

Aim: Virtualization, Installation of virtual machine software and installing of operating system on virtual machine.

Virtualization: In computing, virtualization refers to the act of creating a virtual version of something. Including virtual computer hardware platforms, operating systems, storage devices, and computer network resources. Virtualization began in the 1960s, as a method of logically dividing the system resources provided by mainframe computers between different applications. Since then, the meaning of the term has broadened.

Install VMware Workstation

1). Run the setup file:-

Double click the file that you downloaded. If the user Account control or open file - security warning dialog appears, click 'yes' or 'RUN' respectively.

2). Click Next > to dismiss the welcome dialog box.

3). Select the I accept the terms in linear agreement option. then click next.

4). Click the typical button.

5). Choose the directory in which you want to install VMware workstation. To install it into a directory other than the default one, click change and because to the desired directory. After deciding on

Teacher's Signature : _____

where to install VMware workstation, click Next >

6). Decide if you want VMware workstation to check for updates everytime it is opened then click Next >

7). Decide if you want to send information about VMware workstation's performance on your computer to VMware. This information is used to help improve the program. Click 'Next' > after you have made decision.

8). Select the location that you want to setup to create shortcuts to VMware workstation in. choices include the desktop and the start menu. Deselect or any shortcuts you do not want the installer to create and then click Next >.

9) The installer is ready to begin installing VMware workstation. If you want to change any options, click < Back to return to them change them, then click Next > until you reach this screen again once you have finished. Once you are ready to begin installing VMware workstation, click continue

10). Register VMware workstation. Type a license keyword click Enter >.

11). Close setup. click the finish button once setup complete its operations.

Teacher's Signature : _____

AIM 6 Basics of shell programming. Various types of shell, shell programming in Bash.

Shells The shell provides you with an interface to the UNIX system. It gathers input from you and executes programs based on that input. When a program ~~is~~ finished executing, it displays that programs out.

A shell is an environment in which we can run our commands, programs, and shell scripts. There are different flavours of shells, just as there are different flavors of operating systems. Each flavours of shell has its own set of recognized commands and function.

Shell Prompt: The prompt, \$, which is called command prompt, is issued by the shell. While the prompt is displayed, you can type a command.

The shell reads your input after you press 'Enter'. It determines the command you want executed by looking at the first word of your input. A word is an unbroken set of characters. Spaces and tabs separate words.

Following is a simple example of date command which displays current date and time.

\$ date

2020 Sun May 23 12:30:28 MST 2020

Teacher's Signature : _____

Shell types

In UNIX there are two types of shells

1. The Bourne shell. If you are using a Bourne shell, the default prompt is the \$ character.
2. The C shell. If you are using a C-type shell, the default prompt is the % character. There are again various subcategories for Bourne shell which are listed as follows:
 - (i). Bourne shell (sh)
 - (ii). Korn shell (ksh)
 - (iii). Bourne Again shell (bash)
 - (iv). POSIX shell (sh)

The different C-type shells follows:

- (i). C shell (csh)
- (ii). TENEX/TOPS C shell (tcsh)

The Bourne shell was the first shell to appear on UNIX system, thus it is referred to as "the shell". The Bourne shell is usually installed as /bin/sh on most version of UNIX. For this reason, it is the shell of choice for writing scripts to use on several different versions of UNIX.

Teacher's Signature : _____

SHELL programming in BASH

Bash as scripting language -

To create ~~file~~ a bash script `#!/bin/bash` at top of file.

Then change the permission on the file to make it executable : `chmod +x scriptname`.

To execute the script from the current directory, you can run `./scriptname` and pass any parameters you wish.

When the shell executes a script, it finds the `#!/path` to interpreter.

It then runs the interpreter (in this case, `/bin/bash`) on the file itself.

The `#!/` convention is why so many scripting languages use `#` for comments.

Here's an example bash script that prints out the first ~~arg~~ argument.

```
#!/bin/bash
```

```
# Use $1 to get the first argument:  
echo $1.
```

Teacher's Signature : _____

AIM: Implementation of shell variables and shell keywords.

Shell variables: A variable is a character string to which we assign a value. The value assigned could be a number, text, filename, device, or any other type of data.

Variable names: The name of a variable can contain only letters (a to z or A to Z), numbers (0 to 9) or the underscore character (-).

By convention, UNIX shell variables would have their names in UPPERCASE.

The following examples are valid variable names—

-ALT

TOKEN-A

VAR-L

VAR-2

following are the examples of invalid variable names—

2-VAR

-Variable

VAR! - VAR2

VAR-!

The reason you cannot use another characters such

Teacher's Signature : _____

Expt. No. _____

as /, * or - is that characters have a special meaning for the shell.

Defining variables

Variable name = Variable_value

e.g.,

NAME = "The Hacker"

Above example defines variable Name and assign in the value "The Hacker". Variable of this type are called scalar variable. A scalar can hold one value at a time.

The shell enables you to store any value you want in a variable. e.g.,

VAR1 = "The Hacker"

VAR2 = 100

Accessing variables

To access values stored in the variable, prefix its name with the dollar sign (\$).

Read only variables

The shell provides a way to mark variables as read only by using the readonly command.

After a variable is marked read-only, its value cannot be changed.

Teacher's Signature : _____

Unsetting Variables

Unsetting or deleting a variable tells the shell to remove that it tracks. Once you unset a variable, you would not be able to access stored value in the variables.

Variable types

When a shell is running, three main types of variables are present.

- (i). Local variables - A local variable is a variable that is present within the current instance of the shell. It is not available to programs that are started by the shell. They are set at command prompt.
- (ii). Environment variables - An environment variable is a variable to any child process of shell. Some programs need environment variables in order to function correctly. Usually a shell script defines only those environment variables that are needed by the programs that it runs.
- (iii). Shell variables - A shell variable is a special variable that is set by the shell and is required by the shell in order to function correctly. Some of these variables are environment variables whereas others are local variables.

Teacher's Signature : _____

Special Variables: These variables are reserved for specific function.
e.g., the \$ character represents the process ID number, or PID, of the current shell:
~~\$ shell \$\$ \$ echo \$\$~~

Above example command would write PID of the current shell -
29949

The following table shows a number of special variables that you can use in your shell scripts -

Variable	Description
\$0	The filename of the current script.
\$n	These variables correspond to the arguments with which a script was invoked. Here n is a positive decimal number corresponding to the position of an argument (the first argument is \$1, the second argument is \$2, and so on).
\$#	The number of arguments supplied to a script.
\$*	All the arguments are double quoted. If a script receives two arguments, \$* is equivalent to \$2.

Teacher's Signature : _____

Expt. No. _____

\$@	All the arguments are individually double quoted. If a script receives two arguments, \$@ is equivalent to \$\$2.
\$?	The exit status of the last command executed.
\$\$	The process number of the current shell. For shell scripts this is the process ID under which they are executed.
\$!	The process number of the last background command.

Shell keywords: A keyword is a reserved word, token or operator...Keywords have a special meaning to the shell, and indeed are the building blocks of the shell's syntax. Similar to a built-in, a keyword is hard-coded into Bash, but unlike a built-in, a keyword is not in itself a command, but a subset of a command construct.

Various shell keywords are:

- for
- while
- do
- ! etc

Teacher's Signature : _____

AIM: Implement condition statements, looping statements and case statements in shell programming.

Conditional Statements

Alternatively referred to as a conditional expression and conditional processing, a conditional statement is a set of rules performed if a certain condition is met. Bash conditional statements performs different computations or actions depending on whether a programmer specified boolean condition evaluates to true or false. These statements are used to execute different parts of your shell programs depending on whether certain conditions are true. The ability to branch makes shell scripts powerful.

In Bash, we have the following conditional statements:

1. Bash 'if --- then ---- fi' statements

This if statement is also called as simple if statement. If the given conditional expression is true, it enters and executes the statements enclosed between the keywords "then" and "fi".

2. Bash 'if---then---else---fi' statements

3. Bash 'if--- elif--- else--- fi'

4. Bash 'if--- then--- else--- if---then---fi'

If statement could be nested in bash. The keyword

Teacher's Signature : _____

"fi" indicates the end of the inner if statement and all if statement should end with keyword "fi".

Looping Statements

If we want to repeat task twenty times, we don't want to have in the code twenty times, with maybe slight change each time.

As a result we have for and while loops in Bourne shell.

→ for Loops for loops iterate through a set of values until the list is exhausted.

→ while loops while loop is used when we don't know number of steps in advance.

Case statements

Many programming languages and scripting languages have the concept of a case or select statement. This is generally used as a shortcut for writing if / else statements. The case statement is always preferred when there are many items to select from instead of using a large if / elif / else statement. If is usually used to implement menus in a script. The case statement is terminated with esac (case back ward). Here is a simple example of using case statement.

Teacher's Signature : _____

Date _____

Expt. No. 06

Page No. 16

AIM: Study and usage of vi editor.

Study of vi Editor:

Editors An editor allows the users to see a portion of a file on the screen and to modify characters and lines by simply typing at the cursor position.

VI Editors

- VI stands for visual.
- It is a full screen editor and allows the user to view and edit the entire document at the same time.
- VI is case sensitive.
- It has powerful undo feature

Modes of vi editor - vi editor work in 3 modes.

Command mode: In this mode all the keys pressed by the user are interpreted to be editor commands. No text is displayed on the screen even if corresponding keys are pressed on the keyboard.

Insert mode: This mode permits to insert a new text, editing and replacement of existing text. When vi editor is in insert mode the letters typed at the keyboard are echoed on the screen.

Teacher's Signature : _____

Escape mode: Commands typed at the command line.

Starting with vi editor:

Syntax : vi filename

Moving the cursor

The cursor movement commands are :

Command	Action
H or back space	left one character
L or spacebar	Right one character
K or -	Up one line
J or T	Down one line
I	Moves forward a word
#b	Moves back a word
#e	Moves to the last character in the word.
f [character]	Moves right to the specified character in a line
T [character]	Moves right and places one character before the specified character
	Teacher's Signature : _____

D or ^	Move to the beginning of the file.
# \$	Move to the end of the file.
L	Moves to the line of the file.
(n)	Moves to the specified line number

Editing the files

- Open the file using \$ vi filename.
- To add text at the end of the file, position the cursor at the last character of the file.
- Switch from command mode to text input mode by pressing 'i'.
- Here 'i' stands for append.
- Inserting text in the middle of the file is possible by pressing 'i'. The editor accepts and inserts the typed character until Esc key is pressed.

<u>Command</u>	<u>Purpose</u>
i	Inserts text to the left of the cursor
g	Inserts text at the beginning of the line.
A	Appends text to the right of the cursor.
A	Appends text at the end of the line.
o	Appends a new line below.
O	Appends a line above

Teacher's Signature : _____

Deleting texts

For deleting text move the cursor to the character, press 'x'. The character will disappear

<u>Command</u>	<u>Purpose</u>
x	Deletes one character.
Nx	Deletes n number of character
#x	Deletes one character at the cursor position.
#X	Deletes on the character before cursor position.
D\$ or d	Deletes a line from the cursor position to the end of the line.
DO	Deletes from the cursor position to the starting of the line.
#dd	Deletes the current line where the cursor is positioned.
#dw	Deletes the word from the cursor position to the end of the word.

Ans/note 9.2.23

The Undo feature ~~is~~ undo the recent changes.
Uundo all changes in the current line

Saving texts : cn - save the file and remains in edit mode.

:wq - save the file and quits from edit mode.

q! - quit without changes from edit mode.

Quitting vi : press ZZ or ":wq" in command mode.

Teacher's Signature : _____