TPCT's

College of Engineering, Osmanabad

Laboratory Manual

OPEN SOURCE LABORATORY

For

Second Year Students

Manual Prepared by

Prof. P.M.Pawar

Author COE, Osmanabad

TPCT's

College of Engineering
Solapur Road, Osmanabad
Department of Computer Science & Engineering

Vision of the Department:

To achieve and evolve as a center of academic excellence and research center in the field of Computer Science and Engineering. To develop computer engineers with necessary analytical ability and human values who can creatively design, implement a wide spectrum of Computer systems for welfare of the society.

Mission of the Department:

The department strives to continuously engage in providing the students with in-depth understanding of fundamentals and practical training related to professional skills and their applications through effective Teacher-Learning Process and state of the art laboratories pertaining to CSE and interdisciplinary areas. Preparing stude nts in developing research, design, entrepreneurial skills and employability capabilities.

College of Engineering

Technical Document

This technical document is a series of Laboratory manuals of Computer Science & Engineering Department and is a certified document of College of Engineering, Osmanabad. The care has been taken to make the document error-free. But still if any error is found. Kindly bring it to the notice of subject teacher and HOD.

Recommended by,

HOD

Approved by,

Principal

Copies:

1. Departmental Library
2. Laboratory
3. HOD
4. Principal

FOREWORD

It is my great pleasure to present this laboratory manual for second year engineering students for the subject of Open Source Laboratory keeping in view the vast coverage required for visualization of concepts of Signals and Systems.

As a student, many of you may be wondering with some of the questions in your mind regarding the subject and exactly what has been tried is to answer through this manual.

Faculty members are also advised that covering these aspects in initial stage itself, will greatly relived them in future as much of the load will be taken care by the enthusiasm energies of the students once they are conceptually clear.

H.O.D.

LABORATORY MANUAL CONTENTS

This manual is intended for the Second year students of engineering branches in the subject of Open Source Laboratory. This manual typically contains practical/Lab Sessions related Open Source Laboratory covering various aspects related to the subject to enhance understanding.

Students are advised to thoroughly go through this manual rather than only topics mentioned in the syllabus as practical aspects are the key to understanding and conceptual visualization of theoretical aspects covered in the books.

Prof.P.M.Pawar

SUBJECT INDEX

1. Do's and Don'ts in the laboratory
2. Lab Experiments:
    - Demonstration of Open source software's installation i.e. xampp, lamp, etc.
    - Design HTML form and retrieve the values in PHP script.
    - PHP variables, arrays (array multiplication, addition ...etc).
    - PHP Functions: array, string, date-time, and calendar.
    - MySQLi connectivity, INSERT, SELECT, DELETE with PHP.
    - PHP Mysqli connectivity using OOP method.
    - PHP script for File uploading.
    - PHP script for-Session Management (login form).
    - AJAX Script using XMLHttpRequest, Data Formats, PHP.
    - PHP script to update and retrieve data stored in database from user using Ajax.
3. Quiz on the subject
4. Conduction of Viva-Voce Examinations
5. Evaluation and Marking System

Dos and Don'ts in Laboratory :-

1. Maintain Punctuality of time for lab and also for works and assignment completion.
2. Make entry in the Log Book as soon as you enter the Laboratory.
3. All the students should sit according to their roll numbers starting from their left to right.
4. All the students are supposed to enter the terminal number in the log book.
5. Do not change the terminal on which you are working.
6. All the students are expected to get at least the algorithm of the program/concept to be implemented.
7. Strictly follow the instructions given by the teacher/Lab Instructor.
8. Handle Equipments with care.
9. Turn off the machine once you are done using it.
10. Do not Install/Remove any software on system without permission.
11. Do not open any irrelevant Internet Sites on lab computer.
12. Do not plug in external devices without scanning them for computer viruses.

Instructions for Laboratory Teachers:-

1. Submission related to whatever lab work has been completed should be done during the next lab session.
2. The immediate arrangements for printouts related to submission on the day of practical assignments.
3. Students should be taught for taking the printouts under the observation of lab teacher.
4. The promptness of submission should be encouraged by way of marking and evaluation patterns that will benefit the sincere students.

EXPERIMENT NO. 1

AIM: Demonstration of Open source software's installation i.e. lamp.

OBJECTIVE: To learn installation of Linux, Apache, PHP and Mysql.

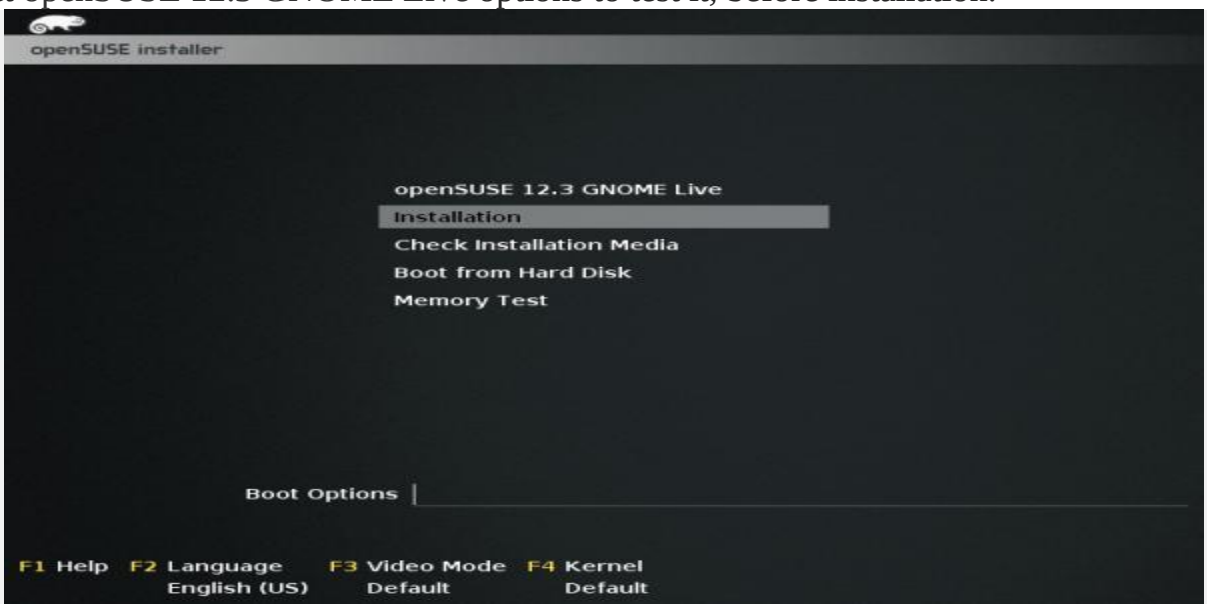
TOOLS REQUIRED: CD or image of Linux Operating System

THEORY:

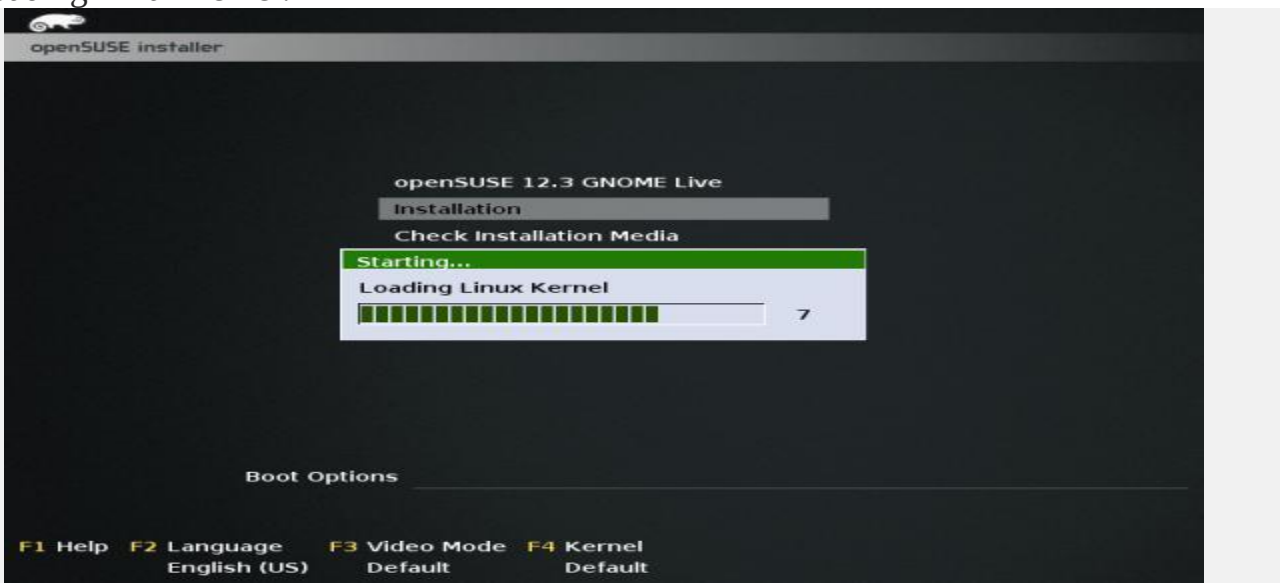
LAMP stack is a group of open source software used to get web servers up and running. The acronym stands for Linux, Apache, MySQL, and PHP.

1. Boot your system with OpenSUSE 12.3 installation media i.e CD/DVD or ISO image.

2. Choose installation options to install openSUSE 12.3 on you system. Please select openSUSE 12.3 GNOME Live options to test it, before installation.

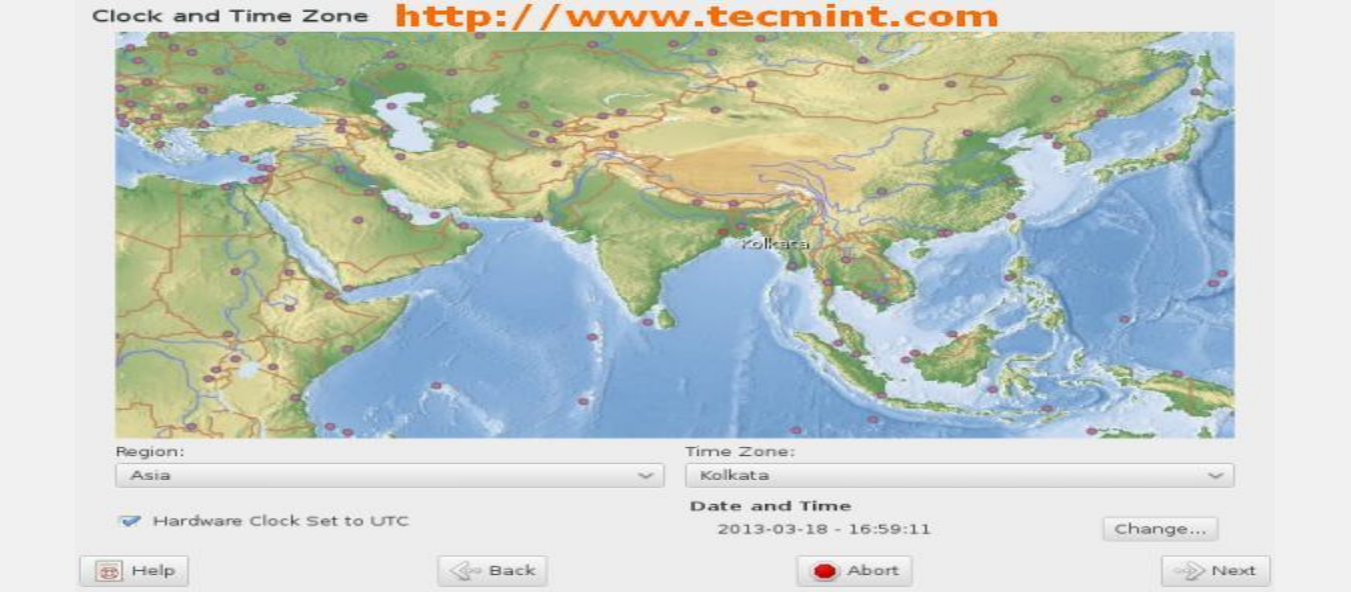


3. Loading Linux kernel.



4. Welcome screen, From where we can select Language and keyboard layout. Read license agreement and proceed further installation once agreed.

5. Clock and timezone settings.



6. Please click on change if you want custom setting of date and time. You can change it manually or sync with NTP Server as show below. Click Accept once done.

Set Date and Time

7. File system partitioning. We opted default filesystem partition. You may choose manual filesystem partitioning as options provided.

OpenSuse File System Partitioning

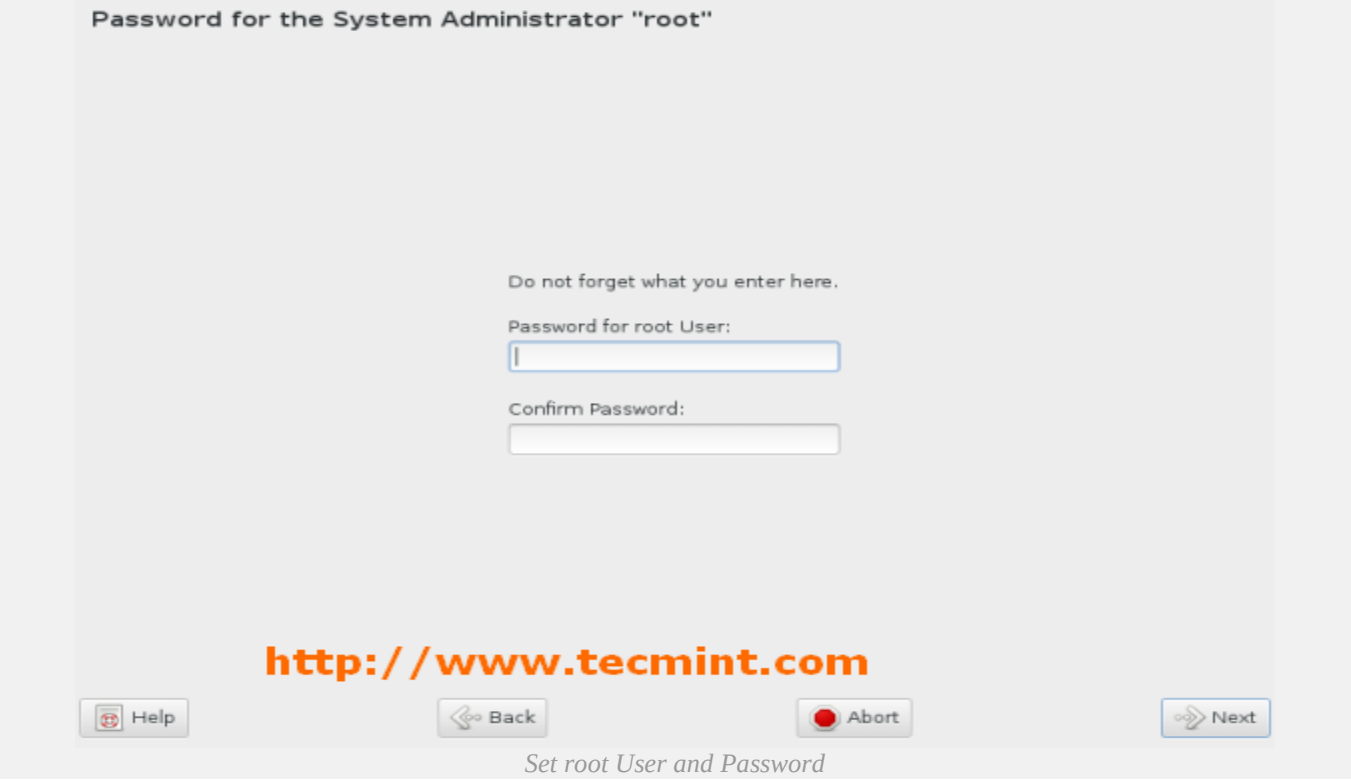
8. Create new user and it's password. Uncheck all three options. Click on change to select authentication method.

Create User and Password

9. Please select authentication method and click on Accept.

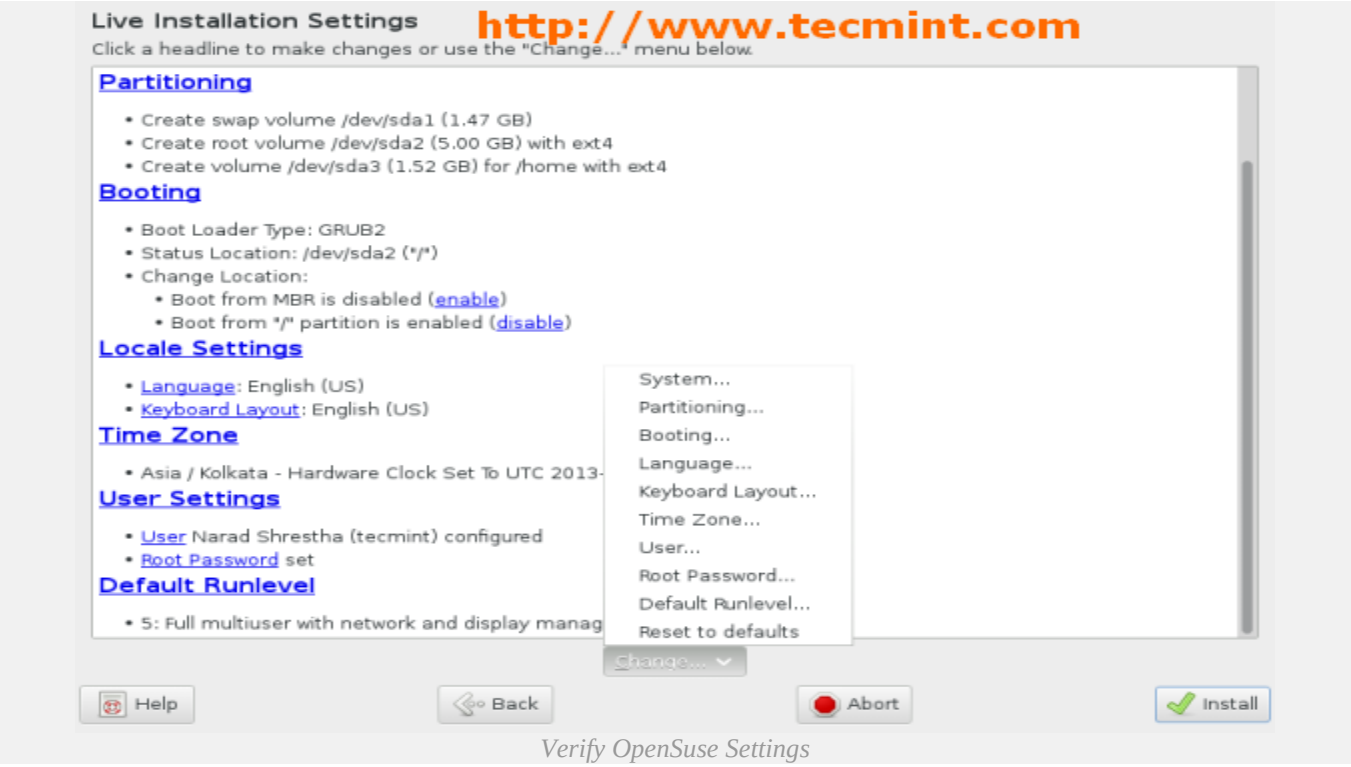
Select Authentication Method

10. Set root user password and click on Next.



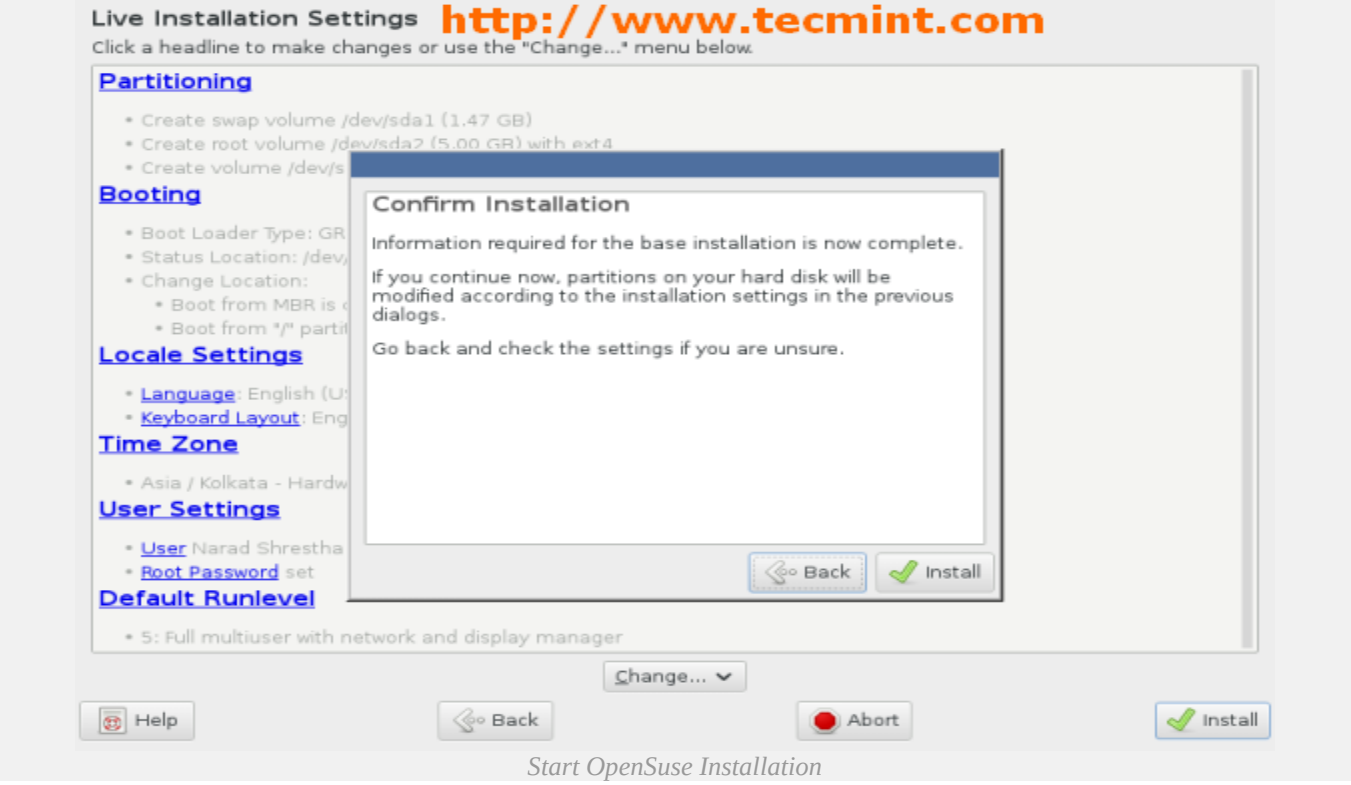
Set root User and Password

11. Verify settings, you may change settings after clicking on headlines or click on Change button. Once done click on Install.



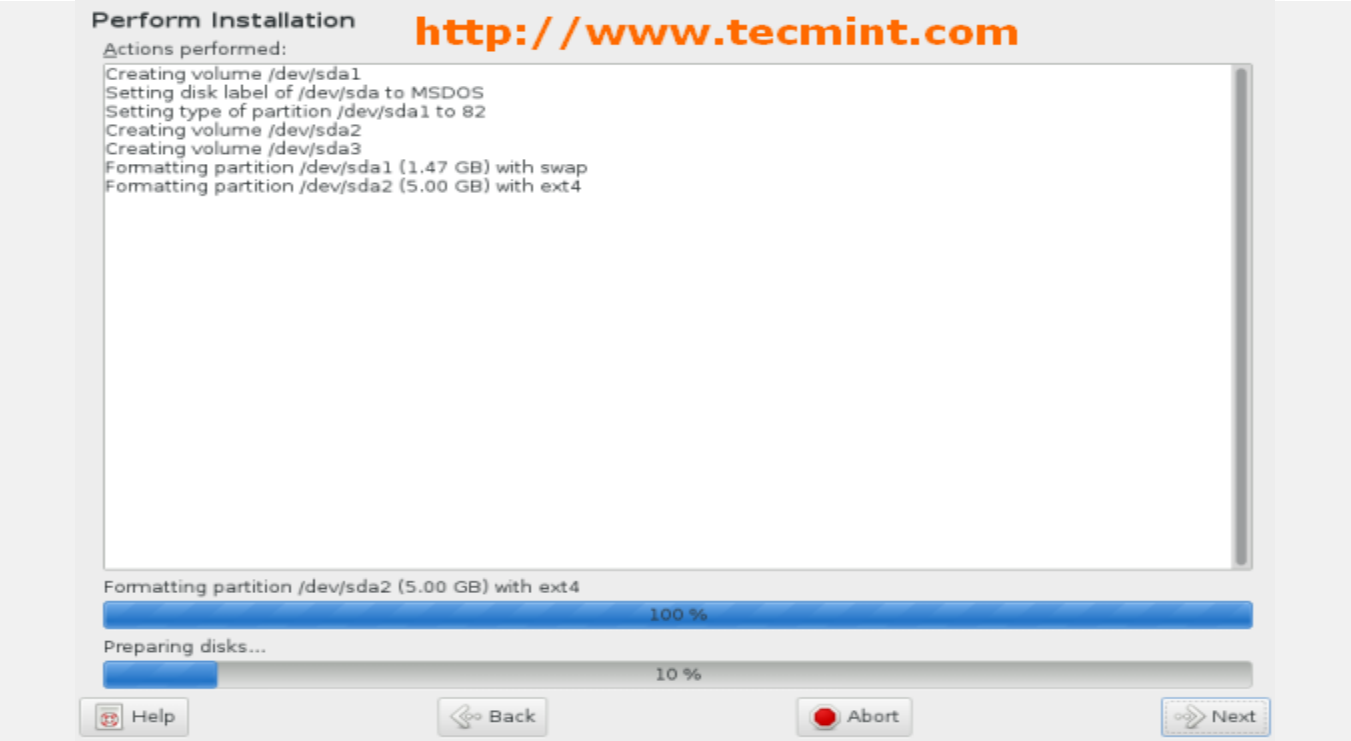
Verify OpenSuse Settings

12. Installation confirmation. Click on Install to proceed.



Start OpenSuse Installation

13. Performing installation. Creating volume and formatting filesystem for installation. Sit back and relax… This may take several time.

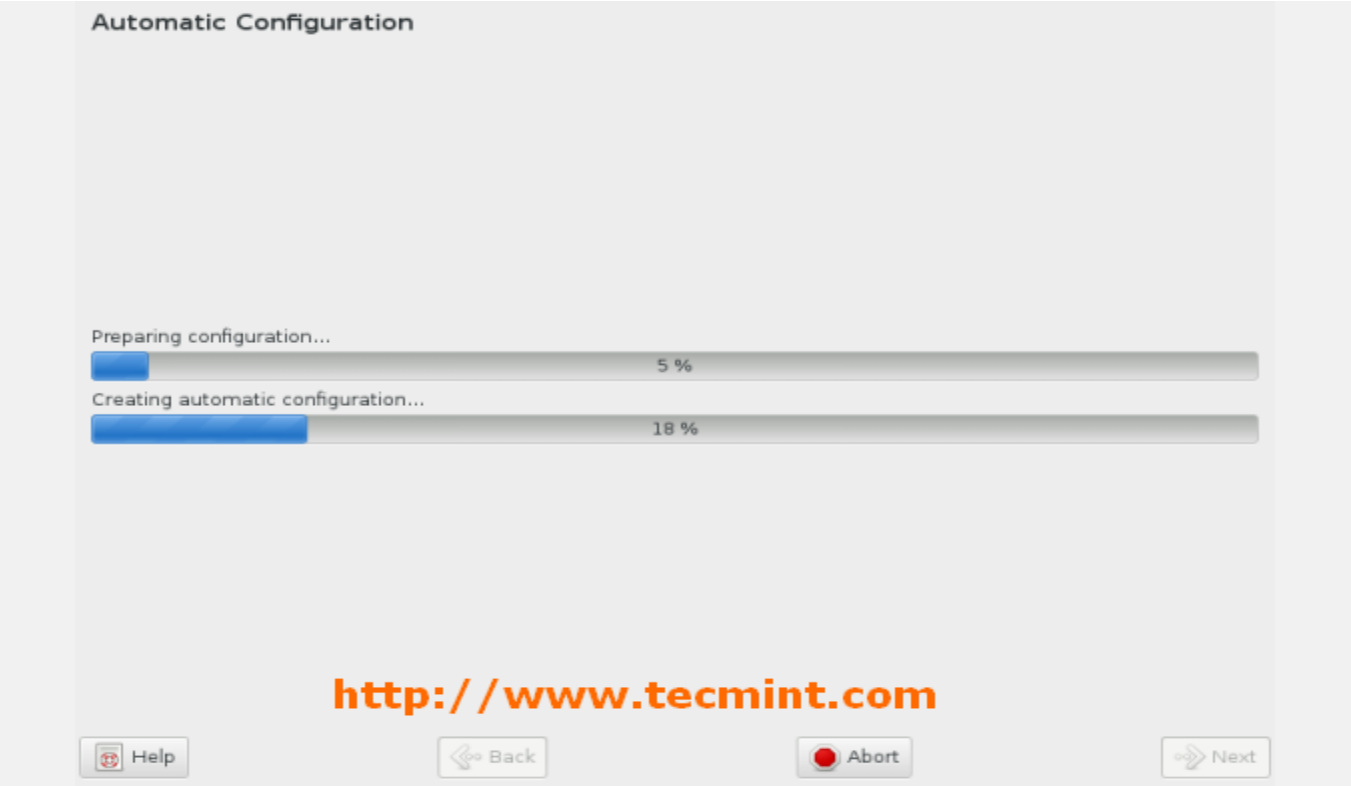


Performing OpenSuse Installation

14. Installation completed, remove installation media and click on Reboot Now.

OpenSuse Installation Completed

15. Post installation.



OpenSuse Post Installation

16. Login screen. Supply password for user created during installation.

OpenSuse Login Screen

17. openSUSE 12.3 Desktop.

Install Apache

Apache is a free open source software which runs over 50% of the world's web servers.

To install apache, open terminal and type in this command:

```
sudo yum install httpd
```

Once it installs, you can start apache running on your VPS:

```
sudo service httpd start
```

That's it.

Install MySQL

MySQL is a powerful database management system used for organizing and retrieving data on a virtual server

To install MySQL, open terminal and type in these commands:

```
sudo yum install mysql-server
sudo service mysqld start
```

During the installation, MySQL will ask you for your permission twice. After you say Yes to both, MySQL will install.

Install PHP

PHP is an open source web scripting language that is widely used to build dynamic webpages.

To install PHP on your virtual private server, open terminal and type in this command:

```
sudo yum install php php-mysql
```

Once you answer yes to the PHP prompt, PHP will be installed.

Result: Hence we have installed Linux, Apache, Mysql and Php.

Conclusion: Hence we have installed Linux, Apache, Mysql and Php.

EXPERIMENT NO. 2

AIM: Design HTML form and retrieve the values in PHP script.

OBJECTIVE: To learn how to retrieve values in PHP script.

TOOLS REQUIRED: Linux Operating System, PHP and Web Browser.

THEORY:

Both GET and POST create an array (e.g. array(key => value, key2 => value2, key3 => value3, ...)). This array holds key/value pairs, where keys are the names of the form controls and values are the input data from the user.

Both GET and POST are treated as $_GET and $_POST. These are superglobals, which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.

$_GET is an array of variables passed to the current script via the URL parameters.

$_POST is an array of variables passed to the current script via the HTTP POST method.

When to use GET?

Information sent from a form with the GET method is **visible to everyone** (all variable names and values are displayed in the URL). GET also has limits on the amount of information to send. The limitation is about 2000 characters. However, because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases.

GET may be used for sending non-sensitive data.

Note: GET should NEVER be used for sending passwords or other sensitive information!

When to use POST?

Information sent from a form with the POST method is **invisible to others** (all names/values are embedded within the body of the HTTP request) and has **no limits** on the amount of information to send.

Moreover POST supports advanced functionality such as support for multi-part binary input while uploading files to server.

However, because the variables are not displayed in the URL, it is not possible to bookmark the page.

Developers prefer POST for sending form data.

PROGRAM:

```
<html>
<body>

<form action="welcome.php" method="post">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```

When the user fills out the form above and clicks the submit button, the form data is sent for processing to a PHP file named "welcome.php". The form data is sent with the HTTP POST method.

To display the submitted data you could simply echo all the variables. The "welcome.php" looks like this:

```
<html>
<body>

Welcome <?php echo $_POST["name"]; ?><br>
Your email address is: <?php echo $_POST["email"]; ?>

</body>
</html>
```

Result:

Welcome John
Your email address is john.doe@example.com

Conclusion:Hence we have implemented retrieving values in PHP script.

EXPERIMENT NO. 3

AIM: PHP variables, arrays.

OBJECTIVE: To learn variables and arrays in PHP.

TOOLS REQUIRED: Linux Operating System, PHP and Web Browser.

THEORY:

PHP Variables

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).

Rules for PHP variables:

- A variable starts with the $ sign, followed by the name of the variable
- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive ($age and $AGE are two different variables)

Remember that PHP variable names are case-sensitive!

PHP is a Loosely Typed Language

PHP Variables Scope

In PHP, variables can be declared anywhere in the script.

The scope of a variable is the part of the script where the variable can be referenced/used.

PHP has three different variable scopes:

- local
- global
- static

What is an Array?

- An array is a special variable, which can hold more than one value at a time.
- If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:
- $cars1 = "Volvo";
  $cars2 = "BMW";
  $cars3 = "Toyota";

- However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?
- The solution is to create an array!
- An array can hold many values under a single name, and you can access the values by referring to an index number.

Create an Array in PHP

In PHP, the array() function is used to create an array:

array();

In PHP, there are three types of arrays:

- **Indexed arrays** - Arrays with a numeric index
- **Associative arrays** - Arrays with named keys
- **Multidimensional arrays** - Arrays containing one or more arrays

PHP Indexed Arrays

- There are two ways to create indexed arrays:
- The index can be assigned automatically (index always starts at 0), like this:
- $cars = array("Volvo", "BMW", "Toyota");
- or the index can be assigned manually:
- $cars[0] = "Volvo";
  $cars[1] = "BMW";
  $cars[2] = "Toyota";

PHP Associative Arrays

Associative arrays are arrays that use named keys that you assign to them.

There are two ways to create an associative array:

$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

or:

$age['Peter'] = "35";
$age['Ben'] = "37";
$age['Joe'] = "43";

PHP - Multidimensional Arrays

A multidimensional array is an array containing one or more arrays.

PHP understands multidimensional arrays that are two, three, four, five, or more levels deep. However, arrays more than three levels deep are hard to manage for most people.

The dimension of an array indicates the number of indices you need to select an element.

- For a two-dimensional array you need two indices to select an element
- For a three-dimensional array you need three indices to select an element

- PHP - Two-dimensional Arrays

- A two-dimensional array is an array of arrays (a three-dimensional array is an array of arrays of arrays).
- We can store the data from the table above in a two-dimensional array, like this:
- $cars = array
  (
  array("Volvo",22,18),
  array("BMW",15,13),
  array("Saab",5,2),
  array("Land Rover",17,15)
  );
- Now the two-dimensional $cars array contains four arrays, and it has two indices: row and column.

PROGRAM:

```
<?php
$txt = "W3Schools.com";
echo "I love $txt!";
?>
```

OUTPUT:

I love W3Schools.com!

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

OUTPUT:

I like Volvo, BMW and Toyota.

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

OUTPUT:

Peter is 35 years old.

```php
<!DOCTYPE html>
<html>
<body>

<?php
$cars = array
  (
  array("Volvo",22,18),
  array("BMW",15,13),
  array("Saab",5,2),
  array("Land Rover",17,15)
  );

echo $cars[0][0].": In stock: ".$cars[0][1].", sold: ".$cars[0][2].".<br>";
echo $cars[1][0].": In stock: ".$cars[1][1].", sold: ".$cars[1][2].".<br>";
echo $cars[2][0].": In stock: ".$cars[2][1].", sold: ".$cars[2][2].".<br>";
echo $cars[3][0].": In stock: ".$cars[3][1].", sold: ".$cars[3][2].".<br>";
?>

</body>
</html>
```

OUTPUT:

Volvo: In stock: 22, sold: 18.
BMW: In stock: 15, sold: 13.
Saab: In stock: 5, sold: 2.
Land Rover: In stock: 17, sold: 15.

Conclusion: Hence we have implemented variables and arrays in PHP.

EXPERIMENT NO. 4

AIM: PHP Functions: array, string, date-time, and calendar.

OBJECTIVE: To learn different types of built-in functions related to array, string, date-time and calendar in PHP.

TOOLS REQUIRED: Linux Operating System, PHP and Web Browser.

THEORY:

PHP Array Functions

Function	Description
array()	Creates an array
array_change_key_case()	Changes all keys in an array to lowercase or uppercase
array_chunk()	Splits an array into chunks of arrays
array_column()	Returns the values from a single column in the input array
array_combine()	Creates an array by using the elements from one "keys" array and one "values" array
array_count_values()	Counts all the values of an array
array_diff()	Compare arrays, and returns the differences (compare values only)
array_diff_key()	Compare arrays, and returns the differences (compare keys only)
sizeof()	Alias of count()
sort()	Sorts an indexed array in ascending order
uasort()	Sorts an array by values using a user-defined comparison function
uksort()	Sorts an array by keys using a user-defined comparison function
usort()	Sorts an array using a user-defined comparison function

PHP 5 String Functions

Function	Description
print()	Outputs one or more strings
printf()	Outputs a formatted string
sprintf()	Writes a formatted string to a variable
sscanf()	Parses input from a string according to a format
str_pad()	Pads a string to a new length
str_repeat()	Repeats a string a specified number of times
str_replace()	Replaces some characters in a string (case-sensitive)
str_split()	Splits a string into an array
str_word_count()	Count the number of words in a string
strcasecmp()	Compares two strings (case-insensitive)
strchr()	Finds the first occurrence of a string inside another string (alias of strstr())
strcmp()	Compares two strings (case-sensitive)
strcoll()	Compares two strings (locale based string comparison)
strlen()	Returns the length of a string
strnatcasecmp()	Compares two strings using a "natural order" algorithm (case-insensitive)

PHP Date/Time Introduction

The date/time functions allow you to get the date and time from the server where your PHP script runs. You can then use the date/time functions to format the date and time in several ways.

Function	Description
checkdate()	Validates a Gregorian date
date_add()	Adds days, months, years, hours, minutes, and seconds to a date
date_create_from_format()	Returns a new DateTime object formatted according to a specified format
date_create()	Returns a new DateTime object

date_date_set()	Sets a new date
date_diff()	Returns the difference between two dates
date_format()	Returns a date formatted according to a specified format
date_get_last_errors()	Returns the warnings/errors found in a date string
date_modify()	Modifies the timestamp
date_offset_get()	Returns the timezone offset
date_parse_from_format()	Returns an associative array with detailed info about a specified date, according to a specified format
date_parse()	Returns an associative array with detailed info about a specified date
date_time_set()	Sets the time
date_timestamp_get()	Returns the Unix timestamp
date_timestamp_set()	Sets the date and time based on a Unix timestamp
time()	Returns the current time as a Unix timestamp

PHP Calendar Introduction

The calendar extension contains functions that simplifies converting between different calendar formats.

It is based on the Julian Day Count, which is a count of days starting from January 1st, 4713 B.C.

Note: To convert between calendar formats, you must first convert to Julian Day Count, then to the calendar of your choice.

Note: The Julian Day Count is not the same as the Julian Calendar!

Function	Description
cal_days_in_month()	Returns the number of days in a month for a specified year and calendar
cal_from_jd()	Converts a Julian Day Count into a date of a specified calendar
cal_info()	Returns information about a specified calendar
cal_to_jd()	Converts a date in a specified calendar to Julian Day Count
easter_date()	Returns the Unix timestamp for midnight on Easter of a specified year

easter_days()	Returns the number of days after March 21, that the Easter Day is in a specified year
frenchtojd()	Converts a French Republican date to a Julian Day Count
gregoriantojd()	Converts a Gregorian date to a Julian Day Count
jddayofweek()	Returns the day of the week
jdmonthname()	Returns a month name

PROGRAM:

Example Of Array_flip Function

```
<?php
$array = array
(
"a" => "apple",
"b" => "banana",
"o"=>"orange"
);
print_r( array_flip($array) );
?>
```

Output Of Given Php Script

Array ([apple] => a [banana] => b [orange] => o)

Example Of Array Count Function

```
<?php
$name = array ("php","java","html");
$a=count($name);
echo "Number of element in an array: ";
echo $a;
?>
```

Output Of Given PHP Script

Number of element in an array: 3

Example Of Array Sum Function

```
<?php
$num = array(20, 50, 10);
echo "sum of array = " . array_sum($num);
?>
```

Output Of Given PHP Script

sum of array = 80

Example Of Strstr Function

```
<?php
```

```
$string = "understanding means transforming some text";
$txt = "some";
echo strstr($string, $txt);
?>
```

Output Of Given Php Script

some text

Example Of Strrev Function

```
<?php
echo strrev("Welcome to PHP");
?>
```

Output Of Given PHP Script

PHP ot emoclew

Example Of Strlen Function

```
<?php
$name="John";
if ( strlen( $name ) != 5 )
{
echo "Correct input.";
}
else
{
echo "incorrect input";
}
?>
```

Output Of Given PHP Script

Correct input

```
<?php
echo "Today is " . date("Y/m/d") . "<br>";
echo "Today is " . date("Y.m.d") . "<br>";
echo "Today is " . date("Y-m-d") . "<br>";
echo "Today is " . date("l");
?>
```

OUTPUT:

Today is 2017/11/29
Today is 2017.11.29
Today is 2017-11-29
Today is Wednesday

```
<?php
echo "The time is " . date("h:i:sa");
?>
```

OUTPUT:

The time is 01:33:17am

Conclusion: Hence we have implemented PHP functions.

EXPERIMENT NO. 6

AIM: MySQLi connectivity, INSERT, SELECT, DELETE with PHP

OBJECTIVE: To learn Mysql database connectivity using PHP and performing operations using PHP.

TOOLS REQUIRED: Linux Operating system, PHP, Mysql and web browser.

THEORY:

After a database and a table have been created, we can start adding data in them.

Here are some syntax rules to follow:

- The SQL query must be quoted in PHP
- String values inside the SQL query must be quoted
- Numeric values must not be quoted
- The word NULL must not be quoted

The INSERT INTO statement is used to add new records to a MySQL table:

```
INSERT INTO table_name (column1, column2, column3,...)
VALUES (value1, value2, value3,...)
```

Select Data From a MySQL Database

The SELECT statement is used to select data from one or more tables:

```
SELECT column_name(s) FROM table_name
```

or we can use the * character to select ALL columns from a table:

```
SELECT * FROM table_name
```

Delete Data From a MySQL Table Using MySQLi and PDO

The DELETE statement is used to delete records from a table:

```
DELETE FROM table_name
WHERE some_column = some_value
```

PROGRAM:

The following examples add a new record to the "MyGuests" table:

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "INSERT INTO MyGuests (firstname, lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";

if ($conn->query($sql) === TRUE) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}

$conn->close();
?>
```

The following example selects the id, firstname and lastname columns from the MyGuests table and displays it on the page:

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "<br>";
    }
} else {
    echo "0 results";
}
```

```
$conn->close();
?>
```

OUTPUT:
id: 1 - Name: John Doe
id: 2 - Name: Mary Moe
id: 3 - Name: Julie Dooley

The following examples delete the record with id=3 in the "MyGuests" table:

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// sql to delete a record
$sql = "DELETE FROM MyGuests WHERE id=3";

if ($conn->query($sql) === TRUE) {
    echo "Record deleted successfully";
} else {
    echo "Error deleting record: " . $conn->error;
}

$conn->close();
?>
```

After the record is deleted, the table will look like this:

id	firstname	lastname	email	reg_date
1	John	Doe	john@example.com	2014-10-22 14:26:15
2	Mary	Moe	mary@example.com	2014-10-23 10:22:30

Conclusion: Hence we have implemented Mysql connectivity and basic operations on database with PHP.

EXPERIMENT NO. 5

AIM: PHP Mysqli connectivity.

OBJECTIVE: To learn connecting to Mysql database.

TOOLS REQUIRED: Linux operating syste,. PHP and Mysql.

THEORY:

To connect PHP to MySQL database you need to know following important things:

1. Host name.
2. MySQL user name.
3. MySQL password.

PROGRAM:

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = new mysqli($servername, $username, $password);

// Check connection
if ($conn->connect_error) {
   die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully"
?>
```

Output:

Connected successfully

Conclusion: Hence we have implemented PHP Mysql connectivity.

EXPERIMENT NO. 7

AIM: PHP script for File uploading.

OBJECTIVE: To learn file uploading using PHP script.

TOOLS REQUIRED: Linux operating system, PHP.

THEORY:

With PHP, it is easy to upload files to the server.

Configure The "php.ini" File

First, ensure that PHP is configured to allow file uploads.

In your "php.ini" file, search for the file_uploads directive, and set it to On:

file_uploads = On

Some rules to follow for the HTML form above:

- Make sure that the form uses method="post"
- The form also needs the following attribute: enctype="multipart/form-data". It specifies which content-type to use when submitting the form

Without the requirements above, the file upload will not work.

Other things to notice:

- The type="file" attribute of the <input> tag shows the input field as a file-select control, with a "Browse" button next to the input control

$target_dir = "uploads/" - specifies the directory where the file is going to be placed

$target_file specifies the path of the file to be uploaded

$uploadOk=1 is not used yet (will be used later)

$imageFileType holds the file extension of the file (in lower case)

PROGRAM:

Create The HTML Form

Next, create an HTML form that allow users to choose the image file they want to upload:

```
<!DOCTYPE html>
<html>
<body>

<form action="upload.php" method="post" enctype="multipart/form-data">
    Select image to upload:
    <input type="file" name="fileToUpload" id="fileToUpload">
    <input type="submit" value="Upload Image" name="submit">
</form>

</body>
</html>
```

Create The Upload File PHP Script

The "upload.php" file contains the code for uploading a file:

```
<?php
$target_dir = "uploads/";
$target_file = $target_dir . basename($_FILES["fileToUpload"]["name"]);
$uploadOk = 1;
$imageFileType = pathinfo($target_file,PATHINFO_EXTENSION);
// Check if image file is a actual image or fake image
if(isset($_POST["submit"])) {
    $check = getimagesize($_FILES["fileToUpload"]["tmp_name"]);
    if($check !== false) {
        echo "File is an image - " . $check["mime"] . ".";
        $uploadOk = 1;
    } else {
        echo "File is not an image.";
        $uploadOk = 0;
    }
}
?>
```

Conclusion: Hence we have implemented file uploading in PHP.

EXPERIMENT NO. 8

AIM: PHP script for-Session Management (login form).

OBJECTIVE: To learn session management in PHP.

TOOLS REQUIRED: Linux operating system, PHP, Web browser.

THEORY:

An alternative way to make data accessible across the various pages of an entire website is to use a PHP Session.

A session creates a file in a temporary directory on the server where registered session variables and their values are stored. This data will be available to all pages on the site during that visit.

The location of the temporary file is determined by a setting in the **php.ini** file called **session.save_path**. Before using any session variable make sure you have setup this path.

When a session is started following things happen −

- PHP first creates a unique identifier for that particular session which is a random string of 32 hexadecimal numbers such as 3c7foj34c3jj973hjkop2fc937e3443.
- A cookie called **PHPSESSID** is automatically sent to the user's computer to store unique session identification string.
- A file is automatically created on the server in the designated temporary directory and bears the name of the unique identifier prefixed by sess_ ie sess_3c7foj34c3jj973hjkop2fc937e3443.

When a PHP script wants to retrieve the value from a session variable, PHP automatically gets the unique session identifier string from the PHPSESSID cookie and then looks in its temporary directory for the file bearing that name and a validation can be done by comparing both values.

A session ends when the user loses the browser or after leaving the site, the server will terminate the session after a predetermined period of time, commonly 30 minutes duration.

Starting a PHP Session

A PHP session is easily started by making a call to the **session_start()**function.This function first checks if a session is already started and if none is started then it starts one. It is recommended to put the call to **session_start()** at the beginning of the page.

Session variables are stored in associative array called **$_SESSION[]**. These variables can be accessed during lifetime of a session.

The following example starts a session then register a variable called **counter**that is incremented each time the page is visited during the session.

Make use of **isset()** function to check if session variable is already set or not.

Destroying a PHP Session

A PHP session can be destroyed by **session_destroy()** function. This function does not need any argument and a single call can destroy all the session variables. If you want to destroy a single session variable then you can use **unset()** function to unset a session variable.

PHP login with session

Php login script is used to provide the authentication for our web pages. the Script executes after submitting the user login button.

Login Page

Login page should be as follows and works based on session. If the user close the session, it will erase the session data.

PROGRAM:

```
<?php
  ob_start();
  session_start();
?>

<?
  // error_reporting(E_ALL);
  // ini_set("display_errors", 1);
?>

<html lang = "en">
```

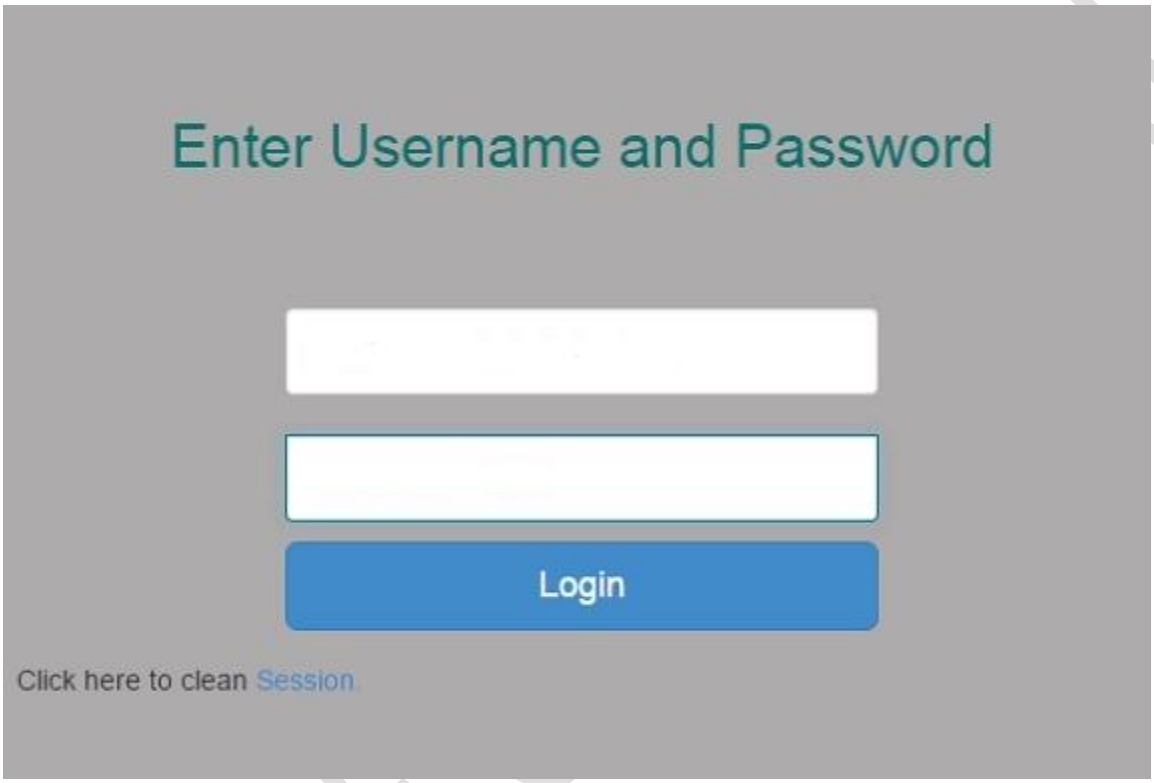
```
<head>
  <title>Tutorialspoint.com</title>
  <link href = "css/bootstrap.min.css" rel = "stylesheet">

  <style>
    body {
      padding-top: 40px;
      padding-bottom: 40px;
      background-color: #ADABAB;
    }

    .form-signin {
      max-width: 330px;
      padding: 15px;
      margin: 0 auto;
      color: #017572;
    }

    .form-signin .form-signin-heading,
    .form-signin .checkbox {
      margin-bottom: 10px;
    }

    .form-signin .checkbox {
      font-weight: normal;
    }

    .form-signin .form-control {
      position: relative;
      height: auto;
      -webkit-box-sizing: border-box;
      -moz-box-sizing: border-box;
      box-sizing: border-box;
      padding: 10px;
      font-size: 16px;
    }

    .form-signin .form-control:focus {
      z-index: 2;
    }

    .form-signin input[type="email"] {
      margin-bottom: -1px;
      border-bottom-right-radius: 0;
      border-bottom-left-radius: 0;
      border-color:#017572;
    }

    .form-signin input[type="password"] {
      margin-bottom: 10px;
      border-top-left-radius: 0;
      border-top-right-radius: 0;
      border-color:#017572;
    }
```

```
      h2{
        text-align: center;
        color: #017572;
      }
    </style>

  </head>

  <body>

    <h2>Enter Username and Password</h2>
    <div class = "container form-signin">

      <?php
        $msg = '';

        if (isset($_POST['login']) && !empty($_POST['username'])
          && !empty($_POST['password'])) {

          if ($_POST['username'] == 'tutorialspoint' &&
            $_POST['password'] == '1234') {
            $_SESSION['valid'] = true;
            $_SESSION['timeout'] = time();
            $_SESSION['username'] = 'tutorialspoint';

            echo 'You have entered valid use name and password';
          }else {
            $msg = 'Wrong username or password';
          }
        }
      ?>
    </div> <!-- /container -->

    <div class = "container">

      <form class = "form-signin" role = "form"
        action = "<?php echo htmlspecialchars($_SERVER['PHP_SELF']);
        ?>" method = "post">
        <h4 class = "form-signin-heading"><?php echo $msg; ?></h4>
        <input type = "text" class = "form-control"
          name = "username" placeholder = "username = tutorialspoint"
          required autofocus></br>
        <input type = "password" class = "form-control"
          name = "password" placeholder = "password = 1234" required>
        <button class = "btn btn-lg btn-primary btn-block" type = "submit"
          name = "login">Login</button>
      </form>

      Click here to clean <a href = "logout.php" tite = "Logout">Session.

    </div>

  </body>
</html>
```

Logout.php
It will erase the session data.

```
<?php
  session_start();
  unset($_SESSION["username"]);
  unset($_SESSION["password"]);

  echo 'You have cleaned session';
  header('Refresh: 2; URL = login.php');
?>
```

Result:



Conclusion: Hence we have implemented PHP script for-Session Management (login form).

EXPERIMENT NO. 9

AIM: AJAX Script using XMLHttpRequest PHP.

OBJECTIVE:

TOOLS REQUIRED:

THEORY:

What is AJAX ?

- AJAX stands for **A**synchronous **Ja**vaScript and **X**ML. AJAX is a new technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS and Java Script.
- Conventional web application transmit information to and from the sever using synchronous requests. This means you fill out a form, hit submit, and get directed to a new page with new information from the server.
- With AJAX when submit is pressed, JavaScript will make a request to the server, interpret the results and update the current screen. In the purest sense, the user would never know that anything was even transmitted to the server.

PROGRAM:

```
<html>
<head>
<script>
function showHint(str) {
   if (str.length == 0) {
      document.getElementById("txtHint").innerHTML = "";
      return;
   } else {
      var xmlhttp = new XMLHttpRequest();
      xmlhttp.onreadystatechange = function() {
         if (this.readyState == 4 && this.status == 200) {
            document.getElementById("txtHint").innerHTML = this.responseText;
         }
      };
      xmlhttp.open("GET", "gethint.php?q=" + str, true);
      xmlhttp.send();
   }
}
</script>
</head>
<body>

<p><b>Start typing a name in the input field below:</b></p>
```

```
<form>
First name: <input type="text" onkeyup="showHint(this.value)">
</form>
<p>Suggestions: <span id="txtHint"></span></p>
</body>
</html>
```

Result:

The XMLHttpRequest Object

Start typing a name in the input field below:

First name: a

Suggestions: Anna, Amanda

Conclusion: Hence we have implemented AJAX Script using XMLHttpRequest.

EXPERIMENT NO. 10

AIM: PHP script to update and retrieve data stored in database from user using Ajax.

OBJECTIVE: To learn how to update and retrieve data using Ajax.

TOOLS REQUIRED: Linux operating system, PHP, Mysql, Browser.

THEORY:

To clearly illustrate how easy it is to access information from a database using Ajax and PHP, we are going to build MySQL queries on the fly and display the results on "ajax.html". But before we proceed, lets do ground work. Create a table using the following command.

NOTE − We are assuming you have sufficient privilege to perform following MySQL operations.

```
CREATE TABLE `ajax_example` (
  `name` varchar(50) NOT NULL,
  `age` int(11) NOT NULL,
  `sex` varchar(1) NOT NULL,
  `wpm` int(11) NOT NULL,
  PRIMARY KEY  (`name`)
)
```

Now dump the following data into this table using the following SQL statements.

```
INSERT INTO `ajax_example` VALUES ('Jerry', 120, 'm', 20);
INSERT INTO `ajax_example` VALUES ('Regis', 75, 'm', 44);
INSERT INTO `ajax_example` VALUES ('Frank', 45, 'm', 87);
INSERT INTO `ajax_example` VALUES ('Jill', 22, 'f', 72);
INSERT INTO `ajax_example` VALUES ('Tracy', 27, 'f', 0);
INSERT INTO `ajax_example` VALUES ('Julie', 35, 'f', 90);
```

PROGRAM:

Client Side HTML file

Now lets have our client side HTML file which is ajax.html and it will have following code

```
<html>
  <body>

    <script language = "javascript" type = "text/javascript">
      <!--
        //Browser Support Code
        function ajaxFunction(){
          var ajaxRequest;  // The variable that makes Ajax possible!

          try {
            // Opera 8.0+, Firefox, Safari
            ajaxRequest = new XMLHttpRequest();
          }catch (e) {
            // Internet Explorer Browsers
```

```
            try {
              ajaxRequest = new ActiveXObject("Msxml2.XMLHTTP");
            }catch (e) {
              try{
                ajaxRequest = new ActiveXObject("Microsoft.XMLHTTP");
              }catch (e){
                // Something went wrong
                alert("Your browser broke!");
                return false;
              }
            }
          }

          // Create a function that will receive data
          // sent from the server and will update
          // div section in the same page.

          ajaxRequest.onreadystatechange = function(){
            if(ajaxRequest.readyState == 4){
              var ajaxDisplay = document.getElementById('ajaxDiv');
              ajaxDisplay.innerHTML = ajaxRequest.responseText;
            }
          }

          // Now get the value from user and pass it to
          // server script.

          var age = document.getElementById('age').value;
          var wpm = document.getElementById('wpm').value;
          var sex = document.getElementById('sex').value;
          var queryString = "?age=" + age ;

          queryString +=  "&wpm=" + wpm + "&sex=" + sex;
          ajaxRequest.open("GET", "ajax-example.php" + queryString, true);
          ajaxRequest.send(null);
        }
      //-->
    </script>

    <form name = 'myForm'>
      Max Age: <input type = 'text' id = 'age' /> <br />
      Max WPM: <input type = 'text' id = 'wpm' />
      <br />

      Sex: <select id = 'sex'>
        <option value = "m">m</option>
        <option value = "f">f</option>
```

```
      </select>

      <input type = 'button' onclick = 'ajaxFunction()' value = 'Query MySQL'/>

    </form>

    <div id = 'ajaxDiv'>Your result will display here</div>
  </body>
</html>
```

Server Side PHP file

So now your client side script is ready. Now we have to write our server side script which will fetch age, wpm and sex from the database and will send it back to the client. Put the following code into "ajax-example.php" file.

```
<?php
  $dbhost = "localhost";
  $dbuser = "dbusername";
  $dbpass = "dbpassword";
  $dbname = "dbname";

  //Connect to MySQL Server
  mysql_connect($dbhost, $dbuser, $dbpass);

  //Select Database
  mysql_select_db($dbname) or die(mysql_error());

  // Retrieve data from Query String
  $age = $_GET['age'];
  $sex = $_GET['sex'];
  $wpm = $_GET['wpm'];

  // Escape User Input to help prevent SQL Injection
  $age = mysql_real_escape_string($age);
  $sex = mysql_real_escape_string($sex);
  $wpm = mysql_real_escape_string($wpm);

  //build query
  $query = "SELECT * FROM ajax_example WHERE sex = '$sex'";

  if(is_numeric($age))
  $query .= " AND age <= $age";

  if(is_numeric($wpm))
  $query .= " AND wpm <= $wpm";
```

```php
  //Execute query
  $qry_result = mysql_query($query) or die(mysql_error());

  //Build Result String
  $display_string = "<table>";
  $display_string .= "<tr>";
  $display_string .= "<th>Name</th>";
  $display_string .= "<th>Age</th>";
  $display_string .= "<th>Sex</th>";
  $display_string .= "<th>WPM</th>";
  $display_string .= "</tr>";

  // Insert a new row in the table for each person returned
  while($row = mysql_fetch_array($qry_result)) {
    $display_string .= "<tr>";
    $display_string .= "<td>$row[name]</td>";
    $display_string .= "<td>$row[age]</td>";
    $display_string .= "<td>$row[sex]</td>";
    $display_string .= "<td>$row[wpm]</td>";
    $display_string .= "</tr>";
  }
  echo "Query: " . $query . "<br />";

  $display_string .= "</table>";
  echo $display_string;
?>
```

Result:

Max Age:

Max WPM:

Sex: m ▾ Query MySQL

Your result will display here

Conclusion: Hence we have implemented PHP script to update and retrieve data stored in database from user using Ajax.

3. Quiz on the Subject

What is Linux?
What is the difference between UNIX and LINUX?
Why we use LINUX?
What's the difference between the include() and require() functions?
What is the difference between GET and POST?
What is PHP?
What is the use of "echo" in php?
How to include a file to a php page?
How to declare an array in php?
What is the use of 'print' in php?
What is use of in_array() function in php ?
What is the difference between Session and Cookie?
How to set cookies in PHP?
How to Retrieve a Cookie Value?
How to create a session? How to set a value in session ? How to Remove data from a session?
How to create a mysql connection?
How to execute an sql query? How to fetch its result ?
How to create a text file in php?
What are the different types of errors in PHP ?
What is the purpose of php.ini file?
What are the different types of PHP variables?
Explain the syntax for 'foreach' loop.
What is associate array?
What is Multidimensional array?
How will you concatenate two strings in PHP?
How will you find the length of a string in PHP?
What is Ajax?
What are Ajax applications?
What are the advantages of Ajax?
What are all the technologies used by Ajax?
What are all the browsers support AJAX?
How can we cancel the XMLHttpRequest in AJAX?
What are the protocols used by Ajax?
What is XML?
What are the features of XML?
What are the differences between HTML and XML?
What is XML DOM Document?
What is XPath?
What is an attribute?
What is XML Element?
What is XML Parser?

4. Conduction of VIVA-VOCE Examinations:

Teacher should conduct oral exams of the students with full preparation. Normally the objective questions with guess are to be avoided. To make it meaningful, the questions should be such that depth of the student in the subject is tested. Oral Exams are to be conducted in co-cordial situation. Teachers taking oral exams should not have ill thoughts about each other & courtesies should be offered to each other in case of opinion, which should be critically suppressed in front of the students.

5. Evaluation and marking system:

Basic honesty in the evaluation and marking system is essential and in the process impartial nature of the evaluator is required in the exam system. It is a primary responsibility of the teacher to see that right students who really put their effort & intelligence are correctly awarded.

The marking pattern should be justifiable to the students without any ambiguity and teacher should see that students are faced with just circumstance.