrades, support and development are done by the licensor.	Licensee may do its own development and support or hire any third party to do it.
Fees are for the software license, maintenance and upgrades	Fees if any are for integration, packing, support and consulting.
Sublicensing is prohibited or is a very limited right	Sublicensing is permitted
Example: Microsoft Windows	Example: Wikipedia

Principles and Open Source Methodology:

History:

Open Source S software dogma takes behind its development processes the termed collective invention. This proposes that when firms collectively invent, they make available to their competitors the result of new plant designs so that their competitors can incorporate extensions of those designs into new facilities. (Robert C Allen 1983, 1). Although this dogma does not

concern open source but the main idea remains that when people of the same interest comes together to achieve a common goal, they can affect substantially the course of events. George Siemens 2003 explains that the same goes for open source software developments which have been said to be fostered by: Hackers Culture influence on the history of the Internet

- Transition from open to commercial software development
- Richard Stallman- Free software Foundation
- Linus Torvalds- the Linux Kernel
- Formation of OS Initiative

The internet and the whole computer industry are been said to be based on an open culture right after the World war II funded by agencies, administered with software standards and government purchasing rules, the United States Federal government has helped stimulate open source software and open standards for decades (Nathan Newman, 1999). The steady flow of innovations and free software fuelled the accelerated enactment of the Internet and open source ideology simultaneously. According to George Siemens 2003, the followings timeline shows the evolvement of the Internet: 20 1957 - USSR launches Sputnik. The US government responds by creating the Advanced
Research Projects Agency (ARPA).

- 1960 – IBM have free software in the early 1960s
- 1961 – MIT Tech Model Railroad club acquired the first PDP- 1 and invented programming tools, slangs and a culture and first adopt the word “hackers” Kim Johnson (2001 in George Siemens 2003).
- 1965 - 1969 - Development and design of ARPANET (an experimental network for research and development of networking technologies - the birthing place of the Internet) (Nathan Newman , 1999). 1972 - ARPANET email
- 1970 - UNIX grows rapidly due to favorable licensing terms with universities (UNIX is an operating system developed at Bell Laboratories with the intent of allowing multiple users simultaneous access to the computer). Multiple versions of Unix were developed - most were proprietary.
- 1991 – Linus Torvalds decided to create a Unix kernel that he could make use alone but later open it to the community for use and further development and named Linux (Eric S. Raymond 1999).
- 1983 - TCP/IP adopted by ARPANET as a means of connecting various networks (the Internet is essentially a network of networks communicating via TCP/IP) 1984-1985 – Richard Stallman founded the GNU project (GNU not Unix) which is a Unix like operating system

- 1986 - NSFNET - five supercomputers connected to ARPANET allowing for rapid development of connections - particularly universities
- 1991 - Tim Berners-Lee creates vision for World Wide Web
- 1993 - Mosaic introduced as "an Internet information browser and World Wide Web client"
- 1994 – Netscape made an attempt to commercialize the internet
- 1995 - Netscape released
- 1995 - Microsoft release Internet Explorer. All of these developments will not have happen if not for the free sharing of codes and programs norms adopted by highly committed programmers.

Open Source Initiatives:

Abbreviated *OSI*, a non-profit organization that promotes the integrity of the open source definition by certifying products with the OSI Certified Open Source Software mark. OSI is comprised of board members who make up its directorship and is not a membership organization

The **Open Source Initiative** (**OSI**) is an organization dedicated to promoting open-source software.The organization was founded in late February 1998 by Bruce Perens and Eric S. Raymond, part of a group inspired by the Netscape Communications Corporation publishing the source code for its flagship Netscape Communicator product. Later, in August 1998, the organization added a board of directors.

Open Standard Principles:

An Open Standard is more than just a specification. An open standard is a standard that is publicly available and has various rights to use associated with it, and may also have various properties of how it was designed

The Open Source Principles are:

1. Availability:

Open Standards are available for all to read and implement.

- The best practice is for the standards text and reference implementation to be available for free downloads via the Internet. Thus

- Any software project should be able to copy without any expense.
- Licenses attached to the standards documentation must not restrict any party from implementing the standard using any form of software license.
- The best practice is for the software reference platforms to be licensed in a way that is compatible with all forms of software licensing, both free and proprietary software.

2. Maximize End-User Choice:

Open Standards create a fair, competitive market for implementations of the standard. They do not lock the customer in to a particular vendor or group. Thus

- They must allow wide range of implementations, by businesses, academic and public projects.
- They must support a range of pricing from very expensive to zero price.

3. No Royalty:

Open standards are free for all to implement, with no royalty or fee. Certification of agreement by the standards organization must have a fee. Thus

- Patents embedded in standards must be licensed royalty-free.
- Certification Programs should include a low to zero cost self- certification.

4. No Discrimination:

Open Standard and the organizations that administer them do not favor one implementer over another for any reason other than the technical standards of a vendor's implementation.

- A standard organization that wishes to support itself through certification branding should establish a premium track and a low-cost to zero-cost track.

5. Extension or Subset

Implementations of Open Standards may be extended, or offered in subset form. However, certificate organizations may decline to certify subset implementations, and may place requirements upon extensions.

6. Predatory Practices:

Open Standards may employ license terms that protect against subversion of the standard by embrace-and-extend tactics. The license may require the publication of reference information and license to create and redistribute software compatible with the extensions. It may not prohibit the implementations of extensions.

- The standards organization may wish to apply an agreement similar to the Sun Industry Standards Source License to the standard documentation and its reference implementation. The Sun agreement requires publication of a reference implementation for any extensions to the standard. This makes it possible for a standards organization to actively preserve interoperability.

7. Modularity:

Modularity is the principle of maintain well defined boundaries between components. Adhering to open standards it is a noticeable principle in open source software because it helps to define the boundaries.

8. Early and Often:

Early and often lead to making things available as early as possible. This includes not only the software but also plans and design. Many open source project achieve the early and often using a couple of the following common methods.

- **Source Code Availability:**

  The source code of the software is made available either on a daily as a zip file or lives in a source code repository such as CVS or Subversions. Community members using these systems have to compile or build the software from its source code. This used by a small percentage of the community.

- **Milestones builds:**

  On a periodic basis the administrators or developers of the project compile the software and make resulting executables for downloading. These downloads are used by the members of community that are interested in the functionality of the software. Many open source projects release milestones builds.

- **Nightly builds:**

On a nightly basis the source code is compiled to generate the executable binaries. These builds are made available for community members want to test the latest source code.

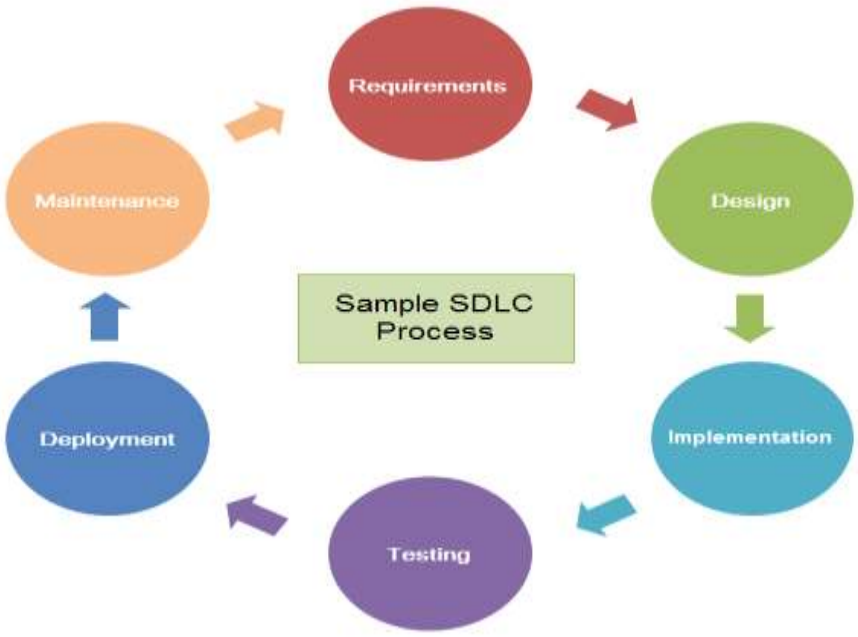
SOFTWARE ENGINEERING METHODOLOGIES:

1. Engineering Software:

Software engineering methodologies are the framework that tells us how to develop a software system. These frameworks define different phases of the development process, such as planning, requirement analysis, design, development, testing and maintenance of software in a systematic method. The choice of which methodology to use in a development project is based on the size of the software system and the environment it is to function in.

2. The Life- Cycle Paradigm:

The Software **development life cycle** (**SDLC**), also referred to as the application development life-cycle, is a term used in systems engineering, information systems and software engineering to describe a process for planning, creating, testing, and deploying an information system. The work is started at the system level and passes of analysis, design, coding, testing and maintenance. Six activities represent the overall development



process.

- **System Engineering and Analysis:** This activity is characterized by system level requirements gathering for all system elements.

- **Software Requirements Analysis:** This activity is usually executed together with the customer to collect the software requirements. The main goal of this phase is to document all function, performance and interfacing requirements for the software.

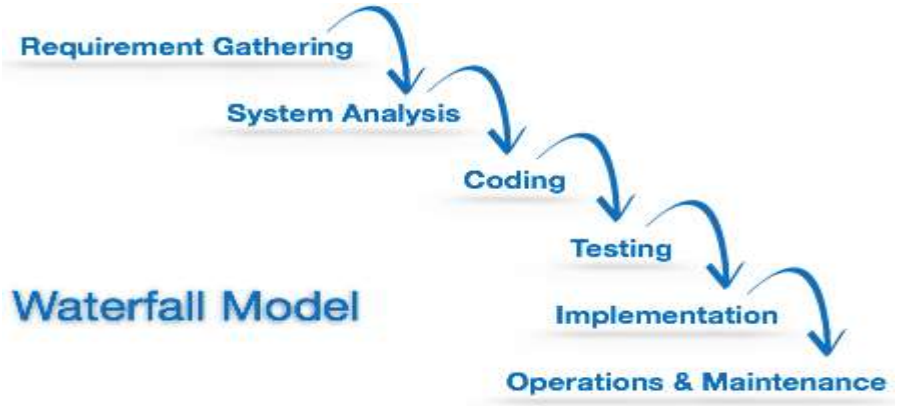
- **Design:** When creating the design of the software system, requirements are transformed into a representation of software that can be assessed for quality before the actual design begins.

- **Coding:** This step is also known as programming phase. The implementation of software design starts in terms of writing program code in the suitable programming language and developing error-free executable programs efficiently. This activity is the transition of the design specification into the software program.

- **Testing:** It is the process of executing a program or application with the intent of finding the software bugs.

- **Maintenance:** It is the modification of a software product after delivery to correct faults, to improve performance or other attributes. A common perception of maintenance is that it merely involves fixing defects.

The life-cycle paradigm of software engineering sometimes referred as "Water fall model". Waterfall model is the simplest model of software development paradigm. It says the all the phases of SDLC will function one after another in linear manner. That is, when the first phase is finished then only the second phase will start and so on.



This model assumes that everything is carried out and taken place perfectly as planned in the previous stage and there is no need to think about the past issues that may arise in the next phase. This model does not work smoothly if there are some issues left at the previous step. The sequential nature of model does not allow us go back and undo or redo our actions.

3. The Prototyping Paradigm:

Different to the static, procedural approach offered by the life-cycle paradigm, the prototyping model can be used. Similarly to the life-cycle paradigm, the process begins by gathering requirements of the system. The developers meet with the customers; determine the overall objective of the software.

In prototyping quick iteration are planned and quick modeling occurs. The quick design emphasizes on a representation of those features of the software visible to the customer or end-user. In a simple language the input forms and output reports are quickly designed by the software engineers and are shown to the customers.

The design model is then implement a first prototype, which may take one of three forms.

- **Iteration Prototype:** This is a paper or computer software prototype that makes it possible for users to understand how to interact with the software.
- **Subset function Prototype:** This is a working software program that implements a subset of the required functionality.
- **Existing Program:** This is an program that implements most or all of the required functionality.
- **A throwaway prototype:** This is may be usable software program, but it is not suitable as the final soft ware product for various reasons such as poor performance, maintainability and overall quality.
- **A prototype to refine and deliver:** This kind of prototype is enhanced and possibly reworked in various areas. so that it is suitable to deliver as the final software product.

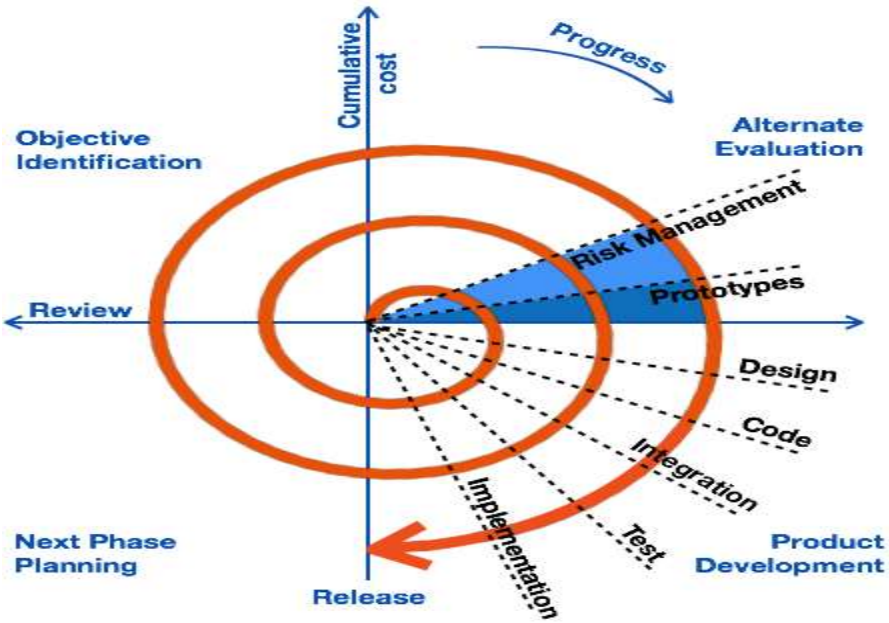
When the prototype is created, it is then assessed by the customer or user. An important thing is feedback from the customer is used to refine requirements for the software.

The prototyping is tuned and iterated till the customer requirements are satisfied. At the same time developers perfectly understands what the customers' needs are and how to fulfill them.

The prototype mainly serves a mechanism for identifying software needs.

4. Spiral Model

Spiral model is a combination of both, prototyping model and one of the SDLC model. The spiral model adds an element of risk analysis to the development process.



This model is presented as a spiral. There are four major activities represents each iteration:

- **Planning:** Determine objectives and constraints of the project, and define the alternatives.
- **Risk Analysis:** Analysis of alternatives, and identification and resolution of risks.
- **Engineering:** Development of the next-level product.
- **Customer Evolution:** Evaluation of the product engineered.

Philosophy:

Open source is a development methodology; free software is a social movement. For the free software movement, free software is an ethical imperative, essential respect for the users' freedom. By contrast, the philosophy of open source considers issues in terms of how to make software "better"—in a practical sense only.

Software Freedom:

Study of the GNU General Public License (herein, abbreviated as *GNU GPL* or just *GPL*) must begin by first considering the broader world of software freedom. The GPL was not created

in a vacuum. Rather, it was created to embody and defend a set of principles that were set forth at the founding of the GNU Project and the Free Software Foundation (FSF) – the preeminent organization that upholds, defends and promotes the philosophy of software freedom. A prerequisite for understanding both of the popular versions of the GPL (GPLv2 and GPLv3) and their terms and conditions is a basic understanding of the principles behind them. The GPL families of licenses are unlike nearly all other software licenses in that they are designed to defend and uphold these principles.

A particular user has software freedom with respect to a particular program if that user has the following freedoms:

- The freedom to run the program, for any purpose.
- The freedom to study how the program works, and modify it
- The freedom to redistribute copies.
- The freedom to distribute copies of modified versions to others.

Open Source Software Development:

In his 1997 essay *The Cathedral and the Bazaar*, open-source evangelist Eric S. Raymond suggests a model for developing OSS known as the *bazaar* model. Raymond likens the development of software by traditional methodologies to building a cathedral, "carefully crafted by individual wizards or small bands of mages working in splendid isolation". He suggests that all software should be developed using the bazaar style, which he described as "a great babbling bazaar of differing agendas and approaches."

In the traditional model of development, which he called the *cathedral* model, development takes place in a centralized way. Roles are clearly defined. Roles include people dedicated to designing (the architects), people responsible for managing the project, and people responsible for implementation. Traditional software engineering follows the cathedral model.

The bazaar model, however, is different. In this model, roles are not clearly defined. Gregorio Robles suggests that software developed using the bazaar model should exhibit the following patterns:

Users should be treated as co-developers

The users are treated like co-developers and so they should have access to the source code of the software. Furthermore, users are encouraged to submit additions to the software, code fixes for the software, bug reports, documentation etc. Having more co-developers increases the rate at which the software evolves. Linus's law states, "Given enough eyeballs all bugs are shallow." This means that if many users view the source code, they will eventually find all bugs and suggest how to fix them. Note that some users have advanced programming skills, and

furthermore, each user's machine provides an additional testing environment. This new testing environment offers that ability to find and fix a new bug.

Early releases

The first version of the software should be released as early as possible so as to increase one's chances of finding co-developers early.

Frequent integration

Code changes should be integrated (merged into a shared code base) as often as possible so as to avoid the overhead of fixing a large number of bugs at the end of the project life cycle. Some open source projects have nightly builds where integration is done automatically on a daily basis.

Several versions

There should be at least two versions of the software. There should be a buggier version with more features and a more stable version with fewer features. The buggy version (also called the development version) is for users who want the immediate use of the latest features, and are willing to accept the risk of using code that is not yet thoroughly tested. The users can then act as co-developers, reporting bugs and providing bug fixes.

High modularization

The general structure of the software should be modular allowing for parallel development on independent components.

Dynamic decision making structure

There is a need for a decision making structure, whether formal or informal, that makes strategic decisions depending on changing user requirements and other factors. Cf. Extreme programming.

Data suggests, however, that OSS is not quite as democratic as the bazaar model suggests. An analysis of five billion bytes of free/open source code by 31,999 developers shows that 74% of the code was written by the most active 10% of authors. The average number of authors involved in a project was 5.1, with the median at 2.

Licenses:

Open Source Licenses are essential in open source software and the reason being that source code must be made legally available for the developing community while making sure that they can be modified, shared and used. An open source license in particular gives the “permission to do or not to do something” according to the OSL. Under the OSL, the initial source code developers still holds the original copyright on the software but at the same time

grants permission to other developers to use the source code and prevent others from claiming your work as their own . The most popularly used OSL according to opensource.org are:

- BSD 3-Clause "New" or "Revised" license
- BSD 2-Clause "Simplified" or "FreeBSD" license
- GNU General Public License (GPL)
- GNU Library or "Lesser" General Public License (LGPL)
- MIT license
- Mozilla Public License 2.0
- Common Development and Distribution License
- Eclipse Public License

Among others, even though there are more than 70 types of open source licenses among others. The above listed licenses all have different usages and grants as well different permission for different usage. The rise in open source software's has also led to the rise in the amount of open source licenses available. Companies that choose to publish the source code and engage in open development hope to gain external contributions from other stakeholders such as customers, partners, and voluntary developers using a variety 31 of business models to gain revenue. For other stakeholders, the legal relationship is based on the open source license, under which the initial source code is published for the community OSI has a list of 10 requirements needed for a license to be called an OSL and in choosing the right license for software; the company must first decide how it is willing to make profit off the software by considering whether it's going to be selling support, selling connected hardware or commercial software.

Copyright:

Copyright law protects original creative works, such as software, video games, books, music, images, and videos. Copyright law varies by country. Copyright owners generally have the right to control certain unauthorized uses of their work (including the right to sue people who use their copyrighted work without permission). As a result, certain images and other copyrighted content may require permissions or licenses, especially if you use the work in a commercial setting. For example, even if you have permission to use an image, you may need additional permission to use what is in the image (e.g., a photo of a sculpture, a person, or a logo) because someone else's copyright, trademark, or publicity rights might also be involved. You are

responsible for obtaining all of the permissions and licenses necessary to use the content in your specific context.

However, even copyright-protected works can be lawfully used without permission from the copyright holder in certain circumstances. The Wikipedia entry on copyright law contains a useful overview of copyright law, including fair use and other exceptions to copyright law.

KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
(Established Under Section 3 of UGC Act, 1956)
Coimbatore-21

DEPARTMENT OF CS, CA & IT

UNIT- I (Objective Type/ Multiple Choice Questions Each Question Carries One Mark)

OPEN SOURCE TECHNOLOGIES (17CAU504B)

S.No	Question	Option1	Option2	Option3	Option4	Answer
1	OSS stands for _____	Open Source Software	Open Standard Software	Open Standard Service	Open Source Service	Open Source Software
2	In _____ the source code used in the software is available to anyone to examine, evaluate and adapt.	close source	open source	shared source	proprietary source	open source
3	Which of the following is not an example of open source software?	Linux	Mozilla	Windowss	Apache	Windows
4	The human readable code of the program is known as _____	source code	byte code	machine code	object code	source code
5	In _____ the source code used in the software is available to anyone to examine, evaluate and adapt.	close source	shared source	open source	proprietary source	open source
6	The human readable code of the program is known as _____	source code	byte code	machine code	object code	source code
7	_____ is a proprietary software made available free of charge	Free software	Open source	Freeware	Malware	Freeware
8	BSD stands for _____	Berkeley Software Distribution	Berkeley Source Distribution	Berkeley Software Destruction	Berkeley Source Destruction	Berkeley Software Distribution
9	Wikipedia is an example of_____ software	freeware	open source	proprietary	free	open source
10	Free software is also known as_____	open libre	software libre	free libre	source libre	software libre

11	_____ is a matter of liberty, not price	free software	freeware	close software	open source	free software
12	MySQL is a _____ software	close source	open source	proprietary	free	open source
13	GDB stands for _____	GNU Database	GNU Data bind	GNU Document b	GNU Debugger	GNU Debugger
14	BSD licenses also referred as_____	BSD style	BSD unique	BSD source	BSD type	BSD style
15	Proprietary software is also known as _____	close source	open source	shared source	free source	close source
16	In open source software the licensor distributes the _____	byte code	source code	machine code	object code	source code
17	In _____ licensor distributes the object code.	close source	open source	shared source	free source	close source
18	OSI stands for _____	Open Standard Initiative	Open Source Initiative	Open Standard Instruction	Open Source Instruction	Open Source Initiative
19	Sublicensing is prohibited in _____software.	freeware	open source	free	proprietary	proprietary
20	FSF stands for_____	Free Software Foundation	Free System Foundation	Free Service Foundation	Free Source Foundation	Free Software Foundation
21	Which of the following license the software cannot be mixed with non-free software	MIT	LGPL	GPL	BSD	GPL
22	_____ is a form of intellectual property , applicable to certain forms creative works	License	patent	copyleft	copyright	copyright
23	The term free in free software represents _____	empty	freedom	freeware	free cost	freedom
24	Berkeley Software Distribution is sometimes called _____	Berkeley Unix	Berkeley Linux	Berkeley Kernel	Berkeley Domain	Berkeley Unix
25	Life cycle paradigm of software engineering is called the _____ model.	waterfall	spiral	agile	spiral	waterfall
26	The term _____ also refers to the document that specifically describes these permissions and rights.	License	patent	copyleft	copyright	License
27	Which of the following license that can be relicensed by anyone?	BSD	MIT	public domain	LGPL	public domain

28	The reversed c in a full circle is the _____ symbol	copyright	license	patent	copyleft	copyleft
29	_____ license does not allow you to take modifications private.	MPL	LGPL	NPL	MIT	LGPL
30	_____ who participate frequently in newsgroups and discussions, but do not do any coding.	Project leaders	Maintainers	Posters	Occasional developers	Posters
31	Eric. Raymond wrote the Cathedral and the Bazaar model in _____	1998	1995	1994	1997	1997
32	_____ is the free software project funded by Netscape to build a WWW browser.	Mozilla	Internet Explorer	Chrome	Opera	Mozilla
33	_____ model adds an element of risk analysis to the software development process.	Scrum	Water fall	Spiral	Prototype	Spiral
34	_____ activity is the transition of the design specification into a software program.	Analysis	Coding	Testing	Maintenance	Coding
35	The principle of maintaining well defined boundaries between components is called _____	availability	modularity	c) Extension	subset	modularity
36	_____ is an interpreted language with lots of libraries	Java	Perl	C	C++	Perl
37	GCC stands for _____	GNU Complete Collection	GNU Complex Collection	GNU Compound Collection	GNU Compiler Collection	GNU Compiler Collection
38	The GNU project is free software, mass collaboration project, announced in 1983 by_____	Richard Stallman	Ken Thompson	Linus Torvalds	Bill Jolitz	Richard Stallman
39	Which of the following is widely used as a WWW server?	BIND name server	Send mail transport	Apache Web server	Samba file server	Apache Web server
40	L in the LGPL represents____	Local	Lesser	Legal Limit		Lesser
41	TCL stands for _____	Tools Computer Language	Tools Command Loader	Tools Command Language	Tools Command Linker	Tools Command Language
42	The c in a full circle is the _____ symbol	copyright	license	patent	copyleft	copyright

43	copyright	license	patent	copyleft		
44	In _____ fees are for the software license, maintenance and upgrades.	free source	open source	shared source	close source	close source
45	In proprietary source, all upgrades, support and development are done by a _____	licensee	licensor	User	third parties	licensor
46	In _____ the licensee may do its own development and support or hire any third party to do it.	free source	open source	shared source	proprietary source	proprietary source
47	SDLC stands for _____	Software Development Life Cycle	Source Development Life Cycle	Software Deployment Life Cycle	Source Deployment Life Cycle	Software Development Life Cycle
48	Openoffice.org was established by _____	IBM	Microsoft	Sun Microsystems	Corel	Sun Microsystems
49	_____ is a non-profit corporation dedicated to managing and promoting the open source.	FSF	OSI	DARPA	ASF	OSI
50	_____ license contains special privileges for original copyright holder over your modifications	Public Domain	LGPL	NPL	GPL	NPL
51	_____ is a non-profit corporation to support Apache software project	FSF	OSI	DARPA	ASF	ASF
52	Who have the overall responsibility for the open source software development?	project leader	Volunteer developers	everyday users	posters	project leader
53	_____ is a bug tracker written in PERL language.	Bugzilla	Request tracker	Mantis	Trac	Request tracker
54	_____ is a web-based PHP/MySQL bug tracker.	Trac	Bugzilla	Request tracker	Mantis	Mantis
55	Which of the following is sophisticated bug tracker from the Mozilla house?	Bugzilla	Mantis	GNATS	LibreSource	Bugzilla
56	XP stands for_____	External Programming	Extreme Programming	Extensible Programming	Executable Programming	Extreme Programming
57	Which phase of the Spiral model determine objectives and constraints of the project and define the alternatives?	Risk analysis	Planning	Engineering	Customer Evolution	Planning

58	_____ helps to manage the files and codes of a project when several people are working on the project at the same time.	CVS	RPM	APT	SVN	CVS
59	_____ is a computer program that is used to debug other programs.	Compiler	Interpreter	Debugger	Loader	Debugger
60	LibreSource is a type of _____	debugger	compiler	interpreter	bugtracker	bugtracker

Unit-II

Open Source Projects

Starting and Maintaining an open Source Project:

Getting Started:

Vaporware is software that is floated to see if there is interest. Many projects, if donated to the open source methodology would find usage, development and markets much quicker. Last code might be resurrected and enhanced. More and more computer users are learning to use the excellent development tools created and enhanced by programmers.

- Create a new open source project.
- Download a project we have released.
- Explore the latest from the Google Summer of code, Wikipedia and Code guru.
- Watch open source developers talk about their craft.
- Learn about the latest technologies through Wikipedia, Google and Code guru for education.

Project Management:

Version Numbers:

Many new users find the FOSS projects often use strange version number systems. For instance it is very common to see software in wide distribution that has a version number less than 1.0. For example, the web browser Firefox was widely used by version 0.6, more than a year before version 1.0 was finished.

Generally FOSS projects break their version numbers into three pieces:

1. ***Major version:***

   Major version is the first number in the version number; it is rare to see free software projects that have a major version greater than 2.

2. ***Minor Update:***

The minor update is the second number. A change in the number signifies a significant update to fix a bug, or security problem. These numbers get very high, often over 30 or 40 overtime.

3. ***Point Release:***

   When a software project is established most of their updates will come in the form of point release.

Version numbers may also carry extra meanings. For instance ever since the Linux kernel went to version 2, it has been the design practice of the design team to use odd numbers for development kernels, and even numbers for stable kernels.

When a software project changes the major version number, this generally indicates a complete re-write of the software or at least an extension over haul. The Apache web server started a version 2.0 in 2003, after many years of 1.3.x versions. Apache2 is a near total re-write of Apache, which provides many new features, and significantly improved the underlying codes base. The Apache Foundation still maintains the 1.3.x code base, although most new efforts for new code are done on the 2.0.x code base.

Writers of software may completely re-write a program, improve software or unnecessarily inflate perfectly good software. If software wors then there is little need to upgrade.

Open Source Hardware:

"Open source hardware," or "open hardware," refers to the design specifications of a physical object which are licensed in such a way that said object can be studied, modified, created, and distributed by anyone.

"Open hardware" is a set of design principles and legal practices, not a specific type of object. The term can therefore refer to any number of objects—like automobiles, chairs, computers, robots, or even houses.

Like open source software, the "source code" for open hardware—schematics, blueprints, logic designs, Computer Aided Design (CAD) drawings or files, etc.—is available for modification or enhancement by anyone under permissive licenses. Users with access to the tools that can read and manipulate these source files can update and improve the code that underlies

the physical device. They can add features or fix bugs in the software. They can even modify the physical design of the object itself and, if they wish, proceed to share such modifications.

Open hardware's source code should be readily accessible, and its components are preferably easy for anyone to obtain. Essentially, open hardware eliminates common roadblocks to the design and manufacture of physical goods; it provides as many people as possible the ability to construct, remix, and share their knowledge of hardware design and function.

Open source hardware licenses generally permit recipients of the designs and documentations to study them, redistribute and modify them, and then to distribute any modifications. Additionally, open hardware licenses don't prevent someone from giving away or even selling the project's documentation.

Like software, hardware designs and inventions are subject to copyright and patent law. And like open source software, open source hardware uses these intellectual property laws creatively to make hardware designs publicly accessible.

Both copyright law (in the case of source code and design documentation) and patent law (in the case of design processes and material technologies) apply to open hardware. Trademark law is also pertinent to the branding names and logos of open hardware.

Open Source Design:

Open design is the application of open source methods to the creation of physical products, machines and systems.

The word software refers both source code and executables, while the words hardware and hardware design clearly refer two different things.

Free Hardware Design:

It refers to a design which can be freely copied, distributed, modified and manufactured. It does not imply that the design cannot be sold, or that any hardware implementation of the design will be free of cost.

There are two design practices are available.

1. Commercial Design Practice:

- Designs are owned by the company which creates them. Ownership is protected by 3 set of laws: copyright, trade secret, and patent. It is not possible to see a design without signing a non-disclosure agreement.
- Designers cannot build on older design unless their company owns the right to use these designs.
- Where the design is for a basic building block, design software can only be written by those allowed to know the secret information. EDA (Electronic Design Automation) software is either written in-house or by software companies that have agreements with the manufacturers.
- Designs are driven by marketing departments. The two main goals are minimizing time to market and minimizing manufacturing costs.
- Users of the final product have no rights to know how they work.

2. Free Design Practice:

- Designs are owned by the people who own them. Ownership is protected by copyright law only.
- The intention is to make designs as widely available as possible.
- There is an incentive to build on older designs, to collaborate with as wide a spread of people as possible, and to make the designs widely known. NGOs in developing countries are encouraged to reuse designs.
- Design software is free software, so that anyone who wishes can participate.
- Designs are driven by the wishes of their creators. The end goal can be whatever they wish.
- Users of the final product cannot know how it works, but are encouraged to create improvements or modify it for their own purposes.
- Libre Hardware Design refers to the same class of design as free hardware design, but tries to make it clear where the word Free refers to the freedom, not price.

- Open Source Hardware refers to hardware for which all the design information is made available to the general public. Open Source hardware may be free hardware design, or the design on which it is based may be restricted in some way.
- Open hardware is a trade mark of the Open Hardware Specification Program. It is a limited from of open source hardware.
- Free hardware is a term used as a synonym for open source hardware.

Open Source Teaching:

Open Source Teaching is a platform that uses emerging technologies to provide learning and the development of communities. It makes the relationship between the leaner by eliminating the barriers of time and distance.

Open-source teaching is an emerging education practice that allows students to capitalize on the scope and power of the Internet to create and manage their own learning experiences and produce interactive material that is available online to everyone.

In an open-source learning environment, individual students work with the guidance of a teacher-mentor to explore and create concepts, source materials, and research to develop their own learning experiences, primarily with online technology. Students form socially dynamic learning networks online and in the local community, communicating and collaborating by using in-depth online research practices, blogs, social media, and other interactive tools.

As a result, students create and manage interactive learning material that is available online to everyone, generating and sharing value that extends beyond the traditional K-16 curriculum. This deeper and more engaged involvement results in significant improvement in academic achievement; it also creates many opportunities for traditional performance evaluation of objective production, including formative and summative tests, as well as alternative assessment of portfolios, which can include a variety of artifacts, including transmedia presentation of content and the learner's choices related to platforms, media, and design.

Resource Bank:

The user can select a set of required or needed learning objects from the resource bank by viewing the object directly or scanning through meta files.

Learning Object:

A learning object is the basic unit in the open source teaching that teaches a single idea or skill. It is a collection of any type of files. Each file is called as an atom. A learning object can have hyper links to other learning objects. Each learning object must have a main atom as a starting or an entry point.

Charter Group:

The charter group plays a major role in open source teaching. The new material for the resource bank for the open source teaching is selected by charter group. It has a significant task to

- Edit the material
- Apply quality control
- Ensure that material fits into the system index and
- Keep control of revisions and version of changes.

Driving Forces:

People contribute to open source because of the following

- Being the part of the center of excellence
- Make an individual more productive than working alone
- It provides access to source of good practice and material
- It leads to a workload reduction
- It provides a kite mark for the courses for the teaching quality assessment.

Citation Index:

It is essential to build a citation index into the materials. This provides recognition, benefit and further motivation to the contributors.

Meta Files:

The Meta files can contain the following data for each object.

- Name
- Caption
- Previous Knowledge required
- Level
- Learning Outcomes
- Next suitable objects to study
- Location
- File List
- Source
- Author Name

Open Source Vs Closed Source:

Closed Source	**Open Source**
Licensor distributes object code only; Source code is kept safe.	Licensor distributes source code
Modifications are prohibited	Modifications are permitted
All upgrades, support and development are done by the licensor.	Licensee may do its own development and support or hire any third party to do it.
Fees are for the software license, maintenance and upgrades	Fees if any are for integration, packing, support and consulting.
Sublicensing is prohibited or is a very limited right	Sublicensing is permitted
Example: Microsoft Windows	Example: Wikipedia

Open Source Government:

Government is committed to implement more innovative ways of working, and a clear re-use and interoperability agenda including ensuring a level playing field for open source and proprietary software. Recognizing the merits of OSS, Government takes the view that where there is no significant overall cost difference between open and non-open source products, open source should be selected on the basis of its additional inherent flexibility.

The increased maturity of open source products and services has made it easier for Government to engage with OSS. However, open source software (OSS) is only slowly gaining traction in Government, particularly when compared with the private sector and other public sectors including some European government sectors. Relatively low levels of adoption have been attributed to a lack of understanding of the potential benefits of OSS, accompanied by a risk-averse technical and procurement culture, compounded by significant levels of is conceptions about open source security and its services ecosystem.

On the whole contracts are large and encompass a large estate, this has limited the suppliers (and solutions) able to meet the requirements and to some extent has excluded SMEs and open source solutions. Contracts have therefore traditionally been awarded to SIs who have their own set of preferred (and usually proprietary) products. Their existing agreements are with proprietary software houses and existing skills are focused on proprietary products, there is not a culture of actively looking for open source software. There may also be commercial incentives for the incumbent systems integrators to work with a limited set of proprietary software vendors.

Government departments are often locked into these contacts and in most cases feel they have little scope to explore alternative open source solutions for evolving requirements within the business. A change in the mindset is required for those involved in writing requirements, including SIs, or undertaking procurement or projects. The challenge is to enable both open source and proprietary solutions to be proposed, compared and fairly assessed on merit.

Change is required in

(1) the bundling of risk and calculation of risk appetite by the customer,
(2) the diversity and competitive tension in the IT supplier market,
(3) an improvement in the intelligent customer function.

The Ethics of Open Source:

Ethics denotes the theory of right action and greater good. The field of ethics (or moral philosophy) involves systematizing, defending, and recommending concepts of right and wrong behavior. Moral has a dual meaning

1. It indicates a person's comprehension of morality and his or her ability to put it into practice

2. It denotes the limited specific acts and defined moral codes.

Types of Ethics:

Personal Ethics:

It refers to a person's personal or self-created values and codes of conduct. From the very beginning, these ethics are instilled in an individual, with a large part having been played by their parents, friends, and family.

Descriptive Ethics:

Descriptive ethics are the morals of a society. People use descriptive ethics as a way to judge particular actions as good or bad based on the social contract of a particular society. It is possible for people in one group to hold a different set of morals than people in another group. Descriptive ethics also change over time. Those working on descriptive ethics aim to uncover people's beliefs about such things as values, which actions are right and wrong, and which characteristics of moral agents are virtuous.

Meta Ethics

Meta ethics is a branch of analytic philosophy that explores the status, foundations, and scope of moral values, properties, and words. Whereas the fields of applied ethics and normative theory focus on what is moral, meta ethics focuses on what morality itself is.

Normative Ethics:

Normative ethics is the study of ethical action. It is the branch of philosophical ethics that investigates the set of questions that arise when considering how one ought to act, morally speaking.

Applied Ethics:

Applied ethics is the philosophical examination, from a moral standpoint, of particular issues in private and public life which are matters of moral judgment. It is thus the attempts to use philosophical methods to identify the morally correct course of action in various fields of everyday life.

Social and Financial Impacts of Open Source Technology:

Open source software impacts developing countries in various ways. Some impacts are positive for example, cost savings, flexibility of software, obtaining negotiation power against big software companies, fighting piracy, building its own software industry, and even increase national security by less dependence on a few foreign companies. The negative impacts would be maintaining the software quality and providing updating or service when the software environment is changed. Many international governments are increasingly supportive of the use of OSS. “Open source software is often touted to be ideal for accelerating the growth of low-income countries’ IT sectors, with the expectation that it will increase their propensity to innovate”.

In countries like China, Japan, South Korea, and India, there is political incentive toward the use of OSS. To insure commitment in the use of OSS, these governments have enacted policies and laws. In June 2002, the European Union's position on this issue was that governments (or public administrations) are not promoting OSS over proprietary software, but are optimizing investments in sharing developed software. Whereas each government has its own political motivation toward the adoption of OSS, the decision must be carefully examined. Governments, specifically in developing countries, that are quick to implement OSS over commercial software for development must take into consideration whether the choice will bring about the required end results.

The popularity of OSS is driving vendors to meet the high demands, especially from the developing countries. For example, Sun Microsystems' executives have suggested that they are considering making their entire software stack open source over time. However, future changes in the way OSS is distributed (i.e., different types of licensing fees for support and maintenance, compatibility issues with other software and hardware technologies, licensing issues in software development) will bring about major changes in structure and costs. Most open-source companies have long offered their software free and built business around value-added services and support.

A much smaller number have been selling open-source software with premium-level add-on components for years; that model is not new. But the number of companies falling into the latter category appears to be increasing, which could eventually change the underlying structure of the open-source community, as we know it. With possible changes in the marketing and applications of open source software, the need for reassessment of policies will be eminent for developing countries in order to stay in the competitive technological global market.

Shared Software:

Shared Software is a different term used to describe free software or open source software that can be shared rather than owned. Sharing is symmetric. The following are the some shared software.

1. CICS: Transaction Control Management

Customer Information Control System (**CICS**) is a family of mixed language application servers that provide online transaction management and connectivity for applications on IBM

Mainframe systems under z/OS and z/VSE. **CICS** is middleware designed to support rapid, high-volume online transaction processing.

2. EOS: Report archive and distribution

3. PL/1: Compiler and Runtime Libraries

PL/I is a third-generation (3GL) programming language developed in the early 1960s as an alternative to assembler language (for low-level computer processing functions), COBOL (for large-scale business applications), and FORTRAN (for scientific and algorithmic applications). **PL/I** stands for "Programming Language **1**".

4. COBOL: Compiler and Runtime Libraries

COBOL (Common Business Oriented Language) was the first widely-used high-level programming language for business applications. Majority of payroll, accounting and other business application programs still use COBOL.

5. DB2: Database Software

DB2 is a Relational Database Management System (RDBMS) originally introduced by IBM in 1983 to run on its MVS (Multiple Virtual Storage) mainframe platform. The name refers to the shift from the then prevalent hierarchical database model to the new relational model.

Shared Source:

Shared Source is Microsoft's mechanism for legal distribution of software source code. It permits both organizations and individuals to access a program's source code as a reference. This access provides developers with debug capabilities, which can be downloaded after satisfying certain eligibility criteria. The associated license may range from only code viewing for reference to permission for modification, which can be used for commercial and noncommercial purposes.

Benefits:

The common benefit of all shared source program is the availability of source code. The availability of source code also permits review and auditing from a security perspective.

Notable Shared Source Program:

1. Microsoft Enterprise Source Licensing Program

Microsoft gives enterprise customers viewing access to some parts of some versions of the Microsoft Windows operating systems. The ESLP license agreement is among the most restrictive of the licenses associated with shared source programs, allowing no modifications of the code.

2. Microsoft Windows Academic Program

The Windows Academic Program provides universities worldwide with concepts, Windows kernel source code, and projects useful for integrating core Windows kernel technologies into teaching and research.

3. Microsoft Government Security Program

The Microsoft Government Security Program is an effort to assist national governments in evaluating the security of Windows and of other Microsoft products. Participating governments have access to the source code for current versions of Windows and Windows service packs, Windows Embedded CE, and Microsoft Office.[23]

4. Most Valuable Professionals Source Licensing Program

Through this program, Microsoft makes Windows source code available to members of their "Most Valuable Professional" program. MVPs are members of the developer and support community who have made significant public, volunteer contributions, primarily through participation in online forums. The MVP Source Licensing Program allows licensees to use the source code for debugging and support purposes, though it may not be used to aid in the development of a commercial product.

5. Microsoft Shared Source Common Language Infrastructure

The first widely distributed Shared Source program was Shared Source CLI, the Shared Source implementation of the Common Language Infrastructure. The licensing permits non-commercial modification and distribution of the source code, as long as all distributions include the original license, or one encompassing the original terms.

6. ASP.Net AJAX Control Toolkit

The ASP.NET AJAX Control Toolkit is a set of controls and extenders that use AJAX technologies to enable developers to improve the client experience on their web sites. The toolkit is licensed under the Microsoft Public license (MS-PL) and is available on CodePlex, Microsoft's online community development portal for collaborative software development projects.

Notable Shared Source Licenses:

1. Open Source Licenses:

i) Microsoft Public License (Ms-PL)

This is the least restrictive of the Microsoft licenses and allows for distribution of *compiled* code for either commercial or non-commercial purposes under any license that complies with the Ms-PL. Redistribution of the source code itself is permitted only under the Ms-PL.

ii) Microsoft Reciprocal License (Ms-RL)

This Microsoft license allows for distribution of derived code so long as the modified source files are included and retain the Ms-RL. The Ms-RL allows those files in the distribution that do not contain code originally licensed under Ms-RL to be licensed according to the copyright holder's choosing.

2. Proprietary Licenses:

i) ***Microsoft Reference Source License (Ms-RSL)***

This is the most restrictive of the Microsoft Shared Source licenses. The source code is made available to view for reference purposes only, mainly to be able to view Microsoft classes source code while debugging. Developers may not distribute or modify the code for commercial or non-commercial purposes.

ii) Microsoft Limited Public License (Ms-LPL)

This is a version of the Microsoft Public License in which rights are only granted to developers of Microsoft Windows-based software.

iii) Microsoft Limited Reciprocal License (Ms-LRL)

This is a version of the Microsoft Reciprocal License in which rights are only granted when developing software for a Microsoft Windows platform. Like the Ms-LPL, this license is not open source because it is not technology-neutral due to its restriction that licensed software must be used on Windows, and is also not considered free by the Free Software Foundation due to this restriction.

KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
(Established Under Section 3 of UGC Act, 1956)
Coimbatore-21

DEPARTMENT OF CS, CA & IT

UNIT- II (

Objective Type/ Multiple Choice Questions Each Question Carries One Mark)

OPEN

SOURCE TECHNOLOGIES (17CAU504B)

S.No	Question	Option1	Option2	Option3	Option4	Answer
1	_____ is a software that is floated to see if there is an interest	Vaporware	Freeware	Free Software	Open Source Software	Vaporware
2	The software in wide distribution that has a version number less than _____	3.0	2.0	1.0	0.5	1.0
3	OSH stands for _____	Open Source Host	Open Source Hardware	Open Source Hub	Open Source Hierarchy	Open Source Hardware
4	The development of open source hardware was initiated in _____	2004	2000	2002	2001	2002
5	F in FOSS stands for _____	feasible	full	free	field	free
6	FOSS projects break their version number into_____ pieces	4	3	2	1	3
7	The second piece of a version number of FOSS project is_____	major version	minor update	point release	minor version	minor update
8	Minor update is the _____ piece of version number of FOSS project.	first	second	third	fourth	second
9	_____ indicates a small update to fix a bug or security problem	point release	minor update	major version	minor version	point release
10	HDL stands for _____	High Definition Language	Hardware Definition Language	High Description Language	Hardware Description Language	Hardware Description Language
11	Pajamas media briefly known as_____	free software media	open hardware media	open source media	close source media	open source media
12	Open Source Media is a start up company founded in the year_____	2001	2002	2003	2004	2004
13	OST stands for_____	Open Source Teaching	Open software Teaching	Open Service Teaching	Open Server Teaching	Open Source Teaching

14	_____ is collection of just about any type of files	Object	Learning Object	Class Object	Source Object	Learning Object
15	Each file in a learning object is referred as ____	document	group	atom	record	atom
16	Each learning object must have_____ as a starting or entry point	prototype atom	sub atom	model atom	main atom	main atom
17	In_____ model, source code must be hidden from the public competitors.	open source	close source	free source	shared source	close source
18	Free software movement demands _____ types of freedoms	4	5	6	7	4
19	_____ is microsoft's framework for sharing computer source code with individuals and organizations	open source	shared source	close source	free source	shared source
20	Richard's Stallman organization is called _____	Free Software Foundation	Open Source Software Foundation	Close Source Software Foundation	Apache Software Foundation	Free Software Foundation
21	Most of the open sourc software licensed under_____	copyright	copyleft	patent	legal license	copyleft
22	_____ denotes the theory of right action and the greater good.	Ethics	Moral	Immoral	legal license	Ethics
23	_____ ethics signifies a moral code applicable to individuals	social	personal	financial	logical	personal
24	_____ ethics can be synonymous with social, political philosophy in as much as it is the foundation of a good society or state	Personal	Financial	Logical	Social	Social
25	The Wix toolset is licensed under the _____	CPL	GPL	LGPL	MPL	CPL
26	The ASP.Net AJAX control toolkit licensed under_____	MPL	MS-PL	CPL	NPL	MS-PL
27	_____ toolkit is a set of controls and extenders use AJAX technologies to enable the developers to improve the client experience on their website	ASP.Net AJAX	ASP.Net	Wix Toolset	AJAX	ASP.Net AJAX
28	Open politics is also known as_____	open software politics	open source politics	open source software politics	free software politics	open source politics
29	In Linux kernel version 2, the design use _____ numbers for Development kernels	odd	even	binary	decimal	odd

30	In Linux kernel version 2, the design use _____ numbers for stable kernels	Random	odd	even	binary	even
31	_____ is the application of open source methods to the creation of products, machines and system	Open design	Software design	Hardware design	System Design	Open design
32	EDA stands for_____	Electronic Design Automation	Electrical Display Architecture	Electronic Device Automation	Electrical Device Architecture	Electronic Design Aut
33	Programmers often prefer to _____ from other's code.	Copy	Reuse	steal	learn	learn
34	What is CVS?	Open source license	Editing software	Version control system	Development Environment	Version control system
35	Which of the following is not true for open source software?	It is owned by a person	It supports distributed development	It supports collaborative development	Its code is available for all	It is owned by a person
36	The acronym of CVS is _____.	Consistent Version System	Common Validation System	Consistent Version System	Concurrent Versions System	Concurrent Versions System
37	_____ has advanced data store than CVS.	RCS	SCCS	SVN	IMS	CVS
38	The acronym of SVN is _____.	Single Version	Subversion	Stimuli Verification	Simple Verified Node	Subversion
39	Subversion is a project of _____.	IBM	Oracle	Apple	Apache	Apache
40	SourceSafe is a _____.	version control	backup	website	search engine	version control
41	What is Bugzilla?	A bug-tracking mechanism	A version control	A bug-correcting software	It involves both version control and bug-tracking mechanism	A bug-tracking mechanism
42	Which of the following is proprietary?	OpenOffice	Oracle	MySQL	Postgres	Oracle
43	Bugs persist longer in _____.	open codebases	proprietary codebases	free codebases	closed codebases	closed codebases
44	A software that has both proprietary license and open source license is called _____.	multiple licensed software	uncontrolled software	dual licensed software	mixed licensed software	dual licensed software

45	Software with academic license _____.	does not provide the source code to the user	requires just an acknowledgement	can be used by the companies	can be used by anyone	requires just an acknowledgement
46	Dual licensing business rely on open source as _____.	production strategy	development strategy	distribution strategy	production and distribution strategy	distribution strategy
47	Open source software are distributed mainly through _____.	CDs	Internet	Memory cards.	HDDs.	Internet
48	Dual licensing works for _____.	work having many contributors	collaboratively developed softwares	single, well defined owner of a work	open source software following development strategy	collaboratively developed softwares
49	_____ is a reciprocal license.	GPL	BSD License	Ms-RSL	Ms-LPL	GPL
50	Why software licensing revenue is considered as a good revenue?	Licensed software can be sold for more value	Licensing a software is very easy	Licensing a second copy of the software is possible without any additional cost	Licensed software are accepted good than others in the market	Licensing a second copy of the software is possible without any additional cost
51	Ownership applies to _____.	tangible properties only	both tangible and intangible properties	intangible properties only	software only	both tangible and intangible properties
52	Which clarifies the issues and resolves disputes among author and reader?	License	Trademark	Warranty	Copyright law	Copyright law
53	Which license enforce sharing?	reciprocal license	Academic license	Berkley Software Distribution	Proprietary license	reciprocal license
54	Which of the following is an academic license?	PUL	GPL	SPL	BSD	BSD
55	Proprietary licenses _____ for that consideration.	have some rights and pay a fee	have no rights and pay little fee	have some rights and pay no fee	have no rights and pay no fee	have some rights and pay a fee
56	Making a software available on open source terms, creates it with a _____.	small and expensive distribution channel	large and inexpensive distribution channel	large and expensive distribution channel	small and inexpensive distribution channel	large and inexpensive distribution channel

57	Reciprocity encourages _____.	isolation	distribution	collaboration	Installed base	collaboration
58	What is warranty?	Rules	Promise	Rights	Offers	Promise
59	Which of the following provides clear warranties?	Proprietary software	Open source software.	Free software	As is' software	Proprietary software
60	Which one of the following can be patented?	Hardware	Software	Things	Names	Software

Unit- III

Apache Web Server:

Introduction:

The web pages you see when you surf the Web (quit slacking, you!) are served up via the Hyper Text Transfer Protocol (HTTP) by an httpd daemon—the "d" at the end means *daemon*, programs that are always running in the background. In some distributions, the daemon is called apache instead of httpd. Currently, Apache is the web server of choice, and not just for Open Source bigots. As of this writing,

Apache has more than 60 percent of the active site web server market. Because it is so widely used, it is widely tested, and when a bug is discovered or a new Web feature is implemented, bug fixes and updates are almost instantaneous. Apache has a BSD-type Open Source license, making it attractive for both commercial and noncommercial applications. Its modular architecture makes it feasible to tailor.

Apache web server recognizes an HTTP request by the URL of the thing requested or by the filename extension. For instance, If the URL www.example.com/content/chapter1/ were loaded into a browser, the web server contacted would receive a request that might look like this:

This example demonstrates the simpler HTTP protocol version 1.0. It is more likely that the version used will be 1.1, but 1.0 still works, and because it is simpler, we use it here.

GET /content/chapter1/ HTTP/1.0

The server determines that the thing requested is underneath the document root, a directory where the HTML files reside.

For the examples in this book, that is /var/www/html. The text /content/chapter1/ directs Apache to navigate to those directories underneath the document root and grab the HTML file named index.html (by default, the server looks for the file with this name, but this is configurable, as are most things related to Apache).

The result is that the server grabs the file /var/www/html/content/chapter1/index.html, which is simply a text file. It then takes the content of this file and pretends an important piece of information called the header. The header tells the client how to interpret the information that is to follow. For an HTML file, the header tells the client that what follows is text, which is to be interpreted as HTML code. The header is separated from the content that follows by a blank line.

Of course, web servers can dish up more than HTML these days: music, streaming video, PDF, etc. It's an instructive exercise to view the header, blank line, and body that the server serves up, and this can be achieved without using a browser. This can be done in a shell window.

This example connects to a server and asks for index.html in the directory /content/chapter1/:

```
$ telnet www.not_a_real_web_server.com 80
Trying 1.299.299.1
Connected to www.not_a_real_web_server.com (1.299.299.1)
Escape character is ´^]´.
GET /content/chapter1/ HTTP/1.0 HTTP/1.1 200 OK
Date: Thu, 17 Jan 2002 19:57:05 GMT
Server: Acme Web Server Version 0.001b
Connection: close
Content-Type: text/html
```

When the server accepts the connection, it tells the client so. Then we make the HTTP request: GET /content/chapter1/ HTTP/1.0 followed by a blank line. The web server prints out some header stuff, including the content type text/html, followed by a blank line, followed by the contents of the HTML file

Starting, Stopping, and Restarting Apache:

If you installed Linux, Apache should be running when you start your machine. To check, load this URL in your browser: http://localhost/ You should see the Apache welcome page, as shown in Figure A. If not, Apache may not be running. Not a major crisis—in fact, it's a good thing if daemons such as Apache don't run unless you explicitly start them. If Apache has been running since you booted and plugged in the TCP/IP connection, other services are probably running, and you should configure them as you wish, firewall them, or turn them off. To check whether Apache is running, try the following:

Figure: Apache welcome Page

ps ax | grep httpd

Although the program is called Apache, the daemon's name is httpd on Red Hat, apache on some others. You should see something like this:

```
1922 ?    S    0:00 /usr/sbin/httpd
1927 ?    S    0:00 /usr/sbin/httpd
1928 ?    S    0:00 /usr/sbin/httpd
1929 ?    S    0:00 /usr/sbin/httpd
1930 ?    S    0:00 /usr/sbin/httpd
1932 ?    S    0:00 /usr/sbin/httpd
1933 ?    S    0:00 /usr/sbin/httpd
1935 ?    S    0:00 /usr/sbin/httpd
1937 ?    S    0:00 /usr/sbin/httpd
```

Several copies of the server are running, so Apache can process more than one request at a time. If you don't see a number of httpd PIDs, start the server as follows:

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

In addition to using the ps command, you can check the status of your server by executing the following command:

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

The output of this command should resemble this:

Reformatted to fit the page. We often have to do this, so don't sweat small differences like this.

```
httpd (pid 1937 1935 1933 1932 1930
      1929 1928 1927 1922) is running...
```

If you didn't get this sort of result, or you got an error message, review the logs in /var/log/httpd for error messages, specifically error_log, and read the man page (man httpd, not man apache). Unfortunately, the number of possible errors is great, and we can't possibly cover all of them—the logs are key to narrowing down the problem.

If httpd is running, and you didn't see the welcome page when you loaded http://localhost/, you have a problem.

First, try restarting it. There are several ways to do this—the easiest is to use the provided start-up script:

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

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

Also good to know is this:

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

The option help is not a defined option, but if you pass an invalid option to /etc/init.d/httpd, it will tell you all the valid ones.

You can also do this—it sends the USR1 signal to all occurrences of httpd, making the daemon reload the configuration file:

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

or:

```
# killall -USR1 httpd
```

Killall may not be installed on your system. If it isn't, consider finding it and installing it; it's invaluable. It saves the step of finding the process ID and passing that to the kill command. The parameter USR1 (man kill and man signal) is the graceful way to reload a process—it allows the process and its children to exit after serving existing requests before starting again. If all else fails, there's always kill -9.

The gentlest way to restart apache is to use the graceful option to the start-up script. Therefore, hereafter when we need to restart the server (usually to reread the configuration file), we will execute the following command:

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

If all this doesn't work, you may need to adopt sterner measures. See the Apache documents at httpd.apache.org/docs/. The Linux Documentation Project at www.linuxdoc.org/ is another excellent place to waste a few hours. Another excellent way to figure out how to fix errors, or to find a support group to commiserate with, is to paste the error (enclosed in quotes) into Google or Google's group interface. Once you have Apache running, make sure it starts at boot. Execute chkconfig:

```
# chkconfig --list httpd
```

```
httpd  0:off  1:off  2:off  3:on  4:on  5:on  6:off
```

If you do not see output indicating that 3, 4, and 5 are on, turn them on:

```
# chkconfig httpd on
```

```
# chkconfig --list httpd
```

Configuration

Apache was designed to be modular, so you can run as lean, or as bloated, a web server as you like. We discuss the basic configuration and some minor tweaks; all the directives are described at httpd.apache.org/docs/mod/directives.html, but during your Linux installation, the documents were placed in /var/www/html/manual/mod/directives.html. You can reach them via the link on the default web page

Modifying the Default Configuration:

Apache's configuration file is /etc/httpd/conf/httpd.conf. We'll start with the default configuration file from Apache version 1.3.24 are many comments in httpd.conf; read them! This is a great way to get a feel for how the Apache configuration file works. Most of it you'll leave unchanged, but the comments will familiarize you with the available capabilities.

Configuration files evolve over time, so don't worry too much if this stuff isn't exactly the same— it probably won't be. By the way, as of this writing, 1.3.24 was the latest Red Hat version, but 1.3.27 had already been released, and version 2.0 was in beta.

First, change the following:

ServerAdmin root@webserver.example.com

to:

ServerAdmin webmaster@example.com

or whoever you want to get all the comments (and complaints) about your web site. You may want joe_user@example.com to be your e-mail address or, more preferably, a webmaster e-mail alias. It should be someone who checks e-mail frequently, for some value of frequent proportional to how often people look at your web site.

Apache logs every hit to the web server (see the section on log files later in this chapter). Some of the information that can be written to the log includes the following:

- The client (Web surfer) IP address
- The date
- The URI requested (all the stuff after www.example.com/)
- The referer --- the web page the client was at when they clicked the link to take them to our web page
- The user agent (the browser the client is using)

There are various predefined CustomLog options, logging more or less information. The default logged in /var/log/httpd/ includes some of this information. There more information, so use the format that includes the most information. Change the following:

CustomLog /var/log/httpd/logs/access_log common

to:

CustomLog /var/log/httpd/logs/access_log combined

Other options include

CustomLog /var/log/httpd/logs/access_log agent

CustomLog /var/log/httpd/logs/access_log referer

But the combined format includes this information, as you might expect from the name. Using combinations of these options, you can customize your logging experience.

Securing Apache:

Most of the following configuration directives are optional; you can do as you wish for your setup, but you should be aware of the choices

Set User and Group

Make sure the user and group are set to:

User apache

Group apache

The user and group could be apache or bozo or whatever. The important thing is that Apache doesn't run as root, which, if Apache were cracked, could allow someone to crack your box from the root Apache account. Red Hat defaults to apache. You could instead create a new user with useradd, with a locked account to run Apache, if you wish. For now, we recommend that you stay with the default.

You might notice that if you do a ps aux | grep httpd, the first process listed is owned by root—this process binds the low port and spawns the apache-owned processes that actually handle the httpd requests.

Remove Online Manuals

If you did the default Red Hat install , the Apache manuals were installed in the html directory

/var/www/html/manual/, which can be accessed via file:///var/www/html/manual/ or http://localhost/manual/, or

www.your_web_site.com/manual/.

If you leave these on your machine, a cracker could gain information about your machine and installation (such as the server version) by

simply hitting this directory. It's a good idea to move the manuals someplace out of the web path:

mv /var/www/html/manual /usr/doc/apache-1.3.24/

Now you have the manual available to you (put the URL file:///usr/doc/apache-1.3.24/ in your browser), but it will not be served by

Apache to any would-be cracker.

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, we suggest you don't allow Apache to serve up these documents, so comment out this directive.

Don't Allow public_html Web Sites (Unless You Want To)

The mod_user module allows users to serve Web content without having access to the main web directory tree. For example, the user jrl, could create a directory called public_html in his home directory, which would be available at the URL http://servername/~jrl/.

You may want to consider whether to allow users (if you have any) to create these public_html sites. Nothing is inherently wrong with allowing public_html, but it should be something you decide to allow rather than just letting it happen by default. Quite a few directives are involved in the configuration of this feature, but if you locate the line:

UserDir public_html

and modify it as follows:

UserDir disabled

this feature is turned off.

.htaccess

You 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 your web page or other parts of your 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. Sequence is important; the last command takes precedent. The rule here is to deny everything except that specifically allowed, a good rule of thumb. More on .htaccess later.

Remove server-status and server-info

The following directives allow clients to find out information about your machine and server. There's no reason to give crackers any more information than necessary. Comment this out for now:

#<Location /server-status>

SetHandler server-status

Order deny,allow

Deny from all

Allow from .your_domain.com

#</Location>

and this:

#<Location /server-info>

SetHandler server-info

Order deny,allow

Deny from all

Allow from .your_domain.com

#</Location>

If you decide to allow this information to be given out (perhaps for debugging from a remote site), change .your_domain.com to the

specific sites you want to have access.

Disallow Symbolic Links

Allowing symbolic links from within your web server document tree to other directories can cause content control problems. We suggest

that you do not allow symbolic links unless you have to.

To disallow symbolic links, be sure that the Options directives do not include FollowSymLinks.

Symbolic links are good for making links to large files instead of having multiple copies, but bad because J. Random Luser could make a link to a sensitive file—for instance, /etc/passwd. Disk space is cheap. Make copies.

Do Not Allow Directory Indexes

If you add Indexes to the Options directive, clients can access directory listings if they type in a directory with no index.html—for example,

www.example.com/directory/. This is generally a bad idea because it lets people look at the directory structure, perhaps to see files that you didn't want served up—.htaccess files, old versions, backups. Better to let them see only the files that you decided to serve up via the web page. Be sure Indexes is not part of your Options directive.

Don't Be a Proxy Server Unless You Want to Be

If you don't want to be a proxy server (if you don't know what this means, you don't want to be a proxy server), make sure the following

sections are commented out:

#LoadModule proxy_module modules/libproxy.so

and:

#AddModule mod_proxy.c

and:

#<IfModule mod_proxy.c>

#ProxyRequests On

#

#<Directory proxy:*>

Order deny,allow

Deny from all

Allow from .your_domain.com

#</Directory>

```
#
# Enable/disable the handling of HTTP/1.1 "Via:" headers.
# ("Full" adds the server version; "Block" removes
# all outgoing Via: headers)
# Set to one of: Off | On | Full | Block
#
#ProxyVia On
#
# To enable the cache as well, edit and uncomment
# the following lines:
# (no caching without CacheRoot)
#
#CacheRoot "/var/cache/httpd"
#CacheSize 5
#CacheGcInterval 4
#CacheMaxExpire 24
#CacheLastModifiedFactor 0.1
#CacheDefaultExpire 1
#NoCache a_domain.com another_domain.edu joes.garage_sale.com
#</IfModule>
```

Disable CGI Programs

We will talk more about CGI later, but for now, disable any CGI scripts that were shipped with Apache, as follows:

chmod -x /var/www/cgi-bin/*

Better yet, remove them:

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

And don't download CGI scripts from the Web. Whether they are malicious or simply badly written, CGI scripts are an excellent way to

get your system cracked. if you do use a script from someplace such as the Comprehensive Perl Archive Network (CPAN) at www.cpan.org, you can vet it for security.

Reload the Configuration File

Now that we are finished securing Apache, let's reload the configuration file as follows:

/etc/init.d/httpd graceful

Create the Web Site

The best way to learn is by doing, so you'll build an example web site on your machine as we go along. After you're done, you'll have installed and configured the sections you're interested in, and you can use this as a model for your own system, then delete it when you're confident you understand what's going on. This way, you can practice cgi-bin, mod_perl, PHP, dynamic web content, etc., in the privacy of your own home before you unleash your server on the World Wide Web.

There are two approaches you can take to experiment with the examples in this book: Either download the entire web site, including all of the examples, or you can create the examples yourself using your favorite editor.

Microsoft Word and its ilk are not suitable for much of the editing you will do in this book. These graphical editors use a nonstandard character set and insert characters that break configuration files, which need to be plain ASCII. They are also the source of most of the screwed-up text you see on web pages and newsgroups. Use something else—emacs, vi.

Downloading the Examples

If you follow the lazy approach to programming, you understand the importance of copy-paste. If you choose to download all the examples, go to www.opensourcewebbook.com/sourcecode/. Follow the instructions on how to obtain a username/password. You will need the password to download the source and view all the examples. Once you are ready, click the link for source.tar.gz and enter the username/password. Save the tarball in a convenient place, say in /tmp.

Now, as root, execute the following commands:

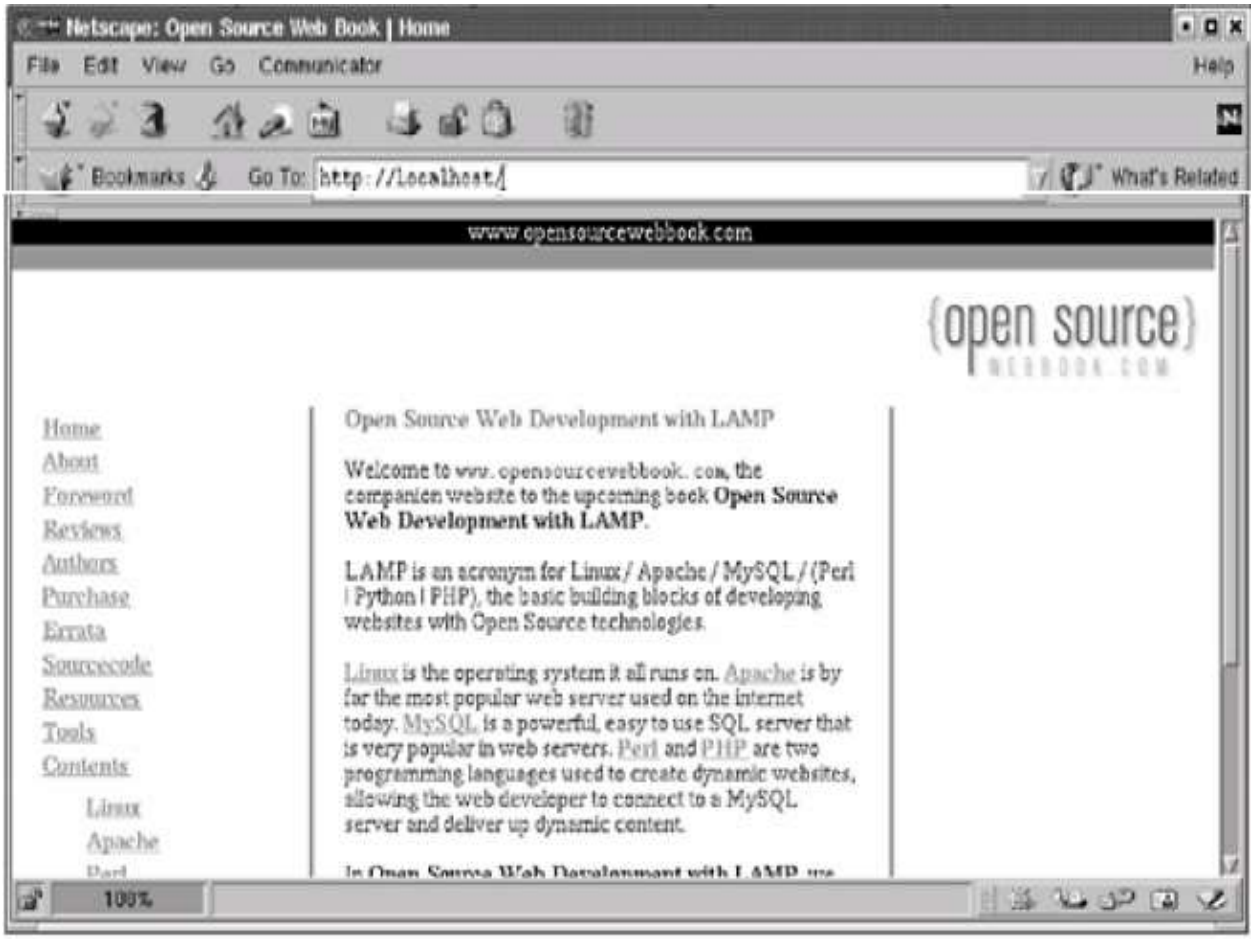
```
# mv /var/www /var/www.old

# mkdir /var/www

# chown jrl /var/www

# cd /var/www

# tar xzvf /tmp/source.tar.gz

# find . -exec chown jrl { }\;
```

First, the web site that came with Red Hat is moved to /var/www.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.

Now, you are ready to rock with the new web site. To see whether everything worked, point your browser to http://localhost/. It should resemble www.opensourcewebbook.com/, which is shown in Figure A.

Recall that 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 file Therefore, the file requested is /var/www/html/index.html, and if you look at that file's contents, you will see the HTML that builds Figure A.

Figure A. Screen capture from www.opensourcewebbook.com



Look in the directory /var/www. You will see several files and many directories—that is the source for all the examples in this book, and you will study them all in short order.

Creating Them Yourself :

If you choose not to download all the examples as we have just shown (are you sure about that?) and would rather create each example as it is discussed, you can either type them all in or download them from the web site one at a time. When it is time to experiment, you can download the example from there, save it into the appropriate directory, and experiment with it at that time. If you are going this route, let's start with a simple example. First, create an HTML file in /var/www/html/index.html.

Place the following text in index.html:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"

 "http://www.w3.org/TR/REC-html40/loose.dtd">

<html>

<head>

</head>

<body>

hello, world

</body>

</html>
```

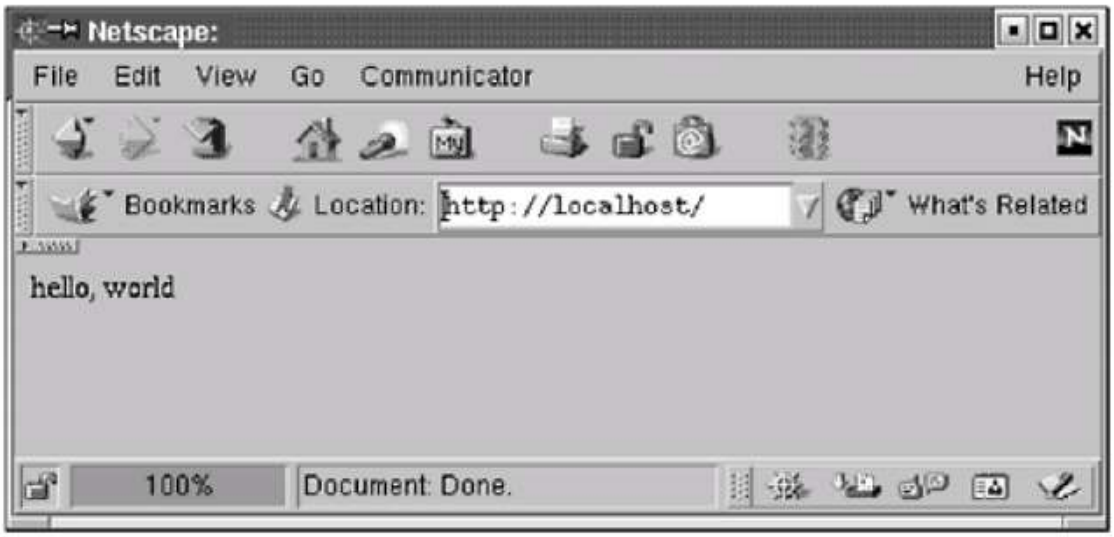
Set the permissions so that you can write to the file and the rest of the world can read it; if the file and directory are not world readable, others won't have permission to view the page in their browser. It's a good idea to view your web pages from a different machine (and using different browsers) to catch this sort of error.

```
$ chmod a+r /var/www/html/index.html
```

The first two lines of our example, starting with <!DOCTYPE and continuing to .dtd">, contain the doctype, which is required for the page to pass strict HTML muster. This page does. However, having a doctype in all our examples seems a bit cluttered (all browsers will render the page without it), so we drop it in the rest of our examples. An excellent place to vet HTML code can be found at www.validator.w3.org.

Reload your browser (Shift-Reload in Netscape and Mozilla to clear the cache and reload from scratch) with http://localhost/. You should see the following figure

Figure B Hello, world



Now, we will do something that is more complicated. For this, go to www.opensourcewebbook.com/. As we saw, this page is shown in Figure A. View the document source for this file by selecting View Page Source from one of the menus in your. Copy all the source that you see to /var/www/html/index.html. Reloading http://localhost/ should display a page similar to Figure A. However, you will notice that the logo is broken—the image file for that logo does not exist on your machine. To copy this logo, go to www.opensourcewebbook.com, hover the mouse over the logo in the top right corner of the page, and, using the right mouse button, save the image to /var/www/icons/logo.gif. Reloading http://localhost/ now shows the image displayed on the page. Creating web pages is as simple as creating directories and .html files. Creating good web pages is another matter—for that, you need a good eye for design in addition to simple geek skills such as writing markup language. For instance, www.opensourcewebbook.com was designed by a graphic artist, not a geek. Created by the fine folks at BDGI (www.bdgi.com).

Apache Log Files:

Apache keeps detailed logs of accesses to your 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 as swatch or logwatch, to keep an eye on these files for security violations and problems. It's also a good idea to rotate them regularly via logrotate, because they'll get big otherwise.

Of course, the most important use of the logs is to look at them when there is a problem to figure out what went wrong. An access_log entry might look like this:

192.168.1.12 - - [21/May/2001:14:10:05 -0600]

"GET / HTTP/1.0" 200 43

"http://www.onsight.com/" "Mozilla/4.77 [en]

(X11; U; Linux 2.4.2-2 i686; Nav)"

Earlier in the chapter, we discussed how to select the information that is displayed in the Apache log file This log selection displays the previous information. Most of these entries should be clear: the IP address of the requestor, the date and time of the request, the browser used (though clever programs such as Opera let you tell the server anything you want, and you can define this to be anything you want if you change the headers and recompile Mozilla), the language and some system details of the requestor. "GET / HTTP/1.0" 200 43 means an HTTP request, no errors (200), and 43 bytes were sent.

Access Control with .htaccess:

Earlier we discussed how to enable .htaccess files in your configuration. Now we'll show you how they're used. This is useful for restricting access to certain portions of your web site, either by allowing access only from specific IP addresses or domains or by password control.

In httpd.conf, look for the line Directory /var/www/html (or whatever the default directory is). There you will see:

AllowOverride None

Change this to:

AllowOverride AuthConfig

This change tells the server to change its behavior from allowing anyone to connect to allowing only those clients whose attributes match those in an authorization file to connect to the files in that directory. Make sure the .htaccess filename definition is uncommented. You could change the name of the .htaccess file via this directive:

AccessFileName .htaccess

We mentioned this directive before, but it deserves mentioning again. Since we use the file .htaccess to control access, make sure the

htaccess directive is uncommented (see Section 3.4.5)—it denies serving any file whose name begins with.ht, meaning that clients can't look at your .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 a+rx private

$ cd private

And create a simple index.html file (remember to make it readable with chmod a+r index.html):

<html>

<head>

<title>

My Private Directory

</title>

</head>

<body>

Congratulations! You now have access to my private directory!

</body>

</html>

Now, create a password file. It's convenient and tempting to put it in the same directory and call it something like .htpasswd. Don't. Place it

outside the document tree. If someone were to get access to this directory because of a server misconfiguration (hey, it happens—the

configuration file is big and mistakes do happen), you wouldn't want them to have access to your password files (even though the

passwords are encrypted), especially because many people tend to use the same passwords for many different purposes. This is a very bad idea, so not only do you as a sysadmin and webmaster want to advocate good habits, you want to defend against bad ones.

$ mkdir /var/www/misc

$ chmod a+rx /var/www/misc

$ cd /var/www/misc

Create a password file:

$ htpasswd -bc private.passwords neo anderson

Adding password for user neo

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

As always, man htpasswd is recommended before you do this stuff—if we told you to, you wouldn't docd / ; rm -rf

*, would you? Even if everybody else was doing it?

$ htpasswd -b private.passwords morpheus sleeps

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

$ cd /var/www/html/private

$ vi .htaccess

The file .htaccess has this in it:

AuthName "My Private Area"

AuthType Basic

AuthUserFile /var/www/misc/private.passwords

AuthGroupFile /dev/null

require valid-user

Figure A. Login

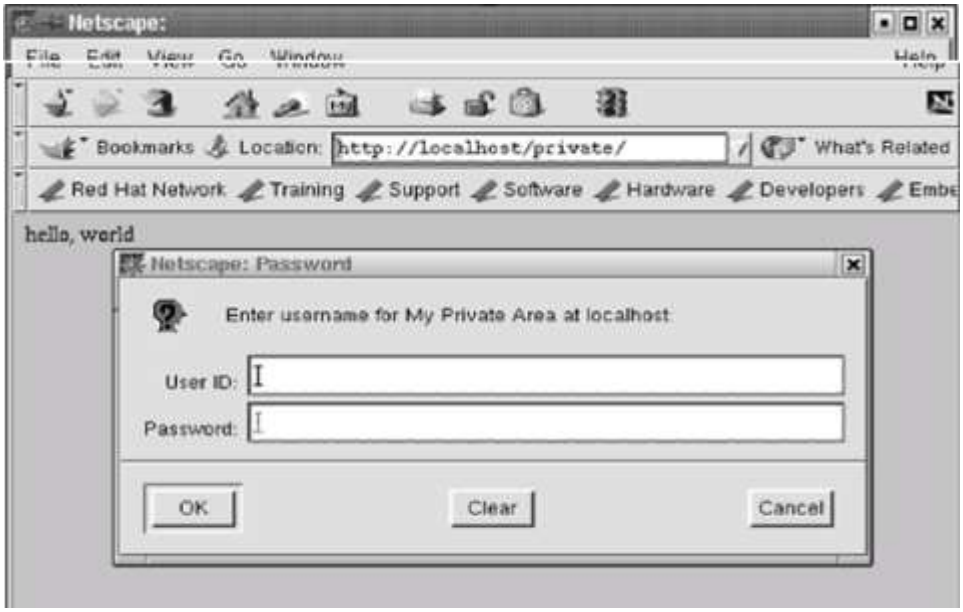


Figure B. Username, password

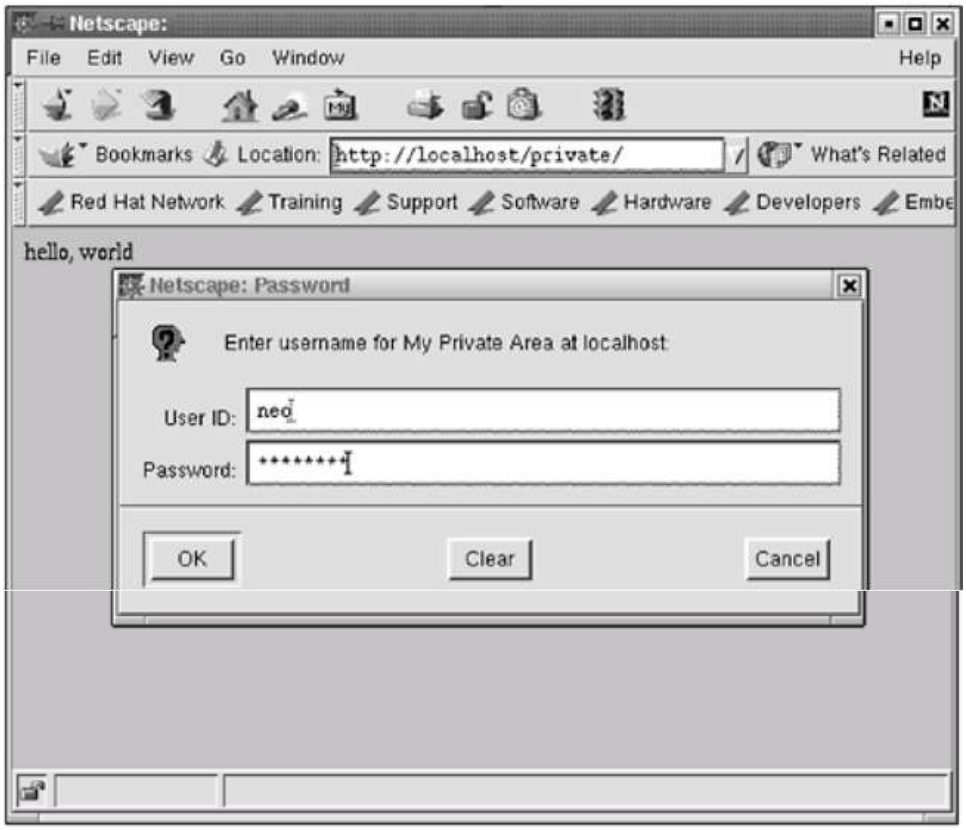

Netscape:
File Edit View Go Window Help
Bookmarks Location: http://localhost/private/ What's Related
Red Hat Network Training Support Software Hardware Developers Embe
hello, world
Netscape: Password
Enter username for My Private Area at localhost:
User ID: neo
Password: ********
OK
Clear
Cancel

Figure C. Access granted

There's no encryption when you use passwords like this—the passwords go over the network in the clear, which as you might imagine, is not an optimal configuration. Be aware that doing so makes you vulnerable to password sniffing. If you do use this, don't use the same password as your Linux login, and be aware of your vulnerabilities. You can monitor access to these directories with your log monitoring program if you desire. You can use HTTPS to have secure communications for this kind of thing, but that is (for now) beyond our scope. You can also do simple IP verification by putting the following in your .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 the preceding 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, in addition to any external IP address. If you point your web browser to localhost or 127.0.0.1, it will serve up the default page of the local Apache host. Your localhost is not the same as the

fellow in the next cubicle, though. You can combine passwords and IP verification for additional security.

KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
(Established Under Section 3 of UGC Act, 1956)
Coimbatore-21

DEPARTMENT OF CS, CA & IT

UNIT- III (Objective Type/ Multiple Choice Questions Each Question Carries One Mark)

OPEN SOURCE TECHNOLOGIES (17CAU504B)

S.No	Question	Option1	Option2	Option3	Option4	Answer
1	_____ has a BSD-type open source license, making it alternative for both commercial and non-commercial applications	TCL	Apache	Perl	PHP	Apache
2	Apache is a _____	Web browser	Web server	Application server	Database server	Web server
3	_____ receives the request from the client	Host	Sever	Node	browser	Server
4	_____ tells the client how to interpret the information that is to follow.	Header	Footer	Content	Command	Header
5	The header is separated from the content by a _____	single white space	tab space	double white space	blank line	blank line
6	To check whether Apache is running, when starting the machine by load _____ Url in the browser	http://localhost	http://www.localhost/	http:\\localhost	http:\\www.localhost\	http://localhost
7	URL stands for _____	Uniform Resource Loader	Unified Resource Locator	Uniform Resource Locator	Unified Resource Loader	Uniform Resource Locator
8	Apache can process _____ request at a time.	one	two	Less than one	more than one	more than one
9	The Apache log file does not contain_____	Client IP address	date	the author name	the referer	the author name
10	The _____ module allows users to serve web content without having access to the main web directory.	user_mod	mod_user	mod#user	user#mod	mod_user
11	The Apache logs are located at_____	/var/log/httpd/access_log	/var/httpd/access_log	/var/access_log	/var/log/access_log	/var/log/httpd/access_log

12	_____ keeps detailed logs of accesses to the web site, errors and more.	CGI	HTML	Apache	HTTP	Apache
13	CGI stands for_____	Common Graphics Information	Common Graphics Interface	Common Gateway Interconnect	Common Gateway Interface	Common Gateway Interface
14	The locations of Apache logs are configurable in _____	httpd.config	httpd.confi	httpd.conf	httpd.con	httpd.conf
15	_____ is a log monitor program monitors the log files for security violations problems	log	watch	logwatch	watching	logwatch
16	Create a new user with _____,with a locked account to run Apache	adduser	useradd	createuser	usercreate	useradd
17	_____ directive allows access to the local documentation in /usr/share/doc	<Location /doc>	</doc Location>	<LocalLocation /doc>	</doc LocalLocation>	<Location /doc>
18	The user and group called Apache or _____	team	bozo	groups	usergroup	bozo
19	_____ is the web page the client was at when they clicked the link them to our web page.	Client	Server	Referer	User Agent	Referer
20	_____ If Apache were cracked, could allow someone to crack your box from the root Apache account is called_____	rooting	rooting your box	cracking	cracking your box	rooting your box
21	The _____ is an American non-profit corporation to support Apache software	ASF	FSF	OSF	BSF	ASF
22	ASF stands for _____	American Software Foundation	Apache Software Foundation	American System Foundation	Apache System Foundation	Apache Software Foundation
23	_____ is the extension of Apache log file	.con	.config	.conf	.confi	.conf
24	Does Apache act as proxy server?	can't be determined	yes, by default	no	yes, using mod_proxy module	yes, using mod_proxy module
25	What option in the Indexoptions Directive is only available after Apache1.3.10?	fancyindexing	iconheight	namewidth	foldersfirst	foldersfirst

26	The Apache HTTP server was launched in the year_____	1990	1995	1992	1994	1995
27	d in httpd stands for	directive	daemon	document	description	daemon
28	The _____ command is used to kill all the processes.	killall	kill	pkill	xkill	killall
29	Apache must be compiled with the _____ module	mod_usertrack	mod_cookies	mod_userlog	mod_userlog	mod_usertrack
30	HTTP stands for _____	Hyper Text Transform Protocol	Hyper Text Transform Prototype	Hyper Text Transfer Protocol	Hyper Text Transfer Prototype	Hyper Text Transfer Protocol
31	_____ is stored in the location /var/log/httpd	System file	Configuration file	Log file	Class file	Log file
32	What action must be done to enable cookies within Apache?	Only installing the modtrack	Installing mod_cookies	Installing mod_usertrack and setting the directive cookietracking on	Only setting the directive cookietracking on	Installing mod_usertrack and setting the directive cookietracking on
33	What is the function of the mod_imap module?	Handles IMAP e-mail protocols	Performs image map	Allows for shared inter module memory protocols	There is no mod_imap	Performs image map
34	_____ is the default location for the http.conf file on windows	c:\Program Files\ASF\Apache2.2\conf	c:\Program Files\ASF\Apache2.2\confi	c:\openssl\conf	c:\Program Files\ASF\Apache2.2\configuration	c:\Program Files\ASF\Apache2.2\conf
35	Apache server works on which port?	http - port 80 only	http - port 80 and https - port 443	http - port 443 only	no port	http - port 80 and https - port 443
36	What is the command to stop Apache?	/etc/init.d/httpd stop	/etc/init.d/httpd end	/etc/init.d/httpd finish	/etc/init.d/httpd close	/etc/init.d/httpd stop
37	What is the command to start Apache?	/etc/init.d/httpd begin	/etc/init.d/httpd start	/etc/init.d/httpd initiate	/etc/init.d/httpd create	/etc/init.d/httpd start
38	What is the command to restart Apache?	/etc/init.d/httpd reassign	/etc/init.d/httpd resume	/etc/init.d/httpd regenerate	/etc/init.d/httpd restart	/etc/init.d/httpd restart
39	_____ is a configuration file for use on web servers running the Apache web server software	.htaccess	.htpasswd	.httpaccess	.httpdaccess	.htaccess
40	_____ command is used to check the status of the server	/etc/init.d/httpd showstatus	/etc/init.d/httpd viewstatus	/etc/init.d/httpd status	/etc/init.d/httpd showstatus	/etc/init.d/httpd status
41	_____ directive is used to find out the information about the machine and server	/Server-Status	/Server-Info	/Server-Status or /Server-Info	No such Directive	/Server-Status or /Server-Info

42	what is .htaccess?	it creates a new user	asks to specify a password for user	it creates a new user and asks to specify a password for user	change the password for that user	it creates a new user and asks to specify a password for user
43	If you specify both deny from all and allow from all what will be the default action of Apache is_____	can't be determined	only allow will be performed	Deny always take precedence over allow	Allow always take precedence over deny	Deny always take precedence over allow
44	which of the following modules must be compiled into the Apache web server?	http_core.c	apache_so.c	charset_so.c	tcpip_core.c	http_core.c
45	When Apache write its log files to syslog, it will run with permissions of which user?	Always as root	Always as nobody	Always as webuser	Always as owner	Always as nobody

46	Which of the following allow to handle multiple requests over a single TCP connection?	Expirations	BrowserMatch	KeepAlive	PersistentConnect	KeepAlive
47	Which of the following is used to see the Apache's default server?	http://localserver	http://localhost	http://localin	http://index	http://localhost
48	Which of the following are true regarding contains in http.com?	containers consist of a paired set of delimiters	containers may only be applied globally	containers may only be applied locally	If two containers contradict, the first is applied	containers consist of a paired set of delimiters
49	Which of the following can be used to send a cookie to the http client?	XML Script	Static HTML	CGI Script	GIF Script	CGI Script
50	CLF stands for_____	Common Log Format	Computer Log Format	Common Logical Time to serve request	Computer Logical Format	Common Log Format
51	Which of the following is not a part of the common log format?	Requesting host	Data of request		http status code	Time to serve request
52	Which httpd.conf directives are used to change the port and the IP address?	Listen	Bind	Active	BindAddress	Listen
53	Which of the following must Apache1.1 and higher have in order to run?	http.conf and srm.conf	http.conf	http.conf and access.conf	http.conf, access.conf and srm.conf	http.conf
54	The commands in Apache may starts with _____	#	%	!	*	#
55	The web document root directory can be changed by which line in httpd.conf?	Root	RootDir	DocumentRoot	BaseDir	DocumentRoot
56	Which of the following line used t include the contents of the file "message.txt" within a page?	<!--#exec "cat message.txt" - ->	<!--#exec cmd "cat message.txt" - ->	<!--#execute cmd "message.tx	<!--#exec "message.txt" - ->	<!--#exec cmd "cat message.txt" - ->
57	To create dynamic HTML content, which of the following must be included with each CGI script?	each $cgi	#bin/bash	#include cgi	each content type	each content type
58	NCSA stands for _____	National Center for Supercomputing Applications	National Control for Security Applications	National Code for Supercomputing Application	National Code for Security Applications	National Center for Supercomputing Applications
59	Apache was originated at the University of _____	California	Illina's Urbana-Champaign	Cambridge	Bitspilani	Illina's Urbana-Champaign

60	Which of the following http.conf entries will tell the server which access directive can be overridden by the .htaccess file?	Order Allow, Deny	Allowoverride	Order Deny	Allow	Allowoverride

Unit-IV

MySQL

Introduction:

Many of the applications that a Web developer wants to use can be made easier by the use of a standardized database to store, organize, and access information. MySQL is an Open Source (GPL) 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 (though all have their nonstandard extensions).

MySQL can easily be integrated into Perl programs by using the Perl DBI (DataBase Independent interface) module. DBI is an Application Program Interface (API) that allows Perl to connect to and query a number of SQL databases.

Tutorial - MySQL:

A few SQL commands go a long way to facilitate learning MySQL/Perl/DBI. To illustrate these, we create a simple database containing information about some people. Eventually, we'll show how to enter this information from a form on the Web, but for now we interface with SQL directly.

First, try to make a connection to our MySQL server as the root MySQL user:

$ mysql -u root

The MySQL root user is different from the Linux root user. The MySQL root user is used to administer the MySQL server only. If you see the following output:

ERROR 2002: Can't connect to local MySQL server through socket

´/var/lib/mysql/mysql.sock´(2)

it likely means the MySQL server is not running. If your system is set up securely, it shouldn't be running, because you had no reason, before now, for it to be running. Use chkconfig as root to make sure it starts the next time the machine boots, and then start it by hand as follows:

chkconfig mysqld on

/etc/init.d/mysqld start

Now you should be able to connect (not logged in as the Linux root user):

$ mysql -u root

If not, see the MySQL log file at /var/log/mysqld.log. If so, you'll see a welcome message and the MySQL prompt:Welcome to the MySQL monitor. Commands end with ; or \g. Your MySQL connection id is 3 to server version: 3.23.36

Type ´help;´ or ´\h´ for help. Type ´\c´ to clear the buffer

mysql>

As suggested, enter help; at the prompt. A list of MySQL commands (not to be confused with SQL commands) will be displayed. These allow you to work with the MySQL server. For grins, enter status; to see the status of the server.

To illustrate these commands, we will create a database called people that contains information about people and their ages.

The SHOW DATABASES and CREATE DATABASE Commands:

First, we need to create the 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> SHOW DATABASES;
+----------+
| Database |
+----------+
| mysql    |
| test     |
+----------+
2 rows in set (0.00 sec)
```

mysql> CREATE DATABASE people;

Query OK, 1 row affected (0.00 sec)

```
mysql> SHOW DATABASES;
+----------+
| Database |
+----------+
| mysql    |
| people   |
| test     |
+----------+
3 rows in set (0.00 sec)
```

SQL commands and subcommands (in the previous example, CREATE is a command; DATABASE is its subcommand) are case-insensitive. The name of the database (and table and field) are case sensitive. It's a matter of style whether one uses uppercase or lowercase, but traditionally the SQL commands are distinguished by uppercase.

One way to think of a 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.

The USE Command:

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

```
mysql> USE people;
```

The CREATE TABLE and SHOW TABLES Commands:

Each table within the database must be defined and created. This is done with the CREATE TABLE command. Create a table named age_information 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. In this case, the first name and the last name are character strings of up to 20 characters each, and the age is an integer:

```
mysql> CREATE TABLE age_information (
   -> lastname   CHAR(20),
   -> firstname  CHAR(20),
   -> age        INT
```

```
   -> );
Query OK, 0 rows affected (0.00 sec)
```

It appears that the table was created properly (it says OK after all), but this can be checked by executing the SHOW TABLES command. If an error is made, the table can be removed with DROP TABLE.

When a database in MySQL is created, a directory is created with the same name as the database (people, in this example):

```
# ls -l /var/lib/mysql
total 3
drwx------    2 mysql    mysql       1024 Dec 12 15:28 mysql
srwxrwxrwx    1 mysql    mysql          0 Dec 13 07:19 mysql.sock
drwx------    2 mysql    mysql       1024 Dec 13 07:24 people
drwx------    2 mysql    mysql       1024 Dec 12 15:28 test
```

Within that directory, each table is implemented with three files:

```
# ls -l /var/lib/mysql/people
total 10

-rw-rw----    1 mysql    mysql       8618 Dec 13 07:24 age_information.frm
-rw-rw----    1 mysql    mysql          0 Dec 13 07:24 age_information.MYD
-rw-rw----    1 mysql    mysql       1024 Dec 13 07:24 age_information.MYI

mysql> SHOW TABLES;
+-----------------+
| Tables_in_people |
+-----------------+
| age_information  |
+-----------------+
1 row in set (0.00 sec)
```

This example shows two MySQL datatypes: character strings and integers. Other MySQL data types include several types of integers

- TINYINT - 128 to 127 (signed) or 0 to 255 (unsigned)

- SMALLINT - 32768 to 32767 (signed) or 0 to 65535 (unsigned)
- MEDIUMINT - 8388608 to 8388607 (signed) or 0 to 16777215 (unsigned)
- INTEGER (same as INT) - 2147483648 to 2147483647 (signed) or 0 to 4294967295 (unsigned)
- BIGINT -9223372036854775808 to 9223372036854775807 (signed) or 0 to 18446744073709551615 (unsigned)

and floating points:

- FLOAT
- DOUBLE
- REAL (same as DOUBLE)
- DECIMAL
- NUMERIC (same as DECIMAL)

There are several data types to represent a date:

- DATE
- YYYY-MM-DD
- DATETIME
- TIMESTAMP
- TIME
- YEAR
- YYYY-MM-DD HH:MM:SS
- YYYYMMDDHHMMSS or YYMMDDHHMMSS or YYYYMMDD or YYMMDD
- HH:MM:SS
- YYYY or YY

The table age_information used the CHAR character data type. The following are the other character data types. Several have BLOB in their name— a BLOB is a Binary Large OBject that can hold a variable amount of data. The types with TEXT in their name are just like their

corresponding BLOBs except when matching is involved: The BLOBs are case-sensitive, and the TEXTs are case-insensitive.

- VARCHAR - variable-length string up to 255 characters
- TINYBLOB - maximum length 255 characters
- TINYTEXT
- BLOB - maximum length 65535 characters
- TEXT
- MEDIUMBLOB - maximum length 16777215 characters
- MEDIUMTEXT
- LONGBLOB - maximum length 4294967295 characters
- LONGTEXT

The DESCRIBE command:

The DESCRIBE command gives information about the fields in a table. The fields created earlier—lastname, firstname, and age—appear to have been created correctly.

```
mysql> DESCRIBE age_information;
+-----------+----------+------+-----+---------+-------+
| Field     | Type     | Null | Key | Default | Extra |
+-----------+----------+------+-----+---------+-------+
| lastname  | char(20) | YES  |     | NULL    |       |
| firstname | char(20) | YES  |     | NULL    |       |
| age       | int(11)  | YES  |     | NULL    |       |
+-----------+----------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

The command SHOW COLUMNS FROM age_information; gives the same information as DESCRIBE age_information; but DESCRIBE involves less typing. (If you're really trying to save keystrokes, you could abbreviate DESCRIBE as DESC.)

The INSERT Command:

For the table to be useful, we need to add information to it. We do so with the INSERT command:

```
mysql> INSERT INTO age_information
    ->      (lastname, firstname, age)
    ->      VALUES (´Wall´ , ´Larry´ , 46);
Query OK, 1 row affected (0.00 sec)
```

The syntax of the command is INSERT INTO, followed by the table in which to insert, a list within parentheses of the fields into which information is to be inserted, and the qualifier VALUES followed by the list of values in parentheses in the same order as the respective fields.

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. The simplest use of SELECT is shown in this example:

```
mysql> SELECT * FROM age_information;
+----------+-----------+------+
| lastname | firstname | age  |
+----------+-----------+------+
| Wall     | Larry     |   46 |
+----------+-----------+------+
1 row in set (0.00 sec)
```

The * means "show values for all fields in the table"; FROM specifies the table from which to extract the information. The previous output shows that the record for Larry Wall was added successfully. To experiment with the SELECT command, we need to add a few more records, just to make things interesting:

```
mysql> INSERT INTO age_information
    ->      (lastname, firstname, age)
    ->      VALUES (´Torvalds´, ´Linus´, 31);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO age_information
    ->      (lastname, firstname, age)
```

```
    ->      VALUES (´Raymond´, ´Eric´, 40);
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM age_information;
+----------+-----------+------+
| lastname | firstname | age  |
+----------+-----------+------+
| Wall     | Larry     |   46 |
| Torvalds | Linus     |   31 |
| Raymond  | Eric      |   40 |
+----------+-----------+------+
3 rows in set (0.00 sec)
```

There are many ways to use the SELECT command—it's very flexible. First, sort the table based on lastname:

```
mysql> SELECT * FROM age_information
    ->      ORDER BY lastname;
+----------+-----------+------+
| lastname | firstname | age  |
+----------+-----------+------+
| Raymond  | Eric      |   40 |
| Torvalds | Linus     |   31 |
| Wall     | Larry     |   46 |
+----------+-----------+------+
3 rows in set (0.00 sec)
```

Now show only the lastname field, sorted by lastname:

```
mysql> SELECT lastname FROM age_information
    ->      ORDER BY lastname;
+----------+
| lastname |
+----------+
| Raymond  |
| Torvalds |
| Wall     |
```

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

Show the ages in descending order:

```
mysql> SELECT age FROM age_information ORDER BY age DESC;
+------+
| age  |
+------+
|   46 |
|   40 |
|   31 |
+------+
3 rows in set(0.00 sec)
```

Show all the last names for those who are older than 35:

```
mysql> SELECT lastname FROM age_information WHERE age > 35;
+----------+
| lastname |
+----------+
| Wall     |
| Raymond  |
+----------+
2 rows in set (0.00 sec)
```

Do the same, but sort by lastname:

```
mysql> SELECT lastname FROM age_information
   ->     WHERE age > 35 ORDER BY lastname;
+----------+
| lastname |
+----------+
| Raymond  |
| Wall     |
+----------+
```

2 rows in set (0.00 sec)

The UPDATE Command:

Since the database is about people, information in it can change (people are unpredictable like that). For instance, although a person's birthday is static, their age changes. To change the value in an existing record, we can UPDATE the table. Let's say the fictional Larry Wall has turned 47:

```
mysql> SELECT * FROM age_information;
+----------+-----------+------+
| lastname | firstname | age  |
+----------+-----------+------+
| Wall     | Larry     |   46 |
| Torvalds | Linus     |   31 |
| Raymond  | Eric      |   40 |
+----------+-----------+------+
3 rows in set (0.00 sec)

mysql> UPDATE age_information SET age = 47
    ->       WHERE lastname = ´Wall´;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM age_information;
+----------+-----------+------+
| lastname | firstname | age  |
+----------+-----------+------+
| Wall     | Larry     |   47 |
| Torvalds | Linus     |   31 |
| Raymond  | Eric      |   40 |
+----------+-----------+------+
3 rows in set (0.00 sec)
```

Be sure to use that WHERE clause; otherwise, if we had only entered UPDATE age_information SET age = 47, all the records in the database would have been given the age of 47!

Although this might be good news for some people in these records (how often have the old-timers said "Oh, to be 47 years old again"—OK, probably not), it might be shocking news to others.

This method works, but it requires the database to know that Larry is 46, turning 47. Instead of keeping track of this, for Larry's next birthday we simply increment his age:

```
mysql> SELECT * FROM age_information;
+----------+-----------+------+
| lastname | firstname | age  |
+----------+-----------+------+
| Wall     | Larry     |   47 |
| Torvalds | Linus     |   31 |
| Raymond  | Eric      |   40 |
+----------+-----------+------+
3 rows in set(0.00 sec)

mysql> UPDATE age_information SET age = age + 1
    ->       WHERE lastname = ´Wall´;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM age_information;
+----------+-----------+------+
| lastname | firstname | age  |
+----------+-----------+------+
| Wall     | Larry     |   48 |
| Torvalds | Linus     |   31 |
| Raymond  | Eric      |   40 |
+----------+-----------+------+
3 rows in set(0.00 sec)
```

The DELETE Command:

Sometimes we need to delete a record from the table (don't assume the worst—perhaps the person just asked to be removed from a mailing list, which was opt-in in the first place, of course).

This is done with the DELETE command

```
mysql> DELETE FROM age_information WHERE lastname = ´Raymond´;
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM age_information;
+----------+-----------+------+
| lastname | firstname | age  |
+----------+-----------+------+
| Wall     | Larry     |   48 |
| Torvalds | Linus     |   31 |
+----------+-----------+------+
2 rows in set (0.00 sec)
```

Eric is in good company here, so put him back:

```
mysql> INSERT INTO age_information
    ->      (lastname, firstname, age)
    ->      VALUES (´Raymond´, ´Eric´, 40);
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM age_information;
+----------+-----------+------+
| lastname | firstname | age  |
+----------+-----------+------+
| Wall     | Larry     |   48 |
| Torvalds | Linus     |   31 |
| Raymond  | Eric      |   40 |
+----------+-----------+------+
3 rows in set (0.00 sec)
```

Some Administrative Details:

All these examples have been executed as the root MySQL user, which, as you might imagine, is not optimal from a security standpoint. A better practice is to create a MySQL user who can create and update tables as needed.

First, as a security measure, change the MySQL root password when logging in to the server:

mysqladmin password IAmGod

Now when mysql executes, a password must be provided using the -p switch. Here is what would happen if we forgot the -p:

$ mysql -u root

ERROR 1045: Access denied for user: ´root@localhost´ (Using password: NO)

Try again using -p. When prompted for the password, enter the one given previously:

Recall that the MySQL user is not the same as a Linux user. The mysqladmin command changes the password for the MySQL user only, not the Linux user. For security reasons, we suggest that the MySQL password never be the same as the password used to log in to the Linux machine. Also, the password IAmGod, which is clever, is a bad password for many reasons, including the fact that it is used as an example in this book. For a discussion on what makes a password good or bad, we suggest you read Hacking Linux Exposed [Hatch+ 02].

$ mysql -u root -p
Enter password:

Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 15 to server version: 3.23.36
Type ´help;´ or ´ \h´ for help. Type ´\c´ to clear the buffer mysql>
Doing all the SQL queries in the people database as the MySQL root user is a Bad Idea (see HLE if you want proof of this). So let's create a new user. This involves modifying the database named mysql, which contains all the administrative information for the MySQL server, so first we use the mysql database and then grant privileges for a new user:

mysql> USE mysql;

Reading table information for completion of table and column names.You can turn off this feature to get a quicker startup with -A

Database changed

```
mysql> GRANT SELECT,INSERT,UPDATE,DELETE
    ->    ON people.*
    ->    TO apache@localhost
    ->    IDENTIFIED BY ´LampIsCool´;
Query OK, 0 rows affected (0.00 sec)
```

The user apache (the same user that runs the webserver) is being granted the ability to do most everything within the database, including being able to delete entries in tables within the people database. However, apache cannot delete the people database, only entries within the tables in the database. The user apache can access the people database from localhost only (instead of being able to log in over the network from another machine).

The IDENTIFIED BY clause in the SQL command sets the apache user's password to LampIsCool. Setting the password is necessaryonly the first time permissions are granted for this user—later, when the apache user is given permissions in other databases, the password doesn't need to be reset.To verify that these changes were made, log in as apache:

```
$ mysql -u apache -p
Enter password:

Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 27 to server version: 3.23.36
Type ´help;´ or ´\h´ for help. Type ´\c´ to clear the buffer

mysql> USE people

Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SHOW TABLES;
```

```
+-----------------+
| Tables_in_people |
+-----------------+
| age_information  |
+-----------------+
1 row in set (0.00 sec)

mysql> SELECT * FROM age_information;
+----------+-----------+------+
| lastname | firstname | age  |
+----------+-----------+------+
| Wall     | Larry     |   48 |
| Torvalds | Linus     |   31 |
| Raymond  | Eric      |   40 |
+----------+-----------+------+
3 rows in set (0.00 sec)
```

Summary:

As discussed, these commands are enough to do basic things with MySQL:

- SHOW DATABASES
- CREATE DATABASE
- USE
- CREATE TABLE
- SHOW TABLES
- DESCRIBE
- INSERT
- SELECT
- UPDATE
- DELETE
- GRANT

Database Independent Interface:

Running MySQL commands from the shell is well and good the first 12 times it has to be done. After that, the typical lazy programmer starts thinking of ways to automate the process. Here, the answer is Perl and the DataBase Independent interface (DBI). DBI enables one to write programs to automate database maintenance and to write other scripts to interface with MySQL.
DBI is a Perl module that provides methods to manipulate SQL databases. With DBI, one can connect to a database within a Perl script and issue all kinds of queries, including SELECT, INSERT, and DELETE. For now, we create Perl scripts that can be run from the shell.
Later, we'll use CGI, mod_perl, Embperl, Mason, and PHP to hook database independent interfaces into web programs.

First, a quick example. We put all these DBI examples in a directory that is under /var/www/ so that the examples are downloadable from www.opensourcewebbook.com/. In the real world, we do not suggest you create a directory under /var/www/ to create arbitrary Perl programs, but for our purposes, it just makes life easier when downloading all the examples. Create the directory and go there:

```
$ mkdir /var/www/bin
$ cd /var/www/bin
```

The first example demonstrates how to connect to a database. This code is stored in the file /var/www/bin/connect.pl and online at http://localhost/mysql/connect.pl or www.opensourcewebbook.com/mysql/connect.pl. The content of connect.pl is:

```
#!/usr/bin/perl -w
# connect.pl
# use the DBI module
use DBI;
# use strict, it is a Good Idea
use strict;
# connect to the database, assigning the result to $dbh
my $dbh = DBI->connect(´DBI:mysql:people´, ´apache´, ´LampIsCool´);
# die if we failed to connect
die "Can't connect: " . DBI->errstr() unless $dbh;
# all is well!
print "Success: connected!\n";
# disconnect from the MySQL server
$dbh->disconnect();
```

First, the use DBI method tells Perl to use the DBI module. This allows us to use all the methods in this class. Calling the connect() method causes the Perl script to connect to the MySQL database using the Perl DBI class. The first argument to this method is the database to which you want to connect. In this example, the string DBI:mysql:people indicates that it should connect with the DBI module to the database people, which is housed on the local MySQL server. The second and third arguments to the connect() method are the username and password used to connect. Here user apache and the supersecret password are passed. If successful, connect() returns a database handle that is assigned to $dbh.If one day we decide that we want to migrate to another database, such as Oracle, we merely need to change mysql to oracle, and the rest of the script stays exactly the same, assuming the script is not executing a query that is specific to that database server—certainly the case with the scripts in this book. Design for portability! If connect() returns false, the script dies, printing the error string returned by the errstr() method.

If the script doesn't die, it prints a message stating that all is well. This gives us a warm, fuzzy feeling (for maximum fuzzy feeling, perhaps we should have printed "hello, world"). The last thing done is to execute the disconnect() method, allowing the Perl script and database to properly shut down the connection. This is only polite, and if you don't call disconnect(), the script may generate an error message, and the MySQL server will not like you. Executing this program from the shell produces:

```
$ ./connect.pl
Success: connected!
```

We've connected. But by itself, connecting isn't exceptionally useful, so let's see what records are in the age_information table. Create (or download) the script /var/www/bin/show_ages.pl. Online, it is at http://localhost/mysql/show_ages.pl or www.opensourcewebbook.com/mysql/show_ages.pl. Its contents are as follows:

```
#!/usr/bin/perl -w
# show_ages.pl
use DBI;
use strict;
# connect to the server, and if connect returns false,
# die() with the DBI error string
my $dbh = DBI->connect(´DBI:mysql:people´, ´apache´, ´LampIsCool´)
```

```
    or die "Can't connect: " . DBI->errstr();
# prepare the SQL, die() if the preparation fails
my $sth = $dbh->prepare(´SELECT * FROM age_information´)
    or die "Can't prepare SQL: " . $dbh->errstr();
# execute the SQL, die() if it fails
$sth->execute()
    or die "Can't execute SQL: " . $sth->errstr();
# loop through each record of our table,
# $sth->fetchrow() returns the next row,
# and we store the values in $ln, $fn and $age
my($ln, $fn, $age);
while (($ln, $fn, $age) = $sth->fetchrow()) {
   print "$fn $ln, $age\n";
}
# finish the statement handle, disconnect from the server
$sth->finish();
$dbh->disconnect();
```

Failure to connect is handled differently by this program. It executes connect() and uses the or to mimic an unless. If the connect() fails, the script dies. The script then prepares the SQL query "SELECT * FROM age_information". The query is just like that we might have typed into the MySQL program in the earlier examples (except the command terminator ; is not required in the prepare() method). The prepare()method returns a statement handle object that can then be used to execute the SQL query by calling the execute() method. Note that with each of these calls, failure is handled with the or die() code. The results of the SELECT query are handled with a while loop. The fetchrow() method returns a list of data for the next row of data that is returned by the query, which is then assigned to $ln (last name), $fn (first name), and $age. The information is then printed. At the end, the finish() method is executed to properly clean up and because it is the right thing to do. Running this from the shell produces:

```
$ ./show_ages.pl
Larry Wall, 48
Linus Torvalds, 31
Eric Raymond, 40
```

How might we enter a new record into the table? This code is in the file /var/www/bin/insert.pl. The entire contents of this program can be found online at http://localhost/mysql/insert.pl or www.opensourcewebbook.com/mysql/insert.pl. Here is the good part:

```
# print a nice dashed line
print ´-´ x 40, "\n\n";

# now, prompt for and read in the data for the new record
print ´Enter last name: ´;
chomp($ln = <STDIN>);
print ´Enter first name: ´;
chomp($fn = <STDIN>);
print ´Enter age: ´;
chomp($age = <STDIN>);
# prepare SQL for insert
$sth = $dbh->prepare(´INSERT INTO age_information
            (
               lastname,
               firstname,
               age
            )
         VALUES
            (
               ?,
               ?,
               ?
            )´)
      or die "Can't prepare SQL: " . $dbh->errstr();

# insert the record - note the arguments to execute()
$sth->execute($ln, $fn, $age)
      or die "Can't execute SQL: " . $sth->errstr();

# print another dashed line
print "\n", ´-´ x 40, "\n\n";
```

Before new data is inserted into the table, the script connects to the server and shows the current contents, just as in show_ages.pl. Then the script asks the user to enter the last name, first name, and age of the person for the new record and chomp()s the newlines.

Be sure to use those question marks as placeholders. This prevents the need to escape quotes and other nasty characters, thus making the code more secure. Also, in this case, the last name is defined in the tables as 20 characters of text. If the user enters more than 20 characters, only the first 20 are used—hence, no overflow problem. The next step is to prepare SQL for the INSERT query. Again, it looks much like what one would have typed in directly to SQL, with whitespace characters for readability, except that it has those three question marks. Those question marks are placeholders for the contents of the variables in the execute() method. The variables $ln, $fn, and $age are inserted into the query where the question marks are, in that order.

To check that the insert worked, the script displays the contents of the table after the INSERT is executed. Then the script cleans up after itself by finishing the statement handle and disconnecting from the MySQL server.

Executing that code produces:

```
$ ./insert.pl
Larry Wall, 48
Linus Torvalds, 31
Eric Raymond, 40
---------------------------------------
Enter last name: Ballard
Enter first name: Ron
Enter age: 31
---------------------------------------
Larry Wall, 48
Linus Torvalds, 31
Eric Raymond, 40
Ron Ballard, 31
```

Table Joins:

In the world of relational databases, data often has complex relationships and is spread across multiple tables. Sometimes it is necessary to grab information from one table based on information in another. This requires that the two tables be JOINed.

For an example, we create a new table in the people database called addresses that contains information about people's addresses. First, it must be created as follows:

```
mysql> CREATE TABLE addresses (
    ->     lastname CHAR(20),
    ->     firstname CHAR(20),
    ->     address CHAR(40),
    ->     city CHAR(20),
    ->     state CHAR(2),
    ->     zip CHAR(10)
    -> );
```

The table needs some data:

```
mysql> INSERT INTO addresses
    ->     (lastname, firstname, address, city, state, zip)
    ->     VALUES ("Wall", "Larry", "Number 1 Perl Way",
    ->             "Cupertino", "CA", "95015-0189"
    -> );

mysql> INSERT INTO addresses
    ->     (lastname, firstname, address, city, state, zip)
    ->     VALUES ("Torvalds", "Linus", "123 Main St.",
    ->             "San Francisco", "CA", "94109-1234"
    -> );

mysql> INSERT INTO addresses
    ->     (lastname, firstname, address, city, state, zip)
    ->     VALUES ("Raymond", "Eric", "987 Oak St.",
    ->             "Chicago", "IL", "60601-4510"
    -> );
```

```
mysql> INSERT INTO addresses
    ->    (lastname, firstname, address, city, state, zip)
    ->    VALUES ("Kedzierski", "John", "3492 W. 75th St.",
    ->           "New York", "NY", "10010-1010"
    -> );

mysql> INSERT INTO addresses
    ->    (lastname, firstname, address, city, state, zip)
    ->    VALUES ("Ballard", "Ron", "4924 Chicago Ave.",
    ->           "Evanston", "IL", "60202-0440"
    -> );
```

To verify the tables were populated, do this:

```
mysql> SELECT * FROM age_information;
+------------+-----------+------+
| lastname   | firstname |  age |
+------------+-----------+------+
| Wall       | Larry     |   46 |
| Torvalds   | Linus     |   31 |
| Raymond    | Eric      |   40 |
| Kedzierski | John      |   23 |
| Ballard    | Ron       |   31 |
+------------+-----------+------+
5 rows in set (0.00 sec)

mysql> SELECT * FROM addresses;
+----------+---------+-----------------+--------------+-----+----------+
| lastname |firstname|address          |city          |state|zip       |
+----------+---------+-----------------+--------------+-----+----------+
|Wall      |Larry    |# 1 Perl Way     |Cupertino     |CA   |95015-0189|
|Torvalds  |Linus    |123 Main St.     |San Francisco |CA   |94109-1234|
|Raymond   |Eric     |987 Oak St.      |Chicago       |IL   |60601-4510|
|Kedzierski|John     |3492 W. 75th St. |New York      |NY   |10010-1010|
|Ballard   |Ron      |4924 Chicago Ave.|Evanston      |IL   |60202-0440|
+----------+---------+-----------------+--------------+-----+----------+
```

5 rows in set (0.00 sec)

Now, on to the JOINs. Let's say we want to find out what city our under-40-year-old people live in. This requires looking up information in two tables: To find out who is under 40, we look in age_information, and to find out the city, we look in addresses. Therefore, we need to tell the SELECT command about both tables. Because both tables are being used, we need to be specific about which table a particular field belongs to. In other words, instead of saying SELECT city, we need to say what table that field is in, so we say SELECT addresses. city. The addresses.city tells MySQL that the table is addresses and the field is city.

Moreover, we need to hook the two tables together somehow—we do so with the following command by making sure the lastname from the addresses row matches the lastname from the age_information row. Ditto for the firstname. So, our command is:

```
mysql> SELECT addresses.city
   ->    FROM addresses, age_information
   ->    WHERE age_information.age < 40 AND
   ->          addresses.lastname = age_information.lastname
   ->     AND addresses.firstname = age_information.firstname;
+---------------+
| city          |
+---------------+
| San Francisco |
| NewYork       |
| Evanston      |
+---------------+
3 rows in set (0.02 sec)
```

In English, we are saying, "give me the city for all the people with ages less than 40, where the last names and first names match in each row." Let's grab the last names and zip codes for all those 40 and over, and order the data based on the last name:

```
mysql> SELECT addresses.lastname, addresses.zip
   ->    FROM addresses, age_information
   ->    WHERE age_information.age >= 40 AND
   ->          addresses.lastname = age_information.lastname AND
   ->          addresses.firstname = age_information.firstname
```

```
   ->     ORDER BY addresses.lastname;
+----------+------------+
| lastname | zip        |
+----------+------------+
| Raymond  | 60601-4510 |
| Wall     | 95015-0189 |
+----------+------------+
2 rows in set (0.02 sec)
```

Loading and Dumping a Database:

We can load a database or otherwise execute SQL commands from a file. We simply put the commands or database into a file—let's call it mystuff.sql— and load it in with this command:

```
$ mysql people < mystuff.sql
```

We can also dump out a database into a file with this command:

```
$ mysqldump people > entiredb.sql
```

For fun, try the mysqldump command with the people database (a gentle reminder: the password is LampIsCool):

```
$ mysqldump -uapache -p people
Enter password:
```

Notice that this outputs all the SQL needed to create the table and insert all the current records. For more information, see man mysqldump.

KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
(Established Under Section 3 of UGC Act, 1956)
Coimbatore-21

DEPARTMENT OF CS, CA & IT

UNIT- IV (Objective Type/ Multiple Choice Questions Each Question Carries One Mark)

OPEN SOURCE TECHNOLOGIES (17CAU504B)

S.No	Question	Option1	Option2	Option3	Option4	Answer
1	The father of MySQL is_____	Micheal Widenious	Bill Joy	Bill Gates	Stephanie Wall	Micheal Widenious
2	MySQL development Project has made its source code available under the terms of _____	LGPL	GPL	MPL	BSD	GPL
3	The letter M in the LAMP represents _____	method	memory	machine	MySQL	MySQL
4	MySQL comes with as standard with client libraries for _____	Java only	C	Java and C only	Perl, PHP and C	C
5	_____ is an open source standard query language database that is fast, reliable for applications of any size.	MySQL	SQL Server	Oracle 10g	Oracle 11g	MySQL
6	SQL stands for _____	Standard Query Language	System Query Language	Structured Query Language	Service Query Language	Structured Query Language
7	DBI stands for_____	DataBase Independent Interface	DataBase Information Interface	Design Independent Interface	Design Information Interface	DataBase Independent Interface
8	API stands for_____	Application Procedural Information	Application Procedural Interface	Application Programming Interface	Application Programming Information	Application Programming Interface
9	_____ is a container for related tables.	database	datawarehouse	datacentre	datamart	database
10	A table is a collection of_____	database	fields	rows	column	rows
11	Each row holding data for _____ record	one	two	three	four	one
12	Each record containing chunks of information called_____	rows	fields	cells	values	fields
13	BLOB stands for _____	Binary Large Object	Basic Logical Object	Binary Logical Object	Basic Large Object	Binary Large Object

14	Which of the following command is used to display the currently available databases?	SHOW DATABASES	DISPLAY DATABASES	PRINT DATABASES	VIEW DATABASES	SHOW DATABASES
15	_____ command is used to connect the database to MySQL	CREATE	USE	CONNECT	CONNECTDB	USE
16	The _____ command gives the information about the fields in a table	DESCRIBE	SHOW	DISPLAY	VIEW	DESCRIBE
17	DESCRIBE Command can be abbreviated as_____	DESCRIBE	DESC	DEBE	DCBE	DESC
18	_____ command used to change the value in an existing record.	INSERT	ALTER	DESCRIBE	UPDATE	UPDATE
19	The keyword SET is used with the command_____	UPDATE	SELECT	DELETE	ALTER	UPDATE
20	_____ method causes the Perl script to connect to the MySQL database.	combine()	connect()	allow()	redirect()	connect()
21	_____ method allowing the perl script and database to properly shutdown the connection	disallow()	shutdown()	disconnect()	terminate()	disconnect()
22	Which symbol in SELECT command shows values for all fields in the table?	*	+	@	#	*
23	DESCRIBE Command is equvalant to _____	SHOW DATABASES	SHOW TABLES	SHOW ROWS	SHOW COLUMNS	SHOW COLUMNS
24	DBH stands for_____	DataBase Handler	DataBase Holder	DataBase Hoster	DataBase Handle	DataBase Handle
25	If connect() return false, the script dies printing the error string returned by the _____ method	error()	errormsg()	errstr()	errorstring()	errstr()
26	The _____ method returns alist of data for the next row of data that is returned by the SELECT query	fetchdata()	fetchreord()	fetchlist()	fetchrow()	fetchrow()
27	MySQL runs on which OS?	Linux and Mac Operating System-X only	Any Operating System at all	Unix, Linux, Windows and other	Unix and Linux only	Unix, Linux, Windows and other
28	To remove duplicate rows from the result of a SELECT use the _____ keyword.	NO DUPLICATE	UNIQUE	DISTINCT	REMOVE DUPLICATE	DISTINCT
29	Which of the following can add a row to a table?	ADD	INSERT	UPDATE	ALTER	INSERT
30	To use MySQL on your computer, it need _____	FTP and Telnet	Some sort of client program to access the databases	a browser	Perl, PHP or Java	Some sort of client program to access the databases

31	In a LIKE clause you could ask for any value ending in "ter" by writing_____	LIKE %ter	LIKE &ter	LIKE *ter	LIKE ^ter	LIKE %ter
32	A NULL Value is treated as_____	balnk	zero	NULL value	No value	NULL value
33	MySQl is a _____	programming language	web designng language	technique for writing reliable program	Relational Database Management System	Relational Database Management System
34	which function used to get the current time in MySQL?	getTime()	Time()	Now()	showTime()	Now()
35	Which of the following is not a valid aggregate function?	COUNT	MIN	MAX	COMPUTE	COMPUTE
36	WhatSQL clause is used to restrict the rows returned by a query?	AND	WHERE	HAVING	FROM	WHERE
37	How much character are allowed to create database name?	55	72	64	40	64
38	Which of the following command is used creates a database?	CREATE DB student	CREATE DATABASE student	CREATE DBASE student	CREATE student DATABASE	CREATE DATABASE student
39	Which one will delete the table data as well as table stucture?	TRUNCATE	DROP	REMOVE	DELETE	DROP
40	The main MySQL program that does all the data handling is called_____	mysql.exe	mysql	mysqld	httpd	mysqld
41	A SELECT command without a WHERE clause returns ____	all records from a table	no records	SELECT is invalid without a WHERE clause	all records from a table that match the previous WHERE clause	all records from a table
42	Which statement is used to access an existing database?	USE database.name	USE databasename	USE	USE DBASE	USE
43	The MySQL command line tool format the results in which of the following format?	Rectangle	Square	Sphere	Circle	Rectangle
44	The "MySQL command line tool" formats are bounded by _____	+-*	+-|	+-/	+-}	+-|
45	What kind of replication is supported by the MySQL server?	multiple master replication	master to slave replication	single file based clustering	MySQL doesn't support replication	master to slave replication
46	Commands passed to the MySQL daemon are written in_____	programming language	web designng language	structured suery Language	machine language	structured suery Language
47	Which of the following is not a valid name for a column?	insert	name	age	gender	insert
48	Which of these commands will delete a table called "xxx"?	DROP xxx	DROP TABLE xxx	DELETE xxx	DELETE TABLE xxx	DROP TABLE xxx

49	Which of the following is not supported by MySQL?	temporary tables	table joining	stored procedures	regular expression	stored procedures
50	_____ command is used to undo a GRANT privilege	REVOKE	UNDO	UNGRANT	ROLLBACK	REVOKE
51	How many distinct, different values can hold in an enum field?	255	7	65535	2	65535
52	Which of the following command is not availble?	REVOKE	FETCH	UPDATE	SELECT	FETCH
53	Which of these field datatyppes would be best to hold a film title?	longblob	tinytext	mediumtext	longtext	tinytext
54	Which of these field datatyppes would be best to hold a .jpg file?	char	nchar	text	blob	blob
55	On executing DELETE command, if you get an error "foreign key constraint"- What does it imply?	Foreign key is not defined	Table is empty	Connectivity issue	Data is present in other table	Data is present in other table
56	How much storage space does DATETIME require?	4 Bytes	2 Bytes	8 Bytes	1 Byte	8 Bytes
57	User() function returns the current user's username and _____	password	hostname	both password and hostname	database name associated with that user	hostname
58	What is a primary Key_____	used to uniquely identify a row	alias for candidate key	used to identify a column	alias for foreign key	used to uniquely identify a row
59	A view is nothing but a _____ view or a stored query.	static	dynamic	virtual	real	virtual
60	_____ aggregate function is used to get the number of records or rows in a table.	COUNTROWS()	COUNT()	SUM()	NUMBER()	COUNT()

Unit-V

PERL

Introduction:

The fourth letter in the LAMP acronym stands for Perl (along with the other Ps). Although you don't have to understand any Perl to build a web site, it helps (Python and PHP work too). In many examples that follow throughout this book, we use the basics of Perl extensively. The purpose of this chapter is to familiarize the Perl novice with the basics. If you already know a C-like language, much of this will be familiar. If not, this chapter should introduce you to the concepts we use throughout the rest of the book. If you already know Perl, good for you! But this will be a rehash—feel free to move on.

Perl has become a popular all-purpose programming language because of its power and ease of use. Once you have mastered the language rules, you can do a lot with a little. One of the mottoes of Perl is Perl Makes Easy Tasks Easy and Hard Tasks Possible. Another is There's More Than One Way To Do It—TMTOWTDI, pronounced "Tim-Toe-Di."

Perl originated as a text processing language. Larry Wall, the creator, needed a language to manage and manipulate a database of text files. He designed Perl to be a language with built-in text processing by incorporating regular expressions and providing a number of text processing functions. There are and have been many other text processing languages: REXX, awk, sed, etc. But Perl has struck a chord, and not only with Open Source developers—it has been ported to all major operating systems and many minor ones. It has been referred to as the duct tape that holds the Internet together. One of the happy results of the spread of Perl is its portability; if created properly, Perl scripts will run on many operating systems with only minor changes, if any.

There are histories of Perl at www.perl.org/press/history.html and history.perl.org. The name Perl is an uncapitalized acronym for Practical Extraction and Report Language, or Pathologically Eclectic Rubbish Lister—TMTOWTDI.

As it evolved, Perl grew (some would say mutated) from a text processing language into a powerful, multipurpose object-oriented programming language used to solve all kinds of real-world problems: system administration, network programming, database management, and CGI programming.

Perl's syntax is C-like. Its operators, constructs, and lexical conventions are similar to those of C. However, some of the language rules, especially regular expressions, can be a bit overwhelming at first. It's common for eyes to cross a bit when looking at one of the ubiquitous strings of punctuation

representing a regular expression: /^ (?:To:)\s*(\S*)$/—not comic strip expletives, but the coin of the Perl realm. However, by concentrating on the basic rules, you can quickly learn to use the language to solve real problems—this is the beauty of the language. Of course, then you are suckered in and have become yet another Perl acolyte. Collectively, the trinity of C, C++, and Java are referred to as C-like languages.

We give only a basic overview here; the Camel Book [Wall+ 00] is the bible for Perl. Perl is the big, fat, top-of-the-line Swiss Army knife. We don't talk about all the nifty gadgets, only the major functionality. We can't teach you programming in this short chapter, so this is a high-level introduction. Again, we assume you know the basics of programming: variables, flow constructs (if statements, while loops), functions, reading from and writing to files, and so on.

Perl Documentation:

A wealth of information is available for Perl both on your Linux machine and online. There are several man pages:

```
$ man perl
$ man perlop
$ man perlsyn
```

A program called perldoc:

```
$ perldoc perldoc
$ perldoc -f print
```

And a Web site: www.perldoc.com/. There are also newsgroups, CPAN at www.perl.com and big, fat Perl books, should these not suffice.

Perl Syntax Rules:

First, the basic Perl syntax rules. Hang in there; although there are a plethora of hello, world's here, the somewhat unnerving friendliness and patience engendered by repeating this mantra will serve you well later when things get more involved.

A First Perl Program—hello, world:

As always, the best way to learn is by doing. For example, here is the ever popular print hello, world program:

```
#! /usr/bin/perl -w
# file: hello.pl
print "hello, world!\n";
```

The first line is a shell directive (also known as the hash/bang or shebang). This line informs the shell that this script uses /usr/bin/perl. Your distribution might have it in a different place, /usr/local/bin/perl, for example. standard error (stderr) output.

The -w switch causes Perl to print warnings to the The -w switch is recommended when you are developing Perl scripts, because it is possible to write programs that are syntactically correct but logically incorrect because of typos or just bad programming.

The terms stdout, stdin, and stderr are Unix file descriptors that, by default, point to the terminal (tty) you ran the program from but can be piped to a file or /dev/null. You can also cat the output to a file with the ">" redirection. The next line, which begins with #, is a comment. Comments extend from the first # to the end of line.

The print statement:
print "hello, world!\n";

executes the built-in print() function, which writes text to standard output (stdout). The "\n" is the newline character and prints a carriage return to standard output.Put the preceding text in a hello.pl text file and make it executable.

$ chmod a+x hello.pl
To execute it, type
$ ./hello.pl

The "./" tells the shell to execute the file in this directory (".") and not look for it in your PATH. If all goes as expected, you'll see this wonderfully reassuring message:

hello, world!

You may notice a few other things about the language from this example.

- Perl is free format—whitespace can be scattered about to make code more readable, as much or as little as you like.
- All statements must end in a semicolon (";") except the first line, which is not really a Perl statement but a shell escape to execute Perl.
- Variables do not have to be declared but can be.
- The Perl scripts are stored as text files. When executed ($ ./hello.pl), the source text file is first compiled into byte code, an intermediate form, not text and not binary. Perl then interprets the byte code, executing it. Therefore, each time a Perl script is executed, it is compiled and then interpreted (this is not the place to go into the compiled versus interpreted languages). This leads to the question, Is there a Perl compiler that compiles the Perl script into binary form? The answer is, Yes, but ... The Perl compiler is still in beta test, and it doesn't always work very well. Feel free to try it—perlcc.
- Built-in functions, such as print() can be invoked with or without parentheses. In the preceding, we are invoking print() without parentheses, but we could if we wanted to or needed to.

Another Example:

This program adds two numbers:

```
#! /usr/bin/perl -w
# file: simpleadd.pl
$a = 5;
$b = 6;
$c = $a + $b;
print "\$c is now: $c\n";
```

$a, $b, and $c are examples of scalar variables. Note that unlike other C-like languages, in Perl you don't have to explicitly declare them as variables. They are created for us. Put this into the file simpleadd.pl and execute it as earlier:

```
$ ./simpleadd.pl
$c is now 11
```

Because in this example we want the dollar sign to be printed for pedantic purposes, the output starts with $c. The dollar sign is escaped in the string. print "\$c is now: $c\n";

Declaring Variables with use strict:

Perl, as a scripting language, has looser rules than other "real" programming languages like C and Java. One such quality is the ability to create variables by simply using them, much as in shell programming. Still, it is considered good style to declare them. Doing so can preclude the silly typo problems that everyone is likely to incur from time to time:

```
$food = ´pizza´;
print "I am hungry for $foo!\n";
```

The variable $food is assigned, but $foo will be printed. Perl allows this without a syntax error because neither variable has to be declared. The output of the program will be "I am hungry for !" because $foo is undeclared and has no value. Perl will warn about this sort of thing if the -w flag is used, as it should be.

Another alternative to this situation is to force declaration of all variables before use, just as in the more strict, prescriptive compiled languages. For example:

```
use strict;
```

With this, variables must be declared before use, and this is done with the my() function as follows:

```
my $a;
my $b;
```

Variables can be declared and initialized at the same time:

```
my $a = 5;
my $b = 6;
```

More than one variable can be declared at the same time (the parentheses are required):

```
my($a, $b, $c);
```

If we do a use strict; and declare the variables, the earlier food example becomes:

```
use strict;
my $food = ´pizza´;
print "I am hungry for $foo!\n";
```

Since $foo is not declared with my(), the use of $foo returns a syntax error, forcing the correction of the typo before the program can

execute.

We recommend use strict;—it catches many common errors, requires a more disciplined programming mind-set, and is in general just good practice—so the examples from this point on use it.

The previous simple example becomes:

```
#! /usr/bin/perl -w
# file: simpleadd.pl
use strict;
my $a = 5;
my $b = 6;
my $c = $a + $b;
print "\$c is now: $c\n";
```

Variables:

Perl has several types of variables. We discuss the major ones. Perl has several minor data types, including loop labels, directory handles, and formats;

Scalar Variables:

A scalar variable is a variable that holds a single value.

It can hold the value of a number (integer or floating) or a string (from zero characters to enough characters to consume your virtual memory).

It can also hold the value of a reference, but that's beyond the scope of this book.

The value can change over time, and it can change type (for example, it can change from an integer to a float to a string and back to an integer)

Numeric literals are in the following format:

- Integers:

  - Decimal: 10 -2884 2_248_188 The underscore can be used to chunk numbers for ease of reading, but don't use the usual comma format, as in 2,248,188, because that means something completely different.

  - Octal: 037 04214

  - Hexadecimal: 0x25 0xaB80

- Floats:

  - Standard: 7.1 .8 9.

  - Scientific: 3.4E-34 -8.023e43

- String literals can be created with either single quotes or double quotes:

  - Single quotes: ´hello, world´

  - Double quotes (needed to evaluate variables and special characters): "hello, world\n", "\$c is now $c\n"

One must take care with quotes and scalar variables in Perl.

First, there is a difference between a tick (´) and a backtick (`), and a double quote ("). Mostly, we use ticks and double quotes—the backtick has a special purpose in Perl, which we will talk about later. To show how these affect scalar variables within quotes, you should assume that $name has a value:

$name = ´Larry Wall´;

Because scalar variables are replaced with their values within double-quoted strings, the string "Hello $name!" has the value Hello, Larry

Wall! If the $ is escaped with the backslash, "Hello \$name!", the value is Hello $name!. Single quotes do not replace the variable with its value, so ´Hello $name!´ has the value Hello $name!.

Array Variables:

An array is an ordered collection of scalars. Array names begin with the @ character:

@data = (´Joe´, 39, "Test data", 49.3);

Like scalars, they can be declared with my(), as in:

my @data = (´Joe´, 39, "Test data", 49.3);

The data in this array includes strings, an integer, and a floating point number. Unlike many languages, Perl allows the scalars in an array to have different types (they are all really the same data type: scalars). This example shows printing an array variable:

```
#! /usr/bin/perl -w
# file: array1.pl
use strict;
my @data = (´Joe´, 39, "Test data", 49.3);
print "Within double quotes: @data\n";
print "Outside any quotes: ", @data, "\n";
print ´Within single quotes: @data´, "\n";
```

This is what prints:
$ ./array1.pl

Within double quotes: Joe 39 Test data 49.3

Outside any quotes: Joe39Test data49.3

Within single quotes: @data

Placing the array within the double quotes inserts spaces between the array elements, while placing it outside does not, and placing it within single quotes prints the array name itself, not the elements thereof.

As in C, Perl arrays start at element 0. For @data, the elements are:

0: ´Joe´

1: 39

2: ´Test data´

3: 49.3

To access an individual array element, use the $array_name[index] syntax:

```
#! /usr/bin/perl -w
# file: array2.pl
use strict;
my @data = (´Joe´, 39, ´Test data´, 49.3);
print "element 0: $data[0]\n";
print "element 1: $data[1]\n";
print "element 2: $data[2]\n";
print "element 3: $data[3]\n";
```

That code produces:

```
$ ./array2.pl

element 0: Joe
element 1: 39
element 2: Test data
element 3: 49.3
```

A useful array-related variable is $#array_name, the last index of the array. For example:

```
#! /usr/bin/perl -w
# file: array3.pl
use strict;
my @data = (´Joe´, 39, ´Test data´, 49.3);
print "last index: $#data\n";
```

That code produces:
$ ./array3.pl

last index: 3

Array Functions:

Perl has a variety of functions that implement commonly performed array functions, saving you from having to write your own.

push() and pop():

The push() and pop() functions modify the right side of an array. The push() function adds elements to the right, and pop() removes the rightmost element:

```
#! /usr/bin/perl -w
# file: pushpop.pl
use strict;
my @a = (2, 4, 6, 8);
print "before: @a\n";
push(@a, 10, 12);
print "after push: @a\n";
my $element = pop(@a);
print "after pop: @a\n";
print "element popped: $element\n";
```

That code changes the array in the following fashion:
before: 2 4 6 8

after push: 2 4 6 8 10 12

after pop: 2 4 6 8 10

element popped: 12

unshift() and shift():

These functions are similar to push() and pop() except that they operate on the left side of the array.

```
#! /usr/bin/perl -w
# file: unshiftshift.pl
use strict;
my @a = (2, 4, 6, 8);
print "before: @a\n";
unshift(@a, 10, 12);
print "after unshift: @a\n";
my $element = shift(@a);
print "after shift: @a\n";
print "element shifted: $element\n";
```

That code alters the array:

```
$ ./unshiftshift.pl
before: 2 4 6 8
after unshift: 10 12 2 4 6 8
after shift: 12 2 4 6 8
element shifted: 10
```

sort() and reverse():

These functions are nondestructive—they don't change the arrays or lists that are being sorted. The sort() function sorts lists (as you might have guessed), and the reverse() function reverses lists (as you might have guessed):

```
#! /usr/bin/perl -w
# file: sortreverse.pl
use strict;
my @a = (´hello´, ´world´, ´good´, ´riddance´);
print "before: @a\n";
my @b = sort(@a);
print "sorted: @b\n";
@b = reverse(@a);
print "reversed: @b\n";
```

```
$ ./sortreverse.pl

before: hello world good riddance
sorted: good hello riddance world
reversed: riddance good world hello
```

Hash Variables:

Hashes (also known as associative arrays) are arrays that are indexed not by a number but by a string. In other words, instead of accessing an array by its position with an element like 0 or 1, we can access a hash with an index like name or age. For instance, data indexed as an array:

index	value
0	Joe
1	39

2 555-1212

3 123 Main. St.

4 Chicago

5 IL

6 60601

could instead be indexed as a hash:

key value

----- -----

name Joe

age 39

phone 555-1212

address 123 Main. St.

city Chicago

state IL

zip 60601

By indexing with a string, instead of a number, we can access the data structure by a more meaningful (at least to us humans) bit of information. A hash variable is defined to be a collection of zero or more key/value pairs (a hash can be empty, with zero pairs). The keys must be unique strings; the values can be any scalar. A hash variable name begins with the percent sign: %person.To create a hash, assign to it a list, which is treated as a collection of key/value pairs:

```
%person = (
   ´name´, ´Joe´,
   ´age´, 39,
   ´phone´, ´555-1212´,
   ´address´, ´123 Main St.´,
```

```
    ´city´, ´Chicago´,
    ´state´, ´IL´,
    ´zip´, ´60601´
);
```

Or you can use the => operator to make the code more readable:

```
%person = (
    name    => ´Joe´,
    age     => 39,
    phone   => ´555-1212´,
    address => ´123 Main St.´,
    city    => ´Chicago´,
    state   => ´IL´,
    zip     => ´60601´
);
```

If the key contains no space characters, like the example above, it does not need to be quoted if the => operator is used. Accessing a value in a hash is similar to accessing values in an array, indexed with the key string in curly braces, instead of a number within square brackets: $person{name}. The key does not need to be quoted within the curly braces (unless the key contains a whitespace character). For example:

```
#! /usr/bin/perl -w
# file: hash1.pl
use strict;
my %person = (
    name    => ´Joe´,
    age     => 39,
    phone   => ´555-1212´,
```

```
    address => ´123 Main St.´,
    city    => ´Chicago´,
    state   => ´IL´,
    zip     => ´60601´
);
print "$person{name} lives in $person{state}\n";
print "$person{name} is $person{age} years old\n";
```

That code produces:

```
$ ./hash1.pl
Joe lives in IL
Joe is 39 years old
```

A hash can be declared with the my() operator just like scalars and arrays, and they must be declared if we do a use strict;.

Hash Functions:

The most common way to process a hash is by looping though the keys. The keys are obtained by executing the built-in keys() function. For example:

```
#! /usr/bin/perl -w
# file: hash2.pl
use strict;
my %person = (
    name    => ´Joe´,
    age     => 39,
    phone   => ´555-1212´,
    address => ´123 Main St.´,
    city    => ´Chicago´,
    state   => ´IL´,
```

```
    zip     => ´60601´
);
my @k = keys(%person);
print "The keys are: @k\n";
```

That code produces:
$ ./hash2.pl

The keys are: state zip address city phone age name

What's this? The keys() function returns the keys in an apparently random order. They are not in the order created, nor the reverse, nor sorted in any obvious way, but according to how the key hashes. In Perl, the keys are returned in the order in which they are stored in memory. The keys can be sorted by using the (surprise!) sort() function. For example:

```
#! /usr/bin/perl -w
# file:  hash3.pl
use strict;
my %person = (
    name    => ´Joe´,
    age     => 39,
    phone   => ´555-1212´,
    address => ´123 Main St.´,
    city    => ´Chicago´,
    state   => ´IL´,
    zip     => ´60601´
);
my @k = keys(%person);
my @sk = sort(@k);
# or: @sk = sort(keys(%person));
print "The sorted keys are: @sk\n";
```

That code produces:
$ ./hash3.pl

The sorted keys are: address age city name phone state zip.

Operators:

Perl operator precedence, the order in which math things are executed, is C-like. For exact rules, see one of the recommended books or man perlop. We discuss each operator but not in precedence order.

Arithmetic Operators:

Perl arithmetic is quite C-like, as you might expect.

+	addition
-	subtraction
*	multiplication
/	division
%	modulus
**	exponentiation

For example:

Strictly speaking, ** is FORTRAN-like, not C-like, though that's C's loss.

```
#! /usr/bin/perl -w
# file: operators.pl
use strict;
my $i = 10;
```

```
my $j = 4;
print ´$i + $j = ´, $i + $j, "\n";
print ´$i * $j = ´, $i * $j, "\n";
print ´$i / $j = ´, $i / $j, "\n";
print ´$i % $j = ´, $i % $j, "\n";
print ´$i ** $j = ´, $i ** $j, "\n";
```

gives the results:

```
$ ./operators.pl
$i + $j = 14
$i * $j = 40
$i / $j = 2.5
$i % $j = 2
$i ** $j = 10000
```

One difference from C-like arithmetic is that when Perl divides integers, the result is automatically typed as floating point—not the usual C integer division, which truncates the result to an integer. That's much more sensible, as with Perl providing an exponentiation operator.
To emulate integer division, use the built-in int() function:

```
$integer_value = int($a / $b);
```

String Operators:

As befits its origins, Perl has powerful built-in string processing. Perl has two useful string operators:

.	string concatenation
X	string replication

For example:

```
#!/usr/bin/perl -w
# file: string.pl
use strict;
my $a = ´hello´;
my $b = ´world´;
my $msg = $a . ´ ´. $b;
print "\$msg : $msg\n";
$a = ´hi´;
$b = $a x 2;
my $c = $a x 5;
print "\$b : $b\n";
print "\$c : $c\n";
```

Executing that code gives:

```
$ ./string.pl
$msg : hello world
$b : hihi
$c : hihihihihi
```

True and False:

There are three false values in Perl:

´´	empty string
´0´	the string of the single character 0

0 the number 0

Every other value is true. So in Perl, the following are all true values:

1

10

´hi´

Numerical Comparison:

Perl has the usual numerical comparison operators, with one addition, the compare operator, <=>. All of these, with the exception of the compare operator, evaluate to either true (1) or false (empty string—not zero!). The following are the operators:

< less than

<= less than or equal to

> greater than

>= greater than or equal to

== equal to

!= not equal to

<=> compare

Here are some examples:

```
if ($i < 10) {
   print "\$i is less than 10\n";
}
if ($j >= 20) {
   print "\$j is greater than or equal to 20\n";
```

}

A common mistake (not just in Perl) is to use the assignment operator (=) when comparing numbers instead of the equivalence operator (==). The former actually changes one of the values—the statement $a = $b changes $a to be the same as $b, while the latter returns a logical true or false but changes neither $a nor $b. Remember that the two operations are different and have different effects.

The compare operator, <=>, evaluates to either 1, 0, or -1. The expression:

$a <=> $b

evaluates to:

1 if $a > $b

0 if $a == $b

-1 if $a < $b

String Comparison:

Since Perl scalars are either numbers or strings, and we already know how to compare numbers, we need a way to compare strings—string comparison operators. These are similar to the numeric operators in that, with the exception of cmp, they evaluate to either true or false:

lt	less than
le	less than or equal to
gt	greater than
ge	greater than or equal to
eq	equal to
ne	not equal to

cmp compare

String comparison is an ASCII-betic (similar to but not the same as alphabetic) comparison based on the ASCII values (man ascii) of the characters.

```
if ($name gt ´Joe´) {
   print "$name comes after Joe alphabetically\n";
}
if ($language ne ´Perl´) {
   print "Your solution is a 2 step process. Step 1, install Perl.\n";
}
```

The string comparison operator, cmp, evaluates to either 1, 0, or -1. The expression:

$a cmp $b

evaluates to:

1 if $a gt $b

0 if $a eq $b

-1 if $a lt $b

Increment and Decrement:

Perl has the following autoincrement and autodecrement operators:

++ autoincrement

-- autodecrement

These operators provide a simple way to add 1 to a variable (with ++) or subtract 1 from a variable (--). The location of the operator is important. If placed on the left side of the variable, it increments before the variable is used (preincrement); on the right side, it increments after the variable is used (postincrement).

For example:

```
#! /usr/bin/perl -w
# file: increment.pl
use strict;
my $i = 10;
my $j = ++$i;
print "\$i = $i, \$j = $j\n";
$i = 10;
$j = $i++;
print "\$i = $i, \$j = $j\n";
```

Executing that code gives:

```
$ ./increment.pl
$i = 11, $j = 11
$i = 11, $j = 10
```

To illustrate pre- and postincrements, consider the statement:

```
$j = ++$i;
```

$j is assigned the result of ++$i. This preincrement (since the ++ is on the left side of $i) causes the variable $i to be incremented first to 11; then the value of $i is assigned to $j. The result is that both $i and $j have the value 11.

Conversely, consider this statement:

```
$j = $i++;
```

In the postincrement, the ++ is on the right side of $i. $i is incremented after its value is taken. First, $j is assigned the current value of $i, 10; then $i is incremented to 11.

The decrement operator (--) works in a similar manner: If the -- is on the left side, it is a predecrement, subtracting 1 first, if it is on the right side, it is a postdecrement, subtracting 1 last.

Logical Operator:

Perl's logical operators are similar to C and C++, with the addition of three very readable ones:

&&	logical and
||	logical or
!	logical not
And	logical and
Or	logical or
Not	logical not

You can use either or both, depending on your tastes. There is a difference in precedence. The operators and, or, and not have extremely low precedence—they are at the bottom of the precedence table. See man perlop for all the details. Here are a couple of examples of their use:

```
if ($i >= 0 && $i <= 100) {
   print "\$i is between 0 and 100\n";
}
if ($answer eq ´y´ or $answer eq ´Y´) {
   print "\$answer" equals ´y´ or ´Y´\n";
}
```

The low precedence of and, or, and not can work in our favor, allowing us to drop parentheses in some cases, but it can cause us problems as well. Here are some gotchas:

```
a = $b && $c;              # like: $a = ($b && $c);     good!
$a = $b and $c;            # like: ($a = $b) and $c;    oops!
$a = $b || $c;             # like: $a = ($b || $c);     good!
```

```
$a = $b or $c;              # like: ($a = $b) or $c;     oops!
```

Flow-Control Constructs:

Flow-control constructs are the familiar if, else, while, etc. Do this if this condition is met, do that if not—flow control. An important thing about these constructs in Perl, unlike C, is that the curly braces, {}, are required. Omitting them is a syntax error.

Conditional Constructs:

The if statement is the most common conditional construct. The syntax for it is:

```
if (condition) {
   statements
} else {
   statements
}
```

If the condition in parentheses evaluates as true, the first set of statements is executed; otherwise, the second set of statements is executed. The else part is optional. The condition can be any statement that evaluates to true or false, such as $a < $b, or far more complicated things.

Nested if statements have this syntax:

```
if (condition1) {
   statements
} elsif (condition2) {
   statements
} else {
   statements
}
```

Here is an example of the if statement:

```
if ($answer eq ´yes´) {
   print "The answer is yes\n";
```

```
} else {
   print "The answer is NOT yes\n";
}
```

Looping Constructs:

The while Loop:

A common looping construct, it has the syntax:

```
while (condition) {
   statements
}
```

The condition is tested, and if true, the statements are executed; then the condition is tested, and if true, the statements are executed; then the condition is tested ... You get the picture. The loop terminates when the condition is false. For example:

```
$i = 1;
while ($i <= 5) {
   print "the value is: $i\n";
   $i++;
}
```

Executing that code produces:

```
the value is 1
the value is 2
the value is 3
the value is 4
the value is 5
```

One might process an array variable (keep in mind that $#names is the last index of the array @names):

```
#! /usr/bin/perl -w
# file: whilearray.pl
use strict;
my @names = (´Joe´, ´Charlie´, ´Sue´, ´Mary´);
my $i = 0;
while ($i <= $#names) {
   print "Element $i is $names[$i]\n";
   $i++;
}
```

That code produces:

```
$ ./whilearray.pl

Element 0 is Joe
Element 1 is Charlie
Element 2 is Sue
Element 3 is Mary
```

Or one might process a hash sorted by its keys:

```
#! /usr/bin/perl -w
# file: whilehash.pl
use strict;
my %capitals = (
   Illinois   => ´Springfield´,
   California => ´Sacramento´,
   Texas      => ´Austin´,
```

```
   Wisconsin  => ´Madison´,
   Michigan   => ´Lansing´
);
my @sk = sort(keys(%capitals));
my $i = 0;
while ($i <= $#sk) {
   print "$sk[$i]:  \t$capitals{$sk[$i]}\n";
   $i++;
}
```

The \t is the tab character. That code produces:

```
$ ./whilehash.pl

California:     Sacramento
Illinois:       Springfield
Michigan:       Lansing
Texas:          Austin
Wisconsin:      Madison
```

The for Loop:

This loop has the syntax

```
for (init_expression; condition; step_expression) {
   statements
}
```

The init_expression is executed first and only once. Then the condition is tested to be true or false. If true, the statements are executed, then the step_expression is executed, and then the condition is tested. The loop stops when the condition evaluates as false. Our earlier while loop example could have been written to process @names as:

```
for ($i = 0; $i <= $#names; $i++) {
```

```
    print "Element $i is $names[$i]\n";
}
```

That gives the same result, and the %capitals example could be written using the for as:

```
for ($i = 0; $i <= $#sk; $i++) {
    print "$sk[$i]:  \t$capitals{$sk[$i]}\n";
}
```

There's more than one way to do it.

The foreach Loop:

Perl has a nifty loop used to process lists and arrays called the foreach loop. It has this syntax:

```
foreach scalar_variable (list) {
   statements
}
```

The statements are executed for each item in the list (hence the name), meaning we don't have to know how long the list is or check to see whether the end has been reached. For example:

```
#! /usr/bin/perl -w
# file: foreach1.pl
use strict;
my @a = (2, 4, 6, 8);
foreach my $i (@a) {
   print $i, ´ ** ´, $i, ´ = ´, $i ** $i, "\n";
}
```

That code produces:

```
$ ./foreach1.pl

2 ** 2 = 4
4 ** 4 = 256
6 ** 6 = 46656
8 ** 8 = 16777216
```

Regular Expressions:

Regular expressions (regexes) are a powerful, strange, beautiful, and complex part of Perl. These can match a string against a general pattern, perform string substitution, and extract text from a string. This sounds simple, but within this simple notion, a world of complexity lies. Much of Perl's power and usefulness derives from the use of regular expressions. To match a string, use the following syntax:

$string_variable =~ /pat/

That is =~ (equal-tilde), not = ~ (equal-space-tilde).

This returns true if the variable $string_variable matches the pattern pat, false if it does not. For a string to match a pattern, only a portion of the string needs to match, not the entire string.

For example:

```
if ($name =~ /John/) {
   print "\$name matches `John´\n";
}
```

If the variable $name matches the pattern "John", the if condition is true and the result is printed. In order for $name to match the regex

"John", only a portion of the string needs to match. For instance, all the following values for $name would match the regex `John´:

John

John Lennon

Andrew Johnson

Memory:

Perl has a mechanism that allows it to remember patterns previously matched. Expressions that have been matched by a regex within a set of parentheses are stored in the special variables $1, $2, $3, and so on.

This memory function can also help with matching recurring text by using the related variables \1, \2, and \3. When one set of parentheses is used, the text within it is stored in $1:

```
#!/usr/bin/perl -w
# file: memory1.pl
use strict;
```

```
my $name = ´John Doe´;
if ($name =~/^(..)/) {
   print "^(..) : $1\n";
}
if ($name =~/^(\w+)/) {
   print "^(\\w+)  :  $1\n";
}
if ($name =~/(\w+)$/) {
   print "(\\w+)\$  :  $1\n";  }
```

Executing that code produces:

```
$ ./memory1.pl

^(..) : Jo

^(\w+) : John

(\w+)$ : Doe
```

If more than one set of parentheses is used, the first set will remember its characters in $1, the second set in $2, the third set in $3, and so on. For instance, you could have a file full of records with three fields, containing an account number, name, and phone number, and write a regex that would go through the record and print each field separately:

```
#!/usr/bin/perl -w
# file: memory2.pl
use strict;
# create a record, 3 fields, colon-separated
my $record = ´32451:John Doe:847 555 1212´;
# see if the record contains:
#  beginning of the string
#  followed by 0 or more of any character but \n
#    (remembered into $1)
```

```
#  followed by a colon (`:´)
#  followed by 0 or more of any character but \n
#    (remembered into $2)
#  followed by a colon (`:´)
#  followed by 0 or more of any character but \n
#    (remembered into $3)
#  followed by the end of the string
if ($record =~/^(.*):(.*):(.*)$/) {
    print "The record is:\n";
    print " account number: $1\n";
    print " name:           $2\n";
    print " phone number:   $3\n";
}
```

Executing that code produces:

```
$ ./memory2.pl

The record is:

    account number: 32451

    name:           John Doe

    phone number:   847 555 1212
```

This record is only one entry long for this example, but we could have processed more than one entry. One could also imagine creating a regex that matched "John Doe," his phone number, or his account number.

Regular expressions are incredibly useful—now you can write a script that goes through your old e-mail files, picking out phone numbers. To make it robust, you'd have to figure out how to disregard parentheses, dashes, and dots, but that isn't hard to do. Or you could write a script to scan your Apache logs for a certain IP address.

Functions:

User-defined functions are created as follows:

```
sub function_name {
   # body
}
```

In Perl, the terms function and subroutine are used interchangeably. Some programmers like to call them functions, other prefer subroutine.

```
function_name()
```

For example:

```
#! /usr/bin/perl -w
```

To invoke the function, do this:

Some say a function is a subroutine that returns a value, but this is not a firm definition.

```
# file: function1.pl
sub say_hello {
   print "hello, world!\n";
}
say_hello();
```

Executing that program produces:

```
$ ./function1.pl
hello, world!
```

By convention, functions are defined above the point where first called but can be defined after it.

```
#! /usr/bin/perl -w
```

```
# file: function2.pl

say_hello();

sub say_hello {

   print "hello, world\n";

}
```

Usually, we prefer to define functions before they are used, especially because it allows us to use the lazy way of calling functions by dropping the parentheses: say_hello;.

File I/O:

Since the Web is interactive, Perl scripts which don't interact with the outside world will be of limited use. The scripts have to be able to accept input and write output. This is where file input/output comes in, or File I/O.

Standard Output:

We have already been sending text to standard output (probably right on the terminal on the monitor you're looking at now) using the print() function:

print "hello, world\n";

Standard Input:

You can imagine how it might be useful to be able to read data from the outside world. This can be done using standard input, <STDIN>, which is typically the keyboard, though it can be redirected to be a file or something else. To read input into a scalar variable, up to and including the newline character \n, you need only to add $line = <STDIN>;. More usefully, this example reads the next line of data into $line, including the newline character:

```
#! /usr/bin/perl -w
# file: stdin1.pl
use strict;
print "Enter your name: ";
my $name = <STDIN>;
print "Hello $name!\n";
```

Executing that program produces:
$ ./stdin1.pl

Enter your name: J. Random Luser

Hello J. Random Luser

!

The exclamation point (!) appeared on the next line after the name. That is because <STDIN> reads up to and including the newline character, so $name contains the text "J. Random Luser\n."

To remove the newline character, chomp() it:

$name = <STDIN>;

chomp($name);

Or:

chomp($name = <STDIN>);

For example:

```
#! /usr/bin/perl -w
# file: stdin2.pl
use strict;
my $name;
print "Enter your name: ";
chomp($name = <STDIN>);
print "Hello $name!\n";
```

Executing that code produces:

```
$ ./stdin2.pl
Enter your name: J. Random Luser
Hello J. Random Luser!
```

You can also read the input into an array variable:

```
@all_lines = <STDIN>;
```

That reads all remaining lines of standard input into the array @all_lines. Each line of input (including the newline character) is now an element of @all_lines. This behavior, reading until the end of file in one statement, is so important that it has a special term: a STDIN slurp.

```
#! /usr/bin/perl -w
# stdin3.pl
use strict;
print "Enter your text: \n";
```

```
my @all_lines = <STDIN>;
my $i = 0;
while ($i <= $#all_lines) {
   print "line $i: $all_lines[$i]";
   $i++;
}
```

That code produces:

```
$ ./stdin3.pl

Enter your text:

hello

world

good bye

^D

line 0: hello

line 1: world

line 2: good bye
```

The ^ D character (Ctrl + D) is the end-of-file character for standard input. Also, each line of text contains the newline character, so when we print each line of text in line 6 of the program, we do not need to include the newline character. The text can be read, and the newlines removed on each line, by passing the array variable as the argument to chomp():

```
@all_lines = <STDIN>;

chomp(@all_lines);
```

Or:

```
chomp(@all_lines = <STDIN>);
```

Perl programmers often slurp and chomp() but otherwise have good manners.

Additional Perl Constructs:

Perl allows you to write code that reads a lot like English. It also allows you to write code that's completely unreadable, but that's another issue.

The unless Statement:

The unless is like the if, except the logic is reversed. The basic syntax is:

```
unless (condition) {
   statements
}
```

The condition is tested, and if it is false, the statements are executed.

Thus, this if statement:

```
if (not $done) {
   print "keep working!\n";
}
```

can be written as:

```
unless ($done) {
   print "keep working!\n";
}
```

The until Loop:

The until loop is like the while loop, except the logic is reversed. The basic syntax is:

```
until (condition) {
   statements
}
```

The condition is tested, and if it is false, the statements are executed. Then the condition is tested, and if it is false ...

Thus, this while loop:

```
while (not $done) {
   keep_working();
}
```

can be rewritten as:

```
until ($done) {
   keep_working();
}
```

Quick Introduction To Object-Oriented Programming:

There is no way to avoid it—we have to talk about object-oriented programming (OOP) using Perl. Object-oriented programming is not a big deal though; it is just a way to give an object (a thing) actions (behaviors, also known as methods). Let's say we have an object named $car. We can drive the car by executing that object's drive() method, using the nifty arrow notation: $car->drive().

A Perl pragma is a message to the compiler to behave in a way that is not the default—in the use pragma, we are telling the compiler to use a specific Perl module that it does not normally include, telling Perl to locate the module file CGI.pm and do some OOP magic, making all the methods, or object functions, available to us. For instance, in the CGI module, a method named header() is defined, and because we told Perl we want to use the CGI module, we can call this method.

Once we have used a module, we can then usually create an object of that type by executing new, which is a method that is defined to construct the object:

$obj = new CGI;

Once an object is created, methods can be invoked by using the arrow notation:

$obj->header();

$obj->pre();

$obj->b();

This syntax means "execute the header() (or pre() or b()) method using the object named $obj, which is an object of class CGI." Hundreds of modules are available for Perl. You can find them all for free at the Comprehensive Perl Archive Network(CPAN)—www.cpan.org/. We suggest that you check out what is there. You might find that the problem you are trying to solve, or will need to solve in the future, has a module written for it that could make your life easier.

Installing a module is a snap. First, find the module you want at www.cpan.org. Let's say we want the module Nifty. We find the latest version of Nifty, let's say version 1.03. We download the tarball from CPAN named Nifty-1.03.tar.gz and save it somewhere convenient, let's say /tmp. Then, all we do is this:As with most things Perl, there is a Better Way to install modules—use the CPAN module. In this book, we demonstrate installing Perl modules the explicit way, as just shown, in case you are unable to get CPAN to install the module you need. If you are interested, check out perldoc CPAN. Here is a quick example of installing the preceding module:

perl -MCPAN -e shell

cpan> install Nifty

That's it! Easy as pie.

cd /tmp

tar xzvf Nifty-1.03.tar.gz

cd Nifty-1.03

perl Makefile.PL

make

make test

make install

What We Didn't Talk About:

As has been said ad nauseam, much of the complex functionality of Perl is beyond the scope of this book; the Camel Book is fat and thick, and we couldn't possibly do it justice here. As you continue your webmaster and Linux experience, you will want to learn more about Perl or a similar scripting language Among the topics we didn't cover, the following might be of interest:

- Special variables,
- especially $_
- Reading input with <>
- Assignable lists
- Slices
- String functions such as index(), rindex(), substr(), and length()
- References and complex data types

KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
(Established Under Section 3 of UGC Act, 1956)
Coimbatore-21

DEPARTMENT OF CS, CA & IT

UNIT- V (Objective Type/ Multiple Choice Questions Each Question Carries One Mark)

OPEN SOURCE TECHNOLOGIES (16CAU504B)

S.No	Question	Option1	Option2	Option3	Option4	Answer
1	Perl is a _____	System Program	Programming language	Application Program	Relational Database	Programming language
2	The fourth letter in the LAMP represents _____	PERL	Prototype	Procedure	Program	PERL
3	PERL stands for_____	Practical Extraction and Report Language	Procedural Extraction and Report Language	Practical Extension and Report Language	Procedural Extension and Report Language	Practical Extraction and Report Language
4	Perl orginated as a _____ processing language.	text	file	database	content	text
5	CPAN stands for _____	Comprehensive Procedural Archive Network	Comprehensive Perl Archive Network	Common Perl Archive Network	CommonProcedural Archive Network	Comprehensive Perl Archive Network
6	Perl program stored with an _____ extension	.perl	.pl	.per	.prl	.pl
7	____ tells the shell to execute the file	./	.#	.\	.$	./
8	_____ function writes text to standard output	write()	show()	print()	display()	print()
9	Perl scripts are stored as_____	text files	batch files	binary files	executable	text files
10	In Perl, all statements must end in a ____	:	.	,	;	;
11	The Perl source code file is first compiled into _____	machine code	ascii code	byte code	object code	byte code
12	A _____ scalar variable that holds a single value	array	scalar	hash	associative	scalar
13	String literals can be created with _____	single quotes	double quotes	single quotes and double quotes	without quotes	single quotes and double quotes
14	The _____ can be used to chunk numbers for easy of reading	dollar	underscore	comma	hyphen	underscore
15	What is the base value for octal numerical literal?	10	2	8	16	8

16	Which of the following is not a string literal?	"Hello"	Good'	"Good"	Hello	Hello
17	_____ is an ordered collection of scalars	array	group	list	file	array
18	_____ function adds elements to the right of an array.	add()	push()	insert()	enter()	push()
19	The push() function adds elements to the _____ right of an array	specified location	specified value	right	left	right
20	Variables can be declared using _____ function	declare()	decl()	my()	dec()	my()
21	_____ function removes the rightmost element from an array.	pop()	del()	delete()	remove()	pop()
22	_____ are arrays that are indexed not by a number but by a string	Unions	Hashes	Structures	Classes	Hashes
23	Hashes are also known as _____ arrays	descriptive	associative	functioal	Procedural Extension and Report Language	associative
24	Which operator is used in sring concatenation?	.	+	@	&	.
25	Which operator is used in sring replication?	+	-	x	*	x
26	Which of the following ia a compare operator in Perl?	==	<=>	>=	<=	<=>
27	$i=10;$j=$i++; What is the value of i and j?	i=11,j=10	i=10,j=10	i=11,j=11	i=10,j=11	i=11,j=10
28	$a=5;$b=$a--; What is the value of a and b?	a=5,b=4	a=5,b=5	a=4,b=4	a=4,b=5	a=4,b=5
29	$i=10;$j=$++i; What is the value of i and j?	i=11,j=10	i=10,j=10	i=11,j=11	i=10,j=11	i=11,j=11
30	$a=5;$b=$--a; What is the value of a and b?	a=5,b=4	a=5,b=5	a=4,b=4	a=4,b=5	a=4,b=4
31	Regular expressions are otherwise called as____	relational expressions	regexes	logical expressions	assignment expressions	regexes
32	_____ is the new line character in Perl	\n	\t	\r	\f	\n
33	The character class is created using _____	( )	" "	[ ]	{ }	[ ]
34	In Perl, the term function is also referred as _____	method	procedure	class	subroutine	subroutine
35	Arguments are passed into functions through the spcial array____	#_	@_	$_	&_	@_

36	The _____ character is the end-of-file character for the standard input	^D	^E	^S	^F	^E
37	What is the keyboard shortcut for ^D?	Ctrl+D	Ctrl+E	Ctrl+F	Ctrl+S	Ctrl+D
38	_____ function is used to open a file	start()	begin()	open()	create()	open()
39	Which of the following datatypes are preceded by a dollar sign in Perl?	Array	Scalar	Local	Hash	Scalar
40	How to delete a key/value pair to a hash?	using end function	using truncate function	using delete function	using remove function	using delete function
41	Which of the following statement jump to the statement labeled with LABEL?	goto EXPR	goto LABEL	goto NAME	goto ADDRESS	goto LABEL
42	Which of the following operator returns true if the left assignment is stringwise less than or equal to the right assignment?	lt	gt	le	ge	lt
43	Which of the following operator returns a list of values counting from the left value to the right value?	.	x	..	++	..
44	Which of the following function returns epoch time?	local time	gmt time	time	strf time	time
45	Which of the following code create a reference for a variable?	$ref=\$FOO	$ref=\@ARGV	$ref=\%ENV	$ref=\&Print Hash	$ref=\$FOO
46	_____ operator is used to create sequential arrays	arithmetic	relational	logical	range	range
47	What is the range operator?	-	..	--	to	..
48	Which of the following operator returns true if the left assignment is string wise not equal to the right assignment?	lt	ne	le	ge	ne
49	FH stands for_____	File Handles	Field Handle	Folder Handle	Format	File Handle
50	The statement open(FH ,<`abc.txt`) _____	opens the file abc.txt for overwritting	opens the file abc.txt for reading	opens the file abc.txt for rwritting	opens the file abc.txt for appending	opens the file abc.txt for reading
51	Which of the following is used in Perl?	elseif	elsif	elife	eleif	elsif
52	All file handles are named with _____	uppercase letters	lowercase letters	Titlecase letters	upper and lowerccase letters	uppercase letters
53	The > symbol in the open( ) function tells Perl that the file is to be opened in _____ mode	read	write	overwrite	append	write
54	Which operator can be used, when comparing numbers for equivalence?	=	==	both = and ==	<=>	==
55	_____ function sorts the given list	sort()	sortlist()	sorting()	tosort()	sort()

56	unshift() function is similar to _____ function	create()	push()	pop()	my()	push()
57	Which feature of Perl provides code reusability?	function	abstraction	inheritance	encapsulation	inheritance
58	Which of the following shows the warnings in torder t oreduce or avoid errors	- warn	-w	- warnings	- we	-w
59	"The method defines in the parent class will always override the methods defined in the child class" , which is referred as _____	inheritence	polymorphism	encapsultion	abstraction	polymorphism
60	unshift() and shift() functions are oerate _____ of an array	specified location	specified value	rightside	leftside	leftside

Reg No. ……

[17CAU504B]

KARPAGAM ACADEMY OF HIGHER EDUCATION

(Deemed to be University Established under Section 3 of UGC Act, 1956)

Eachanari Post, Coimbatore - 641 021, India

BCA DEGREE EXAMINATION

(For the candidates admitted from 2017 onwards)

Fifth Semester

First Internal Examination- July 2019

Open Source Technologies

Time: 2 Hours **Maximum: 50 Marks**

Date:

Part - A (20 x 1 =20 Marks)
(Answer all the Questions)

1. OSS stands for _____
   a.Open Source Software b.Open Standard Software
   c.Open Standard Service d.Open Source Service
2. In _____ the source code used in the software is available to anyone to examine, evaluate and adapt.
   a.close source **b.open source** c.shared source d.proprietary source
3. Which of the following is not an example of open source software?
   a.Linux b.Mozilla **c.Windows** d.Apache
4. The human readable code of the program is known as _____
   a.**source code** b.byte code c.machine code d.object code
5. In _____ the source code used in the software is available to anyone to examine, evaluate and adapt.
   a.close source b.shared source **c.open source** d.proprietary source
6. XP stands for_____
   a.External Programming b.Extreme Programming c.Extensible Programming d.Executable Programming
7. Which phase of the Spiral model determine objectives and constraints of the project and define the alternatives?
   a.Risk analysis **b.Planning** c.Engineering d. Customer
8. _____ helps to manage the files and codes of a project when several people are working on the project at the same time.
   a.CVS b.RPM c.APT d.SVN
9. _____ is a computer program that is used to debug other programs.
   a.Compiler b.Interpreter **c.Debugger** d.Loader
10. LibreSource is a type of _____
    a.debugger b.compiler c.interpreter **d.bugtracker**

11. Eric. Raymond wrote the Cathedral and the Bazaar model in _____
    a) 1998 b) 1995 c) 1994 d) **1997**
12. _____ is the free software project funded by Netscape to build a WWW browser.
    a) **Mozilla** b)Internet Explorer c)Chrome d)Opera
13. _____ model adds an element of risk analysis to the software development
    a)process. b) Scrum c)Water fall d) **Spiral**
14. _____ activity is the transition of the design specification into a software program.
    a) Analysis **b)Coding** c)Testing d)Maintenance
15. The principle of maintaining well defined boundaries between components is called _____
    a)availability **b)modularity** c) Extension d) subset
16. _____ is a software that is floated to see if there is an interest
    a) **Vaporware** b) Freeware c) Free Software d) Open Source Software
17. The software in wide distribution that has a version number less than _____
    a) 3.0 b)2.0 c)**1.0** d)0.5
18. OSH stands for _____
    a) Open Source Host **b) Open Source Hardware**
    c) Open Source Hub d) Open Source Hierarchy
19. The development of open source hardware was initiated in _____
    a) 2004 b) 2000 **c) 2002** d) 2001
20. F in FOSS stands for _____
    a)feasible b)full c)free d) field

Part - B (3 x 2=6 Marks)
(Answer all the Questions)

21. What is open source technology?

    Open-source software (OSS) is computer software with its source code made available with a license in which the copyright holder provides the rights to study, change, and distribute the software to anyone and for any purpose.
    Open Source or Open Source Software is different from proprietary software. In open source, the source code used in software is made available to anyone to examine, evaluate and adapt it.
    Eg: Linux, Apache, BSD, Wikipedia, Mozilla

22. Define Copy right.

Copyright law protects original creative works, such as software, video games, books, music, images, and videos. Copyright law varies by country. Copyright owners generally have the right to control certain unauthorized uses of their work (including the right to sue people who use their copyrighted work without permission). As a result, certain images and other copyrighted content may require permissions or licenses, especially if you use the work in a commercial setting.

For example, even if you have permission to use an image, you may need additional permission to use what is in the image (e.g., a photo of a sculpture, a person, or a logo) because someone else's copyright, trademark, or publicity rights might also be involved. You are responsible for obtaining all of the permissions and licenses necessary to use the content in your specific context.

However, even copyright-protected works can be lawfully used without permission from the copyright holder in certain circumstances. The Wikipedia entry on copyright law contains a useful overview of copyright law, including fair use and other exceptions to copyright law.

23. Differentiate open source versus closed source.

Closed Source	**Open Source**
Licensor distributes object code only; Source code is kept safe.	Licensor distributes source code
Modifications are prohibited	Modifications are permitted
All upgrades, support and development are done by the licensor.	Licensee may do its own development and support or hire any third party to do it.
Fees are for the software license, maintenance and upgrades	Fees if any are for integration, packing, support and consulting.
Sublicensing is prohibited or is a very limited right	Sublicensing is permitted
Example: Microsoft Windows	Example: Wikipedia

Part - C (3 x 8=24 Marks)
(Answer all the Questions)

24. A. List and discuss the open source principals.

The following are the principles of open source software. The distribution terms of open-source software must comply with the following criteria:

1. Free Redistribution

The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.

2. Source Code

The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost preferably, downloading via the Internet without charge.

3. Derived Works
The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

4. Integrity of The Author's Source Code
The license may restrict source-code from being distributed in modified form *only* if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.

5. No Discrimination Against Persons or Groups
The license must not discriminate against any person or group of persons.
Rationale:
In order to get the maximum benefit from the process, the maximum diversity of persons and groups should be equally eligible to contribute to open sources. Therefore we forbid any open-source license from locking anybody out of the process.

6. No Discrimination Against Fields of Endeavor
The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

7. Distribution of License
The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

8. License Must Not Be Specific to a Product
The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.

9. License Must Not Restrict Other Software
The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.

10. License Must Be Technology-Neutral

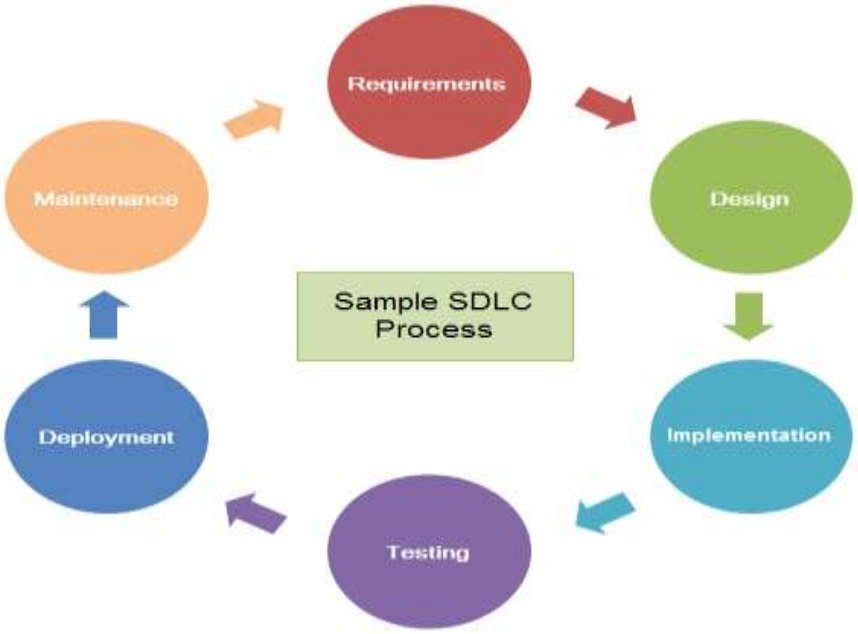
No provision of the license may be predicated on any individual technology or style of interface.

Or

b. Explain the software life cycle paradigm.

The Life- Cycle Paradigm:

The Software **development life cycle** (**SDLC**), also referred to as the application development life-cycle, is a term used in systems engineering, information systems and software engineering to describe a process for planning, creating, testing, and deploying an information system. The work is started at the system level and passes of analysis, design, coding, testing and maintenance. Six activities represent the overall development



process.

- **System Engineering and Analysis:** This activity is characterized by system level requirements gathering for all system elements.

- **Software Requirements Analysis:** This activity is usually executed together with the customer to collect the software requirements. The main goal of this phase is to document all function, performance and interfacing requirements for the software.

- **Design:** When creating the design of the software system, requirements are transformed into a representation of software that can be assessed for quality before the actual design begins.

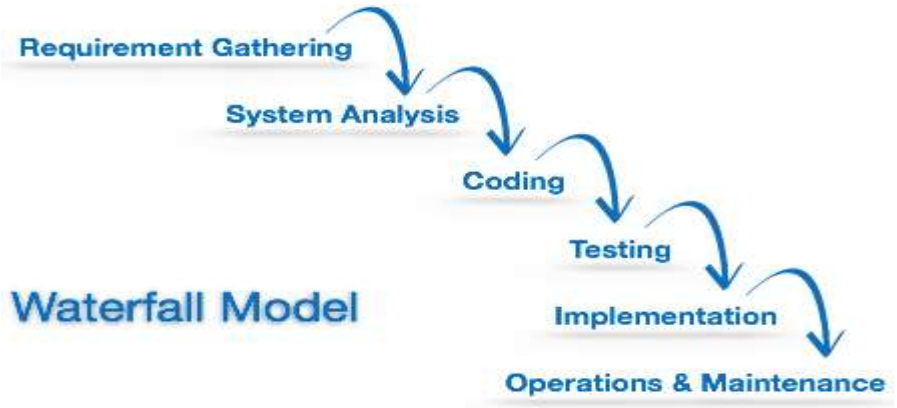
- **Coding:** This step is also known as programming phase. The implementation of software design starts in terms of writing program code in the suitable

programming language and developing error-free executable programs efficiently. This activity is the transition of the design specification into the software program.

- **Testing:** It is the process of executing a program or application with the intent of finding the software bugs.

- **Maintenance:** It is the modification of a software product after delivery to correct faults, to improve performance or other attributes. A common perception of maintenance is that it merely involves fixing defects.

The life-cycle paradigm of software engineering sometimes referred as "Water fall model". Waterfall model is the simplest model of software development paradigm. It says the all the phases of SDLC will function one after another in linear manner. That is, when the first phase is finished then only the second phase will start and so on.



This model assumes that everything is carried out and taken place perfectly as planned in the previous stage and there is no need to think about the past issues that may arise in the next phase. This model does not work smoothly if there are some issues left at the previous step. The sequential nature of model does not allow us go back and undo or redo our actions.

25. a. Explain the open standard principles.

An Open Standard is more than just a specification. An open standard is a standard that is publicly available and has various rights to use associated with it, and may also have various properties of how it was designed.

The Open Source Principles are:

1. Availability:

Open Standards are available for all to read and implement.

- The best practice is for the standards text and reference implementation to be available for free downloads via the Internet. Thus

- Any software project should be able to copy without any expense.

- Licenses attached to the standards documentation must not restrict any party from implementing the standard using any form of software license.

- The best practice is for the software reference platforms to be licensed in a way that is compatible with all forms of software licensing, both free and proprietary software.

2. Maximize End-User Choice:

Open Standards create a fair, competitive market for implementations of the standard. They do not lock the customer in to a particular vendor or group. Thus

- They must allow wide range of implementations, by businesses, academic and public projects.

- They must support a range of pricing from very expensive to zero price.

3. No Royalty:

Open standards are free for all to implement, with no royalty or fee. Certification of agreement by the standards organization must have a fee. Thus

- Patents embedded in standards must be licensed royalty-free.

- Certification Programs should include a low to zero cost self- certification.

4. No Discrimination:

Open Standard and the organizations that administer them do not favor one implementer over another for any reason other than the technical standards of a vendor's implementation.

- A standard organization that wishes to support itself through certification branding should establish a premium track and a low-cost to zero-cost track.

5. Extension or Subset

Implementations of Open Standards may be extended, or offered in subset form. However, certificate organizations may decline to certify subset implementations, and may place requirements upon extensions.

6. Predatory Practices:

Open Standards may employ license terms that protect against subversion of the standard by embrace-and-extend tactics. The license may require the publication of reference information and license to create and redistribute software compatible with the extensions. It may not prohibit the implementations of extensions.

- The standards organization may wish to apply an agreement similar to the Sun Industry Standards Source License to the standard documentation and its reference implementation. The Sun agreement requires publication of a reference implementation for any extensions to the standard. This makes it possible for a standards organization to actively preserve interoperability.

7. Modularity:

Modularity is the principle of maintain well defined boundaries between components. Adhering to open standards it is a noticeable principle in open source software because it helps to define the boundaries.

8. Early and Often:

Early and often lead to making things available as early as possible. This includes not only the software but also plans and design. Many open source project achieve the early and often using a couple of the following common methods.

- **Source Code Availability:**

  The source code of the software is made available either on a daily as a zip file or lives in a source code repository such as CVS or Subversions. Community members using these systems have to compile or build the software from its source code. This used by a small percentage of the community.

- **Milestones builds:**

  On a periodic basis the administrators or developers of the project compile the software and make resulting executables for downloading. These downloads are used by the members of community that are interested in the functionality of the software. Many open source projects release milestones builds.

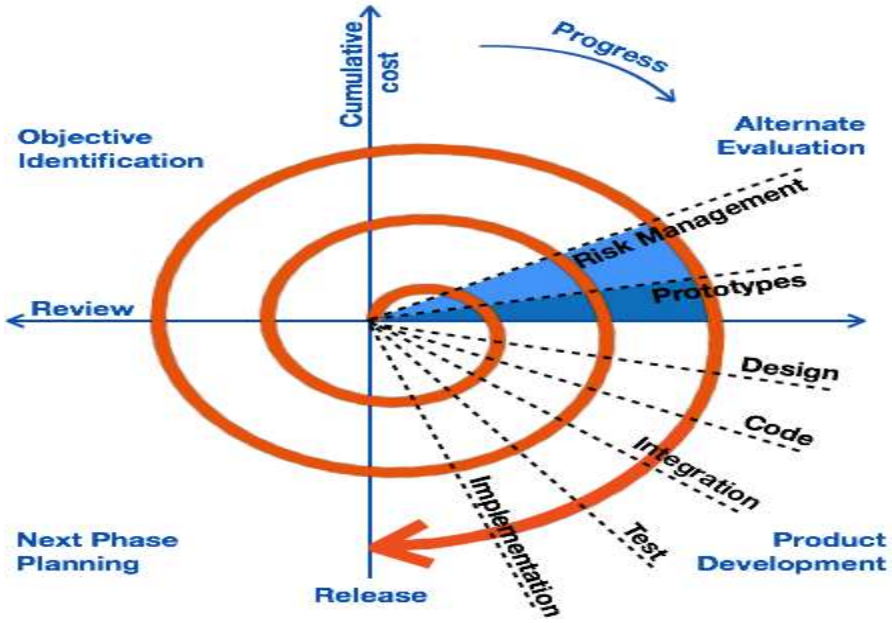
- **Nightly builds:**

On a nightly basis the source code is compiled to generate the executable binaries. These builds are made available for community members want to test the latest source code.

Or

b. What is spiral model in SDLC? Explain in detail.

Spiral model is a combination of both, prototyping model and one of the SDLC model. The spiral model adds an element of risk analysis to the development process.



This model is presented as a spiral. There are four major activities represents each iteration:

- **Planning:** Determine objectives and constraints of the project, and define the alternatives.
- **Risk Analysis:** Analysis of alternatives, and identification and resolution of risks.
- **Engineering:** Development of the next-level product.
- **Customer Evolution:** Evaluation of the product engineered.

25. a. What is ethics? Explain the different types of ethics.

26. The Ethics of Open Source:

27.

28. Ethics denotes the theory of right action and greater good. The field of ethics (or moral philosophy) involves systematizing, defending, and recommending concepts of right and wrong behavior. Moral has a dual meaning

29. 1. It indicates a person's comprehension of morality and his or her ability to put it into practice

30. 2. It denotes the limited specific acts and defined moral codes.

31. Types of Ethics:

32. **Personal Ethics:**

33. It refers to a person's personal or self-created values and codes of conduct. From the very beginning, these ethics are instilled in an individual, with a large part having been played by their parents, friends, and family.

34. Descriptive Ethics:

35. Descriptive ethics are the morals of a society. People use descriptive ethics as a way to judge particular actions as good or bad based on the social contract of a particular society. It is possible for people in one group to hold a different set of morals than people in another group. Descriptive ethics also change over time. Those working on descriptive ethics aim to uncover people's beliefs about such things as values, which actions are right and wrong, and which characteristics of moral agents are virtuous.
36. **Meta Ethics**
37. Meta ethics is a branch of analytic philosophy that explores the status, foundations, and scope of moral values, properties, and words. Whereas the fields of applied ethics and normative theory focus on what is moral, meta ethics focuses on what morality itself is.
38. **Normative Ethics:**
39. Normative ethics is the study of ethical action. It is the branch of philosophical ethics that investigates the set of questions that arise when considering how one ought to act, morally speaking.
40. **Applied Ethics:**
41. Applied ethics is the philosophical examination, from a moral standpoint, of particular issues in private and public life which are matters of moral judgment. It is thus the attempts to use philosophical methods to identify the morally correct course of action in various fields of everyday life.

Or

b. Explain the open source hardware.

"Open source hardware," or "open hardware," refers to the design specifications of a physical object which are licensed in such a way that said object can be studied, modified, created, and distributed by anyone.

"Open hardware" is a set of design principles and legal practices, not a specific type of object. The term can therefore refer to any number of objects—like automobiles, chairs, computers, robots, or even houses.

Like open source software, the "source code" for open hardware—schematics, blueprints, logic designs, Computer Aided Design (CAD) drawings or files, etc.—is available for modification or enhancement by anyone under permissive licenses. Users with access to the tools that can read and manipulate these source files can update and improve the code that underlies the physical device. They can add features or fix bugs in the software. They can even modify the physical design of the object itself and, if they wish, proceed to share such modifications.

Open hardware's source code should be readily accessible, and its components are preferably easy for anyone to obtain. Essentially, open hardware eliminates common roadblocks to the design and manufacture of physical goods; it provides as many people

as possible the ability to construct, remix, and share their knowledge of hardware design and function.

Open source hardware licenses generally permit recipients of the designs and documentations to study them, redistribute and modify them, and then to distribute any modifications. Additionally, open hardware licenses don't prevent someone from giving away or even selling the project's documentation.

Like software, hardware designs and inventions are subject to copyright and patent law. And like open source software, open source hardware uses these intellectual property laws creatively to make hardware designs publicly accessible.

Both copyright law (in the case of source code and design documentation) and patent law (in the case of design processes and material technologies) apply to open hardware. Trademark law is also pertinent to the branding names and logos of open hardware.