Darey.io Project 1

**Self Study Tasks**

1. Conduct a Google search on what software development life cycle (SDLC) is and document your finding in a Google word file.

From: <https://en.wikipedia.org/wiki/Systems_development_life_cycle>

A systems development life cycle is composed of a number of clearly defined and distinct work phases which are used by systems engineers and systems developers to plan for, design, build, test, and deliver [information systems](https://en.wikipedia.org/wiki/Information_system). Like anything that is manufactured on an assembly line, an SDLC aims to produce high-quality systems that meet or exceed customer expectations, based on customer requirements, by delivering systems which move through each clearly defined phase, within scheduled time frames and cost estimates.

Computer systems are complex and often (especially with the recent rise of [service-oriented architecture](https://en.wikipedia.org/wiki/Service-oriented_architecture)) link multiple traditional systems potentially supplied by different software vendors. To manage this level of complexity, a number of SDLC models or methodologies have been created, such as [waterfall](https://en.wikipedia.org/wiki/Waterfall_model), [spiral](https://en.wikipedia.org/wiki/Spiral_model), [Agile software development](https://en.wikipedia.org/wiki/Agile_software_development), [rapid prototyping](https://en.wikipedia.org/wiki/Software_prototyping#Throwaway_prototyping), [incremental](https://en.wikipedia.org/wiki/Incremental_development), and synchronize and stabilize.

From: <https://www.tutorialspoint.com/sdlc/sdlc_overview.htm>

**What is SDLC?**

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

The following figure is a graphical representation of the various stages of a typical SDLC.



A typical Software Development Life Cycle consists of the following stages −

**Stage 1: Planning and Requirement Analysis**

Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational and technical areas.

Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage. The outcome of the technical feasibility study is to define the various technical approaches that can be followed to implement the project successfully with minimum risks.

**Stage 2: Defining Requirements**

Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts. This is done through an **SRS (Software Requirement Specification)** document which consists of all the product requirements to be designed and developed during the project life cycle.

**Stage 3: Designing the Product Architecture**

SRS is the reference for product architects to come out with the best architecture for the product to be developed. Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification.

This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity, budget and time constraints, the best design approach is selected for the product.

A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation with the external and third party modules (if any). The internal design of all the modules of the proposed architecture should be clearly defined with the minutest of the details in DDS.

**Stage 4: Building or Developing the Product**

In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle.

Developers must follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers, etc. are used to generate the code. Different high level programming languages such as C, C++, Pascal, Java and PHP are used for coding. The programming language is chosen with respect to the type of software being developed.

**Stage 5: Testing the Product**

This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However, this stage refers to the testing only stage of the product where product defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

**Stage 6: Deployment in the Market and Maintenance**

Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometimes product deployment happens in stages as per the business strategy of that organization. The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing).

Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment. After the product is released in the market, its maintenance is done for the existing customer base.

**SDLC Models**

There are various software development life cycle models defined and designed which are followed during the software development process. These models are also referred as Software Development Process Models". Each process model follows a Series of steps unique to its type to ensure success in the process of software development.

Following are the most important and popular SDLC models followed in the industry −

Waterfall Model, Iterative Model, Spiral Model, V-Model, Big Bang Model

Other related methodologies are Agile Model, RAD Model, Rapid Application Development and Prototyping Models.

1. Conduct another Google search, understand what LAMP stack means.

From: <https://aws.amazon.com/what-is/lamp-stack/>

## What is a LAMP stack?

A LAMP stack is a bundle of four different software technologies that developers use to build websites and web applications. LAMP is an acronym for the operating system, Linux; the web server, Apache; the database server, MySQL; and the programming language, PHP. All four of these technologies are open source, which means they are community maintained and freely available for anyone to use. Developers use LAMP stacks to create, host, and maintain web content. It is a popular solution that powers many of the websites you commonly use today.

## Why is a LAMP stack important?

Web developers choose a LAMP stack to build web applications for the following reasons.

### ****Cost****

All LAMP technologies are open source, which means that any developer or company can use them without having to pay licensing fees. Instead of purchasing proprietary stack components, you can download the operating system, web server, database, and scripting language for free. This lowers the cost of building web applications.

### ****Efficiency****

Setting up a new web development stack requires rigorous testing of different frameworks, modules, libraries, and tools. On the other hand, a LAMP stack is a tried-and-tested web development solution. Web developers can prioritize and speed up application development to focus on what they are building instead of how they are building it.

### ****Maintenance****

Software experts from around the globe contribute to the development of LAMP stack technologies by changing, commenting on, and reviewing the publicly available source codes. They regularly maintain and update the technologies so that they remain relevant and secure.

### ****Support****

Popular open-source technologies, such as LAMP stacks, have the support of a large, global IT community. Hence, LAMP stack users can more easily find information on public IT forums. Web developers can refer to example codes or use tested plugins created by the open-source community.

### ****Flexibility****

A LAMP stack gives both reliability and flexibility to web developers. While the LAMP architecture specifies the software components for each layer, developers can replace them as they see fit. For example, they can use another operating system besides Linux as the stack foundation.

## What is a LAMP stack used for?

A LAMP stack is used for backend or server-side development. A backend application is software that runs in an environment that’s hidden from end users. Backend applications consist of the following:

* Data processing software
* Database components
* Business logic in code
* API for communicating with other applications

The webpage that shows up on your browser is called the frontend application. When you interact with the page, such as by clicking on a button, your browser communicates with the backend application to retrieve the required information.

Developers use a LAMP stack to create both static and dynamic web content.

### ****Static webpages****

Static webpage information from the web server is the same for all users. For example, the address on a company's website is static content. Web developers create static webpages with HTML and CSS programming languages and store them as files in the web server application.

### ****Dynamic webpages****

Dynamic webpages contain information that changes depending on the user viewing the webpage or web application. For example, a website message that changes based on your location is dynamic content. The web server delivers dynamic websites by processing business logic or retrieving data from a database.

Graphical user interface, text

Description automatically generated with medium confidence

## What is the LAMP architecture?

A software stack is a set of layered tools, libraries, programming languages, and technologies used for building, managing, and running an application. The stack consists of software components that support the application in different ways, such as visual presentation, database, networking, and security.

Similarly, the LAMP architecture consists of four software technologies that work together behind the scenes to create a working web application. It describes how each of these web development technologies interact with each other in a computer server. The LAMP architecture consists of the following layers.

### ****Linux****

Linux is an open-source operating system that you can install and configure to meet different application requirements. Linux sits at the first level of the LAMP stack and supports other components on the upper layers.

### ****Apache****

Apache is an open-source web server that forms the second layer of the LAMP stack. The Apache module stores website files and exchanges information with a browser using HTTP, an internet protocol for transferring website information in plain text. For example, when a browser requests a webpage, the Apache HTTP server does the following:

1. Receives the request
2. Processes the request and finds the required page file
3. Sends the relevant information back to the browser

### ****MySQL****

MySQL is an open-source relational database management system and is the third layer of the LAMP stack. The LAMP model uses MySQL for storing, managing, and querying information in relational databases. For example, developers store application data, such as customer records, sales, and inventories. When a user searches for information, the web server queries the stored data in MySQL. Query refers to special instructions for manipulating data in a relational database with the SQL language.

### ****PHP****

PHP, which stands for PHP: Hypertext Preprocessor, is the fourth and final layer of the LAMP stack. It is a scripting language that allows websites to run dynamic processes. A dynamic process involves information in software that constantly changes. Web developers embed the PHP programming language in HTML to show real-time or updated information on websites. They use PHP to allow the web server, database, and operating system to cohesively process requests from browsers.

#### **HTML compared to PHP**

Web developers use HTML for frontend development, such as designing the layout of webpages. Meanwhile, they use PHP to determine the behavior of certain components when users load a webpage. For example, web developers design the graphical layout of an online product catalog with HTML. They then use PHP code to retrieve the latest product price from the backend server.

## How does a LAMP stack work?

Web applications use a LAMP stack to respond to requests from web browsers. The Apache web server and MySQL database run on the Linux operating system and communicate using PHP. When you open a webpage in a browser, the LAMP stack goes through the following process.

### ****Receives requests****

The Apache web server receives the incoming request from the browser. If the request is to load a static file, the Apache server responds directly with the appropriate content. If the request is for dynamic content, the Apache server passes the request to the PHP component. The PHP component finds and loads the appropriate PHP file that can process the request.

### ****Processes requests****

The PHP file contains PHP functions that are codes for generating dynamic content. The PHP component processes the PHP functions, such as converting measurement units or creating a sales chart. Some PHP functions might require information from the database. In such cases, the PHP code retrieves the stored information from the database and uses it to process the function.

### ****Returns responses****

The PHP passes the calculated results to the web server in HTML format. At the same time, it also stores new data in the MySQL database. The Apache HTTP server sends the dynamic HTML results to the user's browser.

1. Read about **‘chmod’** and **‘chown’** commands in Linux and understand how access and ownership of files and directories work.

From: <http://linuxcommand.org/lc3_lts0090.php>

# Permissions

The Unix-like operating systems, such as Linux differ from other computing systems in that they are not only *multitasking* but also *multi-user*.

What exactly does this mean? It means that more than one user can be operating the computer at the same time. While a desktop or laptop computer only has one keyboard and monitor, it can still be used by more than one user. For example, if the computer is attached to a network, or the Internet, remote users can log in via [ssh](http://linuxcommand.org/lc3_man_pages/ssh1.html) (secure shell) and operate the computer. In fact, remote users can execute graphical applications and have the output displayed on a remote computer. The X Window system supports this.

The multi-user capability of Unix-like systems is a feature that is deeply ingrained into the design of the operating system. If we remember the environment in which Unix was created, this makes perfect sense. Years ago before computers were "personal," they were large, expensive, and centralized. A typical university computer system consisted of a large mainframe computer located in some building on campus and *terminals* were located throughout the campus, each connected to the large central computer. The computer would support many users at the same time.

In order to make this practical, a method had to be devised to protect the users from each other. After all, we wouldn't want the actions of one user to crash the computer, nor would we allow one user to interfere with the files belonging to another user.

This lesson will cover the following commands:

* [chmod](http://linuxcommand.org/lc3_man_pages/chmod1.html) - modify file access rights
* [su](http://linuxcommand.org/lc3_man_pages/su1.html) - temporarily become the superuser
* [sudo](http://linuxcommand.org/lc3_man_pages/sudo8.html) - temporarily become the superuser
* [chown](http://linuxcommand.org/lc3_man_pages/chown1.html) - change file ownership
* [chgrp](http://linuxcommand.org/lc3_man_pages/chgrp1.html) - change a file's group ownership

## File Permissions

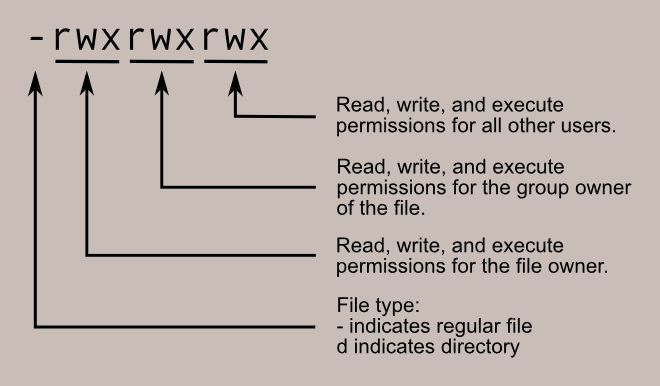
On a Linux system, each file and directory is assigned access rights for the owner of the file, the members of a group of related users, and everybody else. Rights can be assigned to read a file, to write a file, and to execute a file (i.e., run the file as a program).

To see the permission settings for a file, we can use the ls command. As an example, we will look at the bash program which is located in the /bin directory:

[me@linuxbox me]$ ls -l /bin/bash -rwxr-xr-x 1 root root 1113504 Jun 6 2019 /bin/bash

Here we can see:

* The file "/bin/bash" is owned by user "root"
* The superuser has the right to read, write, and execute this file
* The file is owned by the group "root"
* Members of the group "root" can also read and execute this file
* Everybody else can read and execute this file

In the diagram below, we see how the first portion of the listing is interpreted. It consists of a character indicating the file type, followed by three sets of three characters that convey the reading, writing and execution permission for the owner, group, and everybody else.  
  


## chmod

The chmod command is used to change the permissions of a file or directory. To use it, we specify the desired permission settings and the file or files that we wish to modify. There are two ways to specify the permissions. In this lesson we will focus on one of these, called the *octal notation* method.

It is easy to think of the permission settings as a series of bits (which is how the computer thinks about them). Here's how it works:

rwx rwx rwx = 111 111 111

rw- rw- rw- = 110 110 110

rwx --- --- = 111 000 000

and so on...

rwx = 111 in binary = 7

rw- = 110 in binary = 6

r-x = 101 in binary = 5

r-- = 100 in binary = 4

Now, if we represent each of the three sets of permissions (owner, group, and other) as a single digit, we have a pretty convenient way of expressing the possible permissions settings. For example, if we wanted to set some\_file to have read and write permission for the owner, but wanted to keep the file private from others, we would:

[me@linuxbox me]$ chmod 600 some\_file

Here is a table of numbers that covers all the common settings. The ones beginning with "7" are used with programs (since they enable execution) and the rest are for other kinds of files.

|  |  |
| --- | --- |
| **Value** | **Meaning** |
| **777** | **(rwxrwxrwx)** No restrictions on permissions. Anybody may do anything. Generally not a desirable setting. |
| **755** | **(rwxr-xr-x)** The file's owner may read, write, and execute the file. All others may read and execute the file. This setting is common for programs that are used by all users. |
| **700** | **(rwx------)** The file's owner may read, write, and execute the file. Nobody else has any rights. This setting is useful for programs that only the owner may use and must be kept private from others. |
| **666** | **(rw-rw-rw-)** All users may read and write the file. |
| **644** | **(rw-r--r--)** The owner may read and write a file, while all others may only read the file. A common setting for data files that everybody may read, but only the owner may change. |
| **600** | **(rw-------)** The owner may read and write a file. All others have no rights. A common setting for data files that the owner wants to keep private. |

## Directory Permissions

The chmod command can also be used to control the access permissions for directories. Again, we can use the octal notation to set permissions, but the meaning of the r, w, and x attributes is different:

* **r** - Allows the contents of the directory to be listed if the x attribute is also set.
* **w** - Allows files within the directory to be created, deleted, or renamed if the x attribute is also set.
* **x** - Allows a directory to be entered (i.e. cd dir).

Here are some useful settings for directories:

|  |  |
| --- | --- |
| **Value** | **Meaning** |
| **777** | **(rwxrwxrwx)** No restrictions on permissions. Anybody may list files, create new files in the directory and delete files in the directory. Generally not a good setting. |
| **755** | **(rwxr-xr-x)** The directory owner has full access. All others may list the directory, but cannot create files nor delete them. This setting is common for directories that you wish to share with other users. |
| **700** | **(rwx------)** The directory owner has full access. Nobody else has any rights. This setting is useful for directories that only the owner may use and must be kept private from others. |

## Becoming the Superuser for a Short While

It is often necessary to become the superuser to perform important system administration tasks, but as we know, we should not stay logged in as the superuser. In most distributions, there is a program that can give you temporary access to the superuser's privileges. This program is called su (short for substitute user) and can be used in those cases when you need to be the superuser for a small number of tasks. To become the superuser, simply type the su command. You will be prompted for the superuser's password:

[me@linuxbox me]$ su Password: [root@linuxbox me]#

After executing the su command, we have a new shell session as the superuser. To exit the superuser session, type exit and we will return to your previous session.

In most modern distributions, an alternate method is used. Rather than using su, these systems employ the sudo command instead. With sudo, one or more users are granted superuser privileges on an as needed basis. To execute a command as the superuser, the desired command is simply preceded with the sudo command. After the command is entered, the user is prompted for the their own password rather than the superuser's:

[me@linuxbox me]$ sudo some\_command Password for me: [me@linuxbox me]$

In fact, modern distributions don't even set the root account password thus making it impossible to log in as the root user. A root shell is still possible with sudo by using the "-i" option:

[me@linuxbox me]$ sudo -i Password for me: root@linuxbox:~#

## Changing File Ownership

We can change the owner of a file by using the chown command. Here's an example: Suppose we wanted to change the owner of some\_file from "me" to "you". We could:

[me@linuxbox me]$ sudo chown you some\_file

Notice that in order to change the owner of a file, we must have superuser privileges. To do this, our example employed the sudo command to execute chown.

chown works the same way on directories as it does on files.

## Changing Group Ownership

The group ownership of a file or directory may be changed with chgrp. This command is used like this:

[me@linuxbox me]$ chgrp new\_group some\_file

In the example above, we changed the group ownership of some\_file from its previous group to "new\_group". We must be the owner of the file or directory to perform a chgrp.

1. Learn what TCP and UPD terms mean and how they are different. List down ports most commonly used in Web (http, https, ssh, telnet, ftp, sftp, telnet)

From: <https://www.guru99.com/tcp-vs-udp-understanding-the-difference.html>

## What is TCP?

[TCP/IP](https://www.guru99.com/tcp-ip-model.html) helps you to determine how a specific computer should be connected to the internet and how you can transmit data between them. It helps you to create a virtual network when multiple computer networks are connected.

TCP/IP stands for Transmission Control Protocol/ Internet Protocol. It is specifically designed as a model to offer highly reliable and end-to-end byte stream over an unreliable internetwork.

## What is UDP?

UDP is a Datagram oriented protocol. It is used for broadcast and multicast type of network transmission. The full form of UDP is User Datagram Protocol (A datagram is a transfer unit associated with a packet-switched network.) The UDP protocol works almost similar to TCP, but it throws all the error-checking stuff out, all the back-and-forth communication and deliverability.

## KEY DIFFERENCES:

* TCP is a connection-oriented protocol, whereas UDP is a connectionless protocol.
* The speed for TCP is slower while the speed of UDP is faster
* TCP uses handshake protocol like SYN, SYN-ACK, ACK while UDP uses no handshake protocols
* TCP does error checking and also makes error recovery, on the other hand, UDP performs error checking, but it discards erroneous packets.
* TCP has acknowledgment segments, but UDP does not have any acknowledgment segment.
* When we compare TCP vs UDP protocol, TCP is heavy-weight, and UDP is lightweight.

## How TCP work?

A TCP connection is established with the help of [three-way handshake](https://www.guru99.com/tcp-3-way-handshake.html). It is a process of initiating and acknowledging a connection. Once the connection is established, data transfer begins, and when the transmission process is finished, the connection is terminated by the closing of an established virtual circuit.

## How UDP work?

UDP uses a simple transmission method without implied hand-shaking dialogues for ordering, reliability, or data integrity. UDP also assumes that error checking and correction is not important or performed in the application, to avoid the overhead of such processing at the network interface level. It is also compatible with packet broadcasts and multicasting.

## Features of TCP

Here are some important features of TCP:

* Delivery Acknowledgements
* Re transmission
* Delays transmission when the network is congested
* Easy Error detection

Here are some important features of UDP:

* Supports bandwidth-intensive applications that tolerate packet loss
* Less delay
* It sends the bulk quantity of packets.
* Possibility of the Data loss
* Allows small transaction ( DNS lookup)

Differences between TCP and UDP

| **Basis** | **Transmission control protocol (TCP)** | **User datagram protocol (UDP)** |
| --- | --- | --- |
| **Type of Service** | TCP is a connection-oriented protocol. Connection-orientation means that the communicating devices should establish a connection before transmitting data and should close the connection after transmitting the data. | UDP is the Datagram-oriented protocol. This is because there is no overhead for opening a connection, maintaining a connection, and terminating a connection. UDP is efficient for broadcast and multicast types of network transmission. |
| **Reliability** | TCP is reliable as it guarantees the delivery of data to the destination router. | The delivery of data to the destination cannot be guaranteed in UDP. |
| **Error checking mechanism** | TCP provides extensive error-checking mechanisms. It is because it provides flow control and acknowledgment of data. | UDP has only the basic error checking mechanism using checksums. |
| **Acknowledgment** | An acknowledgment segment is present. | No acknowledgment segment. |
| **Sequence** | Sequencing of data is a feature of Transmission Control Protocol (TCP). this means that packets arrive in order at the receiver. | There is no sequencing of data in UDP. If the order is required, it has to be managed by the application layer. |
| **Speed** | TCP is comparatively slower than UDP. | UDP is faster, simpler, and more efficient than TCP. |
| **Retransmission** | Retransmission of lost packets is possible in TCP, but not in UDP. | There is no retransmission of lost packets in the User Datagram Protocol (UDP). |
| **Header Length** | TCP has a (20-60) bytes variable length header. | UDP has an 8 bytes fixed-length header. |
| **Weight** | TCP is heavy-weight. | UDP is lightweight. |
| **Handshaking Techniques** | Uses handshakes such as SYN, ACK, SYN-ACK | It’s a connectionless protocol i.e. No handshake |
| **Broadcasting** | TCP doesn’t support Broadcasting. | UDP supports Broadcasting. |
| **Protocols** | TCP is used by HTTP, HTTPs, FTP, SMTP and Telnet. | UDP is used by DNS, DHCP, TFTP, SNMP, RIP, and VoIP. |
| **Stream Type** | The TCP connection is a byte stream. | UDP connection is message stream. |
| **Overhead** | Low but higher than UDP. | Very low. |

1. Get yourself familiar with basic text editing in [Vi (Vim)](https://www.vim.org) editor. [Practice here](https://www.openvim.com).