**Devops Internship**

* **Basic Linux**

1. Birth of linux : Firest Birth of unix Os

Step 1 : 1)Growth of unix because of open source collbration.

2) commercial scale of unix.

Step 2 : 1)companies devloping their own unix -IBM(AIX),

solaris (Sun-os ) ,HP(HP-UX).

2)Birth of software movement (GNU project).

Step 3: 1)Linus Torvalds put unix kernel source code online .

2)Resulted in usage of 'Linux + GNU'.

* Various Linux distrubution

1) Red hat linux Enterprises Linux.

2) Fedora

3) Debian

4) Others (Ubuntu,Linux mint, Cent os).

* Features

1) Simplified update for all installed Software.

2) Free Software licensing.

3) Access to Source code.

4) Better malware Protection.

5)Multiple distrubution.

* Basic Linux Command

Linux provids a CLI (command line interface) to communicate with os.

CLI is the better of the task which cannot be performed with the GUI.

* Command : 1) man : Description of the command, eg. man ls .

2) pwd : present working directory.(current working directory of the terminal)

3) clear : whole CLI is clear.(command dont deleted ,still present. ctrl + L)

4) ls : List of folder or Document.

5) cd : go to the differnt directory.

6) cd .. : back to the directory.

7) echo : write its argument to standard output.(print argument)

8) su : switch to root user.

9) su <username> : switch to differnt user.

10) sudo : allows you to run programs with the security privileges of another user

(by default, as the superuser).

11) ls <path> : content in that path will be displayed.

12) ls -l : list all the contents along with its owner setting ,permission and

timestamp.

13) ls -a : list all the content in specified directory.

14) ls --author : list all the contents in the specified directory along with its owner.

15) cd ~ : chnages to home directory.

17) cd / : changes to root directory.

18) touch : create a empty files.

19) cat : concatenate (see the content of file.)

20) history : that you previously entered.(ctrl+R).

21) what is : provide brief description command line programs.

22) exit : exit from the cell.

* Working with files : cat command

Displays the content of text files and concatenate sevral files into one.

1) cat -b : add line numbers to non blank lines.

2) cat -n : add line numbers to all lines.

3) cat -s : squeze blank lines into one line.

4) cat -E : show $ at the end line.

5) cat >file.txt : the '>' flag can be used to create a new file and enter text content

from the terminal.

6) cat >> file.txt : the '>>' flag can be used to append text contents to an existing

file from the terminal.

* Working with grep command :

grep command to search for a particular string /word in text file.

1) grep options file.txt : matching string "options"

2) grep -i options file.txt : insentive string (like captial or small).

3) grep -n : matching string along with number.

4) grep -v : Results of the line not matching the string.

5) grep -c : number of lines in which result matched string.

* Working with sort command :

we use the sort command to sort the result of search either alphabetically or

numerically files.

files contents and directories can be sorted.

1) sort file.txt : sort the content of file.txt returns them in alphabetical order.

2) sort -r : reverse order

3) sort -f : insentive sorting.

4) sort -n : as per numerical order.

* working with '|' pipe command :

Result of one command as input to another command

'|' are used to perform two opration in the same command.

eg. $ grep ab file1.txt file2.txt |sort

(search ab string from both files and sort the result.)

* cp command : copy files and directories.

1) cp -r : Recursive for copying directories : copies even hidden files.(for folder)

2) cp -i : Enter interactive mode , CLI asks before over writing files.

3) cp -n : does not overwriting the files.

4) cp -u : updates the destination file only when source file is differnt from

destination file.

5) cp -v : verbose - print informative message.

* mv command : move files and directories.

1) mv -i : Enter the interactive mode ; CLI asks before overwriting files.

2) mv -u :update the destination file only when source file is differnt from

destination file.

3) cp -v : verbose - print informative message.

* mkdir command : create a new directory (folder)

1) mkdir -p : new parent directory and sub directories (folder into folder)

2) mkdir -p fil1 {f2,f3,f4}:- This is used to create multiple subdirectories inside the

new parent directory.

* rm command command : remove specified directory.

1) rm -p : Remove both parent and child directory.

2) rm -pv : Remove all the parent and sub directories along with verbose

(long-winded).

3)rm -r : Removes non empty directories.

4) rm -rp : Remove non empty directories including parent and subdirectories.

* change command :

1) chmod : To change the access permission of files and directories.

2) chown : To change the group ownership of file & Directories.

3) ch grep : to change the group of ownership of file and directories.

* env command (enviroment)

1) echo $home : see the path to your home directory.

2) echo $user : you should see your username.

3) env : the outputs of whole lots of information about the env variable.

eg. env --help , env --version.

* Paste command :

command similar to cat command ,it merge lines toghether in a file.

1) paste -s : all the lines into one file.

2) paste -d : something a little more readable everything should be on one line

delimited by space.

* head command :

head command will show you the firest 10 lines in a file.

eg. head -n15 : see the firest 15 lines instead the -n flag stands for number of lines.

* tail command :

similar to head command ,see the last 10 lines of a file by default.

1) tail -n10 : change the no of lines.

2) tail -f : This will follow the file as it grows.file will be contineslly changing while

interact with your system and using tail -f.

* Expand and unexpand :

The expand command copies file to the standard output ,with tab character,

Backspace character are preserved into the output and and decrement the coloumn

count for tab calculations.

1) expand sample.txt : The command will print output with each tab converted into

group of space.

2) for expand: expand sample.txt > Result.txt

for unexpand : unexpand -a Result .txt.

* Unique command :

using for parsing text and want to remove duplicates.

1) uniq -c : how many occurance of line.

2) uniq - u: lets get uniqn value.

3) uniq -d : get duplicate value.

* join and split :

allows you to join multiple files together.

* tr : Translate set of character into another set of character.

eg. hello - HELLO.

* wc&nl:

1) wc: (words count)totalno of counts words in file.

2) nl : check the count of lines on a file.

* vi :

If user add any content into file use sudo vi abc.txt

* sed : stream editor

sed helps in opration like selecting the text ,substituing text,modifying and original

file ,adding line to text or deleting the lines from text.

* Note : 1) If su command not working use sudo bash.

2) stdout redirection : >

3) stdin redirection : <

4) redirector of stdrr : <>

5) pipe oprator : | (use vertical bar)

* **OS Concept**

An operating system (OS) is the program that, after being initially loaded into the computer by a boot program, manages all of the other application programs in a computer.

1)Memory Management

The term Memory can be defined as a collection of data in a specific format. It is used to store instructions and processed data.

What is Main Memory:

Main memory is a repository of rapidly available information shared by the CPU and I/O devices.

Main memory is also known as RAM(Random Access Memory).

Main memory is associated with the processor, so moving instructions and information into and out of the processor is extremely fast.

What is Memory Management :

The task of subdividing the memory among different processes is called memory management.

Memory management is a method in the operating system to manage operations between main memory and disk during process execution.

Why Memory Management is required:

1.Allocate and de-allocate memory before and after process execution.

2.To keep track of used memory space by processes.

3.To minimize fragmentation issues.

4.To proper utilization of main memory.

5.To maintain data integrity while executing of process.

Logical and Physical Address Space:

1.Logical Address space: An address generated by the CPU is known as “Logical Address”. It is also known as a Virtual address. Logical address space can be defined as the size of the process. A logical address can be changed.

2.Physical Address space: An address seen by the memory unit (i.e the one loaded into the memory address register of the memory) is commonly known as a “Physical Address”. A Physical address is also known as a Real address. The set of all physical addresses corresponding to these logical addresses is known as Physical address space. A physical address is computed by MMU. The run-time mapping from virtual to physical addresses is done by a hardware device Memory Management Unit(MMU). The physical address always remains constant.

Static and Dynamic Loading:

To load a process into the main memory is done by a loader.

1.Static loading:- loading the entire program into a fixed address. It requires more memory space.

2.Dynamic loading:- The entire program and all data of a process must be in physical memory for the process to execute. So, the size of a process is limited to the size of physical memory. To gain proper memory utilization, dynamic loading is used.

Static and Dynamic linking:

To perform a linking task a linker is used. A linker is a program that takes one or more object files generated by a compiler and combines them into a single executable file.

1.Static linking: In static linking, the linker combines all necessary program modules into a single executable program.

2.Dynamic linking: The basic concept of dynamic linking is similar to dynamic loading. In dynamic linking, “Stub” is included for each appropriate library routine reference.

Swapping :

When a process is executed it must have resided in memory. Swapping is a process of swap a process temporarily into a secondary memory from the main memory, which is fast as compared to secondary memory. A swapping allows more processes to be run and can be fit into memory at one time.

Contiguous Memory Allocation :

The main memory should oblige both the operating system and the different client processes. Therefore, the allocation of memory becomes an important task in the operating system.

Memory allocation:

To gain proper memory utilization, memory allocation must be allocated efficient manner. One of the simplest methods for allocating memory is to divide memory into several fixed-sized partitions and each partition contains exactly one process.

Fragmentation:

A Fragmentation is defined as when the process is loaded and removed after execution from memory, it creates a small free hole. These holes can not be assigned to new processes because holes are not combined or do not fulfill the memory requirement of the process.

1)Internal fragmentation:

Internal fragmentation occurs when memory blocks are allocated to the process more than their requested size. Due to this some unused space is leftover and creates an internal fragmentation problem.

2)External fragmentation:

In external fragmentation, we have a free memory block, but we can not assign it to process because blocks are not contiguous.

Paging:

Paging is a memory management scheme that eliminates the need for contiguous allocation of physical memory. This scheme permits the physical address space of a process to be non-contiguous.

2) Processes

A process is defined as an entity which represents the basic unit of work to be implemented in the system.

When a program is loaded into the memory and it becomes a process, it can be divided into four sections ─ stack, heap, text and data.

Stack : The process Stack contains the temporary data such as method/function parameters, return address and local variables.

Heap : This is dynamically allocated memory to a process during its run time.

Text : This includes the current activity represented by the value of Program Counter and the contents of the processor's registers.

Data : This section contains the global and static variables.

Process Life Cycle :

When a process executes, it passes through different states. These stages may differ in different operating systems, and the names of these states are also not standardized.

Process Control Block (PCB) : A Process Control Block is a data structure maintained by the Operating System for every process. The PCB is identified by an integer process ID (PID).

1. Process State: The current state of the process i.e., whether it is ready, running, waiting, or whatever.

2. Process privileges: This is required to allow/disallow access to system resources.

3. Process ID: Unique identification for each of the process in the operating system.

4.Pointer : A pointer to parent process.

5. Program Counter: Program Counter is a pointer to the address of the next instruction to be executed for this process.

6. CPU registers: Various CPU registers where process need to be stored for execution for running state.

7. CPU Scheduling Information: Process priority and other scheduling information which is required to schedule the process.

8. Memory management information: This includes the information of page table, memory limits, Segment table depending on memory used by the operating system.

9.Accounting information: This includes the amount of CPU used for process execution, time limits, execution ID etc.

10.IO status information: This includes a list of I/O devices allocated to the process.

3) Kernel

The kernel is the first program that is loaded after the boot loader whenever we start a system. The Kernel is present in the memory until the Operating System is shut-down.

Kernel provides an interface between the user and the hardware components of the system. When a process makes a request to the Kernel, then it is called System Call.

Functions :

1.Process management

2.Access computer resources

3.Device management

4.Memory management

5.Interrupt handling

6.I/O communication

7.File system...etc.

Types of Kernel :

1.Monolithic Kernels:

In monolithic Kernels both user services and the kernel services are implemented in the same memory space. By doing this, the size of the Kernel is increased and at the same time it increases the size of the Operating System. As there is no separate User Space and Kernel Space, so the execution of the process will be faster in Monolithic Kernels.

2. Microkernel :

A Microkernel is not the same as the Monolithic kernel. It is a little bit different because in a Microkernel, the user services and kernel services are implemented into different spaces. Because of using User Space and Kernel Space separately, it reduces the size of the Kernel and in turn, reduces the size of the Operating System.

As we are using different spaces for user and kernel service, the communication between application and services is done with the help of message parsing because of this it reduces the speed of execution.

3.Nanokernel :

The name suggests the complete code of the kernel is very small, which means the code executing in the privileged mode of the hardware is very small. The term nanokernel is used to explain that the kernel supports a nanosecond clock resolution.

4.Exokernel :

the resource protection is separated from the management which results in allowing us to perform applicationspecific customization.

there is no force abstraction and this is the feature that makes it different from a Monolithic Kernel and Microkernel. But the drawback of this is the complex design. The design of the Exokernel is very complex.

4) Virtualization

Virtualization in operating system changes a normal operating system so that it can run different types of applications that may be handled on a single computer system by many users.

The operating system may appear different to each user and each of them may believe they are interacting with the only operating system i.e. this does not interfere with user experience.

Advantages of Virtualization :

1.Virtualization allows a finite number of hardware resources to be easily distributed to multiple processes that require them.

2.Improved security can be obtained by using virtualisation. This happens because each process inhabits its own instance of the operating system and works independently.

3.Operating system virtualization is very useful for establishing a virtual hosting environment.

4.There is only a little overhead involved for operating system virtualization and so it is very beneficial.

Disadvantages of Virtualization :

1.Specialized experts are required to implement and manage a virtualized system. This results in need for virtualization experts and increased costs.

2.There are many upfront costs involved in virtualization. These include the cost for virtualization software as well as the additional hardware required. The costs also depends on the existing system network.

* Networking Protocols

1) Dynamic Host Configuration Protocol (DHCP) :

Dynamic Host Configuration Protocol (DHCP) is a network management protocol used to dynamically assign an IP address to nay device, or node, on a network so they can communicate using IP (Internet Protocol). DHCP automates and centrally manages these configurations. There is no need to manually assign IP addresses to new devices. Therefore, there is no requirement for any user configuration to connect to a DHCP based network.

DHCP can be implemented on local networks as well as large enterprise networks. DHCP is the default protocol used by the most routers and networking equipment. DHCP is also called RFC (Request for comments)

DHCP does the following:

1.DHCP manages the provision of all the nodes or devices added or dropped from the network.

2.DHCP maintains the unique IP address of the host using a DHCP server.

3.It sends a request to the DHCP server whenever a client/node/device, which is configured to work with DHCP, connects to a network. The server acknowledges by providing an IP address to the client/node/device.

How DHCP works:

1.DHCP runs at the application layer of the TCP/IP protocol stack to dynamically assign IP addresses to DHCP clients/nodes and to allocate TCP/IP configuration information to the DHCP clients. Information includes subnet mask information, default gateway, IP addresses and domain name system addresses.

2.DHCP is based on client-server protocol in which servers manage a pool of unique IP addresses, as well as information about client configuration parameters, and assign addresses out of those address pools.

Components of DHCP

1.HCP Server:

DHCP server is a networked device running the DCHP service that holds IP addresses and related configuration information. This is typically a server or a router but could be anything that acts as a host, such as an SD-WAN appliance.

2.DHCP client:

DHCP client is the endpoint that receives configuration information from a DHCP server. This can be any device like computer, laptop, IoT endpoint or anything else that requires connectivity to the network. Most of the devices are configured to receive DHCP information by default.

3.IP address pool:

IP address pool is the range of addresses that are available to DHCP clients. IP addresses are typically handed out sequentially from lowest to the highest.

4.Subnet:

Subnet is the partitioned segments of the IP networks. Subnet is used to keep networks manageable.

5.Lease:

Lease is the length of time for which a DHCP client holds the IP address information. When a lease expires, the client has to renew it.

6.DHCP relay:

A host or router that listens for client messages being broadcast on that network and then forwards them to a configured server. The server then sends responses back to the relay agent that passes them along to the client. DHCP relay can be used to centralize DHCP servers instead of having a server on each subnet.

Benefits of DHCP

1.Centralized administration of IP configuration.

2.Dynamic host configuration.

3.Seamless IP host configuration.

4.Flexibility and scalability.

2) User Datagram Protocol (UDP) :

User Datagram Protocol (UDP) is a Transport Layer protocol. UDP is a part of the Internet Protocol suite, referred to as UDP/IP suite. Unlike TCP, it is an unreliable and connectionless protocol. So, there is no need to establish a connection prior to data transfer.

UDP Header :

UDP header is an 8-bytes fixed and simple header, while for TCP it may vary from 20 bytes to 60 bytes. The first 8 Bytes contains all necessary header information and the remaining part consist of data. UDP port number fields are each 16 bits long, therefore the range for port numbers is defined from 0 to 65535; port number 0 is reserved. Port numbers help to distinguish different user requests or processes.

1.Source Port: Source Port is a 2 Byte long field used to identify the port number of the source.

2.Destination Port: It is a 2 Byte long field, used to identify the port of the destined packet.

3.Length: Length is the length of UDP including the header and the data. It is a 16-bits field.

4.Checksum: Checksum is 2 Bytes long field. It is the 16-bit one’s complement of the one’s complement sum of the UDP header, the pseudo-header of information from the IP header, and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets.

Applications of UDP:

1.Used for simple request-response communication when the size of data is less and hence there is lesser concern about flow and error control.

2.It is a suitable protocol for multicasting as UDP supports packet switching.

3.UDP is used for some routing update protocols like RIP(Routing Information Protocol).

4.Normally used for real-time applications which can not tolerate uneven delays between sections of a received message.

The application layer can do some of the tasks through UDP-

1.Trace Route

2.Record Route

3.Timestamp

3) Transmission Control Protocol (TCP) :

TCP (Transmission Control Protocol) is one of the main protocols of the Internet protocol suite. It lies between the Application and Network Layers which are used in providing reliable delivery services. It is a connection-oriented protocol for communications that helps in the exchange of messages between the different devices over a network.

Working of TCP :

To make sure that each message reaches its target location intact, the TCP/IP model breaks down the data into small bundles and afterward reassembles the bundles into the original message on the opposite end. Sending the information in little bundles of information makes it simpler to maintain efficiency as opposed to sending everything in one go.

Features of TCP/IP

Some of the most prominent features of Transmission control protocol are

1. Segment Numbering System

TCP keeps track of the segments being transmitted or being received by assigning numbers to each and every single one of them.

A specific Byte Number is assigned to data bytes that are to be transferred while segments are assigned sequence numbers.

Acknowledgment Numbers are assigned to received segments.

2. Flow Control

Flow control limits the rate at which a sender transfers data. This is done to ensure reliable delivery.

The receiver continually hints the sender on how much data can be received (using a sliding window)

3. Error Control

TCP implements an error control mechanism for reliable data transfer

Error control is byte-oriented

Segments are checked for error detection

Error Control includes – Corrupted Segment & Lost Segment Management, Out-of-order segments, Duplicate segments, etc.

4. Congestion Control

TCP takes into account the level of congestion in the network

Congestion level is determined by the amount of data sent by a sender

Advantages

1.It is a reliable protocol

2.It provides an error-checking mechanism as well as one for recovery

3.It gives flow control

4.It makes sure that the data reaches the proper destination in the exact order that it was sent

5.Open Protocol, not owned by any organization or individual

6.It assigns an IP address to each computer on the network and a domain name to each site thus making each device site to be distinguishable over the network.

Disadvantages

1.TCP is made for Wide Area Networks, thus its size can become an issue for small networks with low resources

2.TCP runs several layers so it can slow down the speed of the network

3.It is not generic in nature. Meaning, it cannot represent any protocol stack other than the TCP/IP suite. E.g., it cannot work with a Bluetooth connection.

4.No modifications since their development around 30 years ago.

4) The Hypertext Transfer Protocol (HTTP

Basically, HTTP is a TCP/IP based communication protocol, that is used to deliver data (HTML files, image files, query results, etc.) on the World Wide Web. The default port is TCP 80, but other ports can be used as well. It provides a standardized way for computers to communicate with each other. HTTP specification specifies how clients' request data will be constructed and sent to the server, and how the servers respond to these requests.

Basic Features :

1) HTTP is connectionless.

2) HTTP is media independent.

3) HTTP is stateless.

Basic Architecture :

The HTTP protocol is a request/response protocol based on the client/server based architecture where web browsers, robots and search engines, etc. act like HTTP clients, and the Web server acts as a server.

Client :

The HTTP client sends a request to the server in the form of a request method, URI, and protocol version, followed by a MIME-like message containing request modifiers, client information, and possible body content over a TCP/IP connection.

Server :

The HTTP server responds with a status line, including the message's protocol version and a success or error code, followed by a MIME-like message containing server information, entity meta information, and possible entity-body content.

5) Secure Shell (SSH)

SSH stands for Secure Shell or Secure Socket Shell. It is a cryptographic network protocol that allows two computers to communicate and share the data over an insecure network such as the internet. It is used to login to a remote server to execute commands and data transfer from one machine to another machine.

How does SSH Works?

The SSH protocol works in a client-server model, which means it connects a secure shell client application (End where the session is displayed) with the SSH server (End where session executes).

The basic use of SSH is to connect a remote system for a terminal session and to do this, following command is used: ssh UserName@SSHserver.test.com

The architecture of SSH Protocol

1) Transport Layer

2) User-authentication layer

3) Connection Layer

The SSH protocol can transfer the following:

1) Data

2) Text

3) Commands

4) Files

* **Git Concept**
* Git : Version control System
* Features : 1) Easliy recover files.

2) roll back previosly working state.

3) almost every opration is local.

4) Git itegrity.

* Git Installation :

1) git install - git-scm.com - for window

2) Launch git bash

3) set username and email

$ git config --global username "Ashok"

$ git config --global user.email "Ashok@gmail.com"

5) view username command: $git config user.name

* Git -- three stage architecture
* Tracking Project :

1) check status : $ git status

2) inatlise git repository : $ git init

3) staging area : $git add --a (Track files)

4) Commit File : $git commit -m "msg"

5) view to all commit : $ git log

6) Detect content of git repository : $ rm - rf. git

7) $ git add . : add changes in the working directories to staging area.

6) $ git add <filename> : Respective file gone into staging area.

* Cloning a remote repository :

1. for recieving repository : $git clone <paste link>

* File status life cycle :
* Git ignore :

Ignoring Files in git

1. create one file as a .gitignore : $ touch .gitignore

2. Put files and folder name in gitignore.

* Git differnce : commits and staging area.

1. $git diff : showing differnce between staging area and working directory.

* Skipping the staging area :

$ git commit -a -m "direct commit"

Direct commit all files.

* move and rename files :

1. Remove : $ git rm <filename>

2.Rename : $ git mv <firest.txt> <firest\_Renamed.txt>

3.untracked file start : $ git rm --cached <filename>

* git view :

viewing and changing commits in git.

1. $ git log - : Showing all commits Author

2. $ git log -p : which line remove or added showing change also.

3.$ git log -p -3 : Showing 3 commits with changes.

4. $ git log --start : showing statitics like insertion ,changed and deletion.

5. $ git log --pretty=one line : all commits in one line.

6. $ git --pretty=short : showing Author and commit only.

7. $ git --pretty=full : all commits

8. $ git log --since=2.days : showing last 2 days commit.

9. $ git log --pretty=format : "v.h -- %an" : google search

* unstaging & unmodifying files :

1.unstaging files : $ git restore --staged <file name>

2.unmodifying &Restore file for back commit : $ git checkout --<filename>

3. $git checkout -f : for going to back commit.

* Remote Repository : -

1) SSH key add : search google

2) $git remote add origin git@github.com

git remote -v

3) For push to any repository :

$git push -U origin master

* Setting alias in git

Alias command : Shorcut that refrence command or Shortcut command

Step 1: $git config --global alias.ci commit

Step 2 : $git ci

eg. for last commit :- Step 1: $git config --global alias.last log -p -1

Step 2 : $ git last

* Creating & Switching Branches :

1.new branch create : $git checkout -b <branch name>

2. Switch branch : $git checkout master

$git checkout <branch name>

3.Show branch : $git branch

4.Delete branch : $git branch -- delete <branch name >

* Branching & Merging

1.change branch : $git checkout <branch1>

2.Merge : step 1: $git merge branch2

step 2: $git status

step 3: $git add .

step 4 : $git commit -m

* Merge conflicts

A merge conflict is an event that takes place when Git is unable to automatically resolve differences in code between two commits.

Use merge command and commit.

* branch management

The git branch command lets you create, list, rename, and delete branches.

1. $ git branch -v : showing details of branches.

2. git branch --merged : showing already maeged branches.

3. git branch -- no -merged : showing not merged branches.

4. deleting branches :

step 1 : git branch -d <file name> give error if branch is not merged.

step 2 : git branch -D <file name> no error and branch gets deleted.

Branching overflow:

* Pushing branch into remote repositires

1. push command : $git push origin <branch name>

2.check reomte : $git remote

$git remote -v

* Make new repository in remote :

1.command : $git remote add origin <link>

2. check remote : git remote

git remote -v

3.Push to remote : $ git push -u origin <branch name>

4. Login

* **Bash Scripting**

Bash is a command-line interpreter or Unix Shell and it is widely used in

GNU/Linux Operating System.

* Applications of Bash Scripts:

1) Manipulating files.

2) Executing routine tasks like Backup operation.

3) Automation.

* Advantages of Bash Scripts:

1) It is simple.

2) helps to avoid doing repetitive tasks

3) Easy to use

4) Frequently performed tasks can be automated

5) A sequence of commands can be run as a single command.

* Disadvantages of Bash Scripts:

1) Any mistake while writing can be costly.

2) A new process launched for almost every shell command executed.

3) Slow execution speed.

4) Compatibility problems between different platforms.

* How to Write Bash Scripts?

1) First, we will create a file with the .sh extension.

2) Next, we will write down the bash scripts within it.

3) After that, we will provide execution permission to it.

- To create and write a file with the .sh extension we can use gedit text editor. The

command for it will be –gedit scriptname.sh

-The first line of our script file will be

–#!/bin/bash

-Let’s write down just a simple script that will print some lines in the terminal. The

code for it will be –

#!/bin/bash

echo "Hello World"

- Now we will save the file and provide the execution permission to it. To do so use

the following command – chmod +x scriptname.sh

- Next to execute the following script we will use the following command –

./scriptname.sh

* Variables:

Example Script:

Name="Ashok"

Age=24

echo "The name is $Name and Age is $Age"

Output of Variables: The name is Ashok and Age is 24.

- So, here is have declared two variables Name and another one is Age.

These variables are accessible using $Name and $Age. That means, we can

declare a variable in a bash script using VariableName=Value and can access it

using $VariableName.

* Global variable &Local variable:

1) declared inside a function then it is generally a local variable and if it is declared outside then it is a global variable.

Note : In the case of a bash script, this concept is a little bit different, here any variable whether it is written inside a function or outside a function by default is a global variable. If we want to make a local variable then we need to use the keyword “local”.

Example Script:

#!/bin/bash

var1="Apple" #global variable

myfun(){

local var2="Banana" #local variable

var3="Cherry" #global variable

echo "The name of first fruit is $var1"

echo "The name of second fruit is $var2"

}

myfun #calling function

echo "The name of first fruit is $var1"

#trying to access local variable

echo "The name of second fruit is $var2"

echo "The name of third fruit is $var3"

Output of local and global variables :

The name of first fruit is Apple

The name of second fruit is Banana

The name of first fruit is Apple

The name of second fruit is

The name of third fruit is Cherry

Here in this above example, var2 is a local variable, so when we are accessing it from the function it is doing fine but when we are trying to access it outside the function, it is giving us an empty result in the output.

* Input and Output: A script can take one or more inputs and can also produce zero or many outputs.

Example Script:

echo "Enter filename"

read filename

if [ -e $filename ]

then

echo "$filename is exits on the directory"

cat $filename

else

cat > $filename

echo "File created"

fi

Output of Input & Output:

First time:

Enter filename

File1.txt

Hello world

File created

Second time:

Enter filename

File1.txt

File1is exits on the directory

Hello world

So, in this above example the first time, the script could not find any file with that file name, and the else block gets executed. It created the file and put some data into that file. When we run it a second time with the same file name, then it finds the file. So, is the if block gets executed and that displays the contents of the file. Reading the file contents is input and on the first time putting data into the file is considered to be output.

we have used > for storing the content in a file.

The > notation is used to redirect stdout to a file. On the other hand, we can use 2> notation to redirect stderr, and &> to redirect both stdout and stderr.

* Functions

function is a block of code that performs some tasks and it can be called multiple times for performing tasks.

Syntax of Functions:

#for defining

function\_name(){

commands

.....

}

function\_name # for calling

Example Script:

#!/bin/bash

#It is a function

myFunction () {

echo Hello World

}

#function call

myFunction

Output of Functions: Hello World

* Decision Making

1) If-else statement:

It can be used to execute two different codes based on whether the given condition is satisfied or not.

There are a couple of varieties present within the if-else statement. They are :

a) if-fi

b) if-else-fi

c) if-elif-else-fi

d) nested if-else

Syntax of If-else statement:

if [ expression ]; then

statements

fi

Example Script:

Name="Ashok"

if [ "$Name" = "Ashok" ]; then

echo "His name is Ashok. It is true."

fi

Output of if-else statement: His name is Ashok. It is true.

2) case-sac statement:

case-sac is basically working the same as switch statement in programming.

Syntax of If-else statement:

case $var in

Pattern 1) Statement 1;;

Pattern n) Statement n;;

esac

Example Script:

Name="Ashok"

case "$Name" in

#case 1

"Ashok") echo "Profession : Software Engineer" ;;

#case 2

"Vikas") echo "Profession : Web Developer" ;;

#case 3

"Akshay") echo "Profession : Technical Content Writer" ;;

esac

Output of if-else statement: Profession : Software Engineer

* String and Numeric Comparisons

1) == Returns true if the strings are equal

2) != Returns true if the strings are not equal

3) -n Returns true if the string to be tested is not null

4) -z Returns true if the string to be tested is null

5) -eq Equal

6) -ge Greater Than or Equal

7) -gt Greater Than

8) -le Less Than or Equal

9) -lt Less Than

10) -ne Not Equal

Example Script:

if [ 10 -eq 10 ];then

echo "Equal"

fi

if [ 'Ashok' == 'Ashok' ];

then

echo "same" #output

else

echo "not same"

fi

Output of if-else statement:

Equal

same

* Functions with Passing Arguments

We can add arguments or parameters to a function and pass data using it to the function so that the function can act with those data.

Syntax of Functions with Passing Arguments:

#for defining

function\_name(){

.....

parameter\_1 = $1

parameter\_2 = $2

.

.

.

parameter\_n = $n

....

commands

.....

}

#for calling

function\_name p1 p2 ....pn

Example of Functions with Passing Arguments:

#!/bin/bash

add\_two\_num(){

local sum=$(($1+$2))

echo sum of $1 and $2 is $sum

}

add\_two\_num 2 3

Output of Functions with Passing Arguments: sum of 2 and 3 is 5

Above is a script for adding two numbers. Here we have provided 2 and 3 as arguments. We have accessed them using $1 and $2 from the function and calculated their sum and printed it to the terminal.

* Functions with Return Values

A return value is produced and returned to the calling method by a function after it completes its execution.

A return value can be used to share the produced result or some status code about whether one function was successfully executed or not.

1)Example of Functions with Return Values:

#!/bin/bash

myfun(){

return 7

}

myfun

echo The return value is $?

1)Output of Functions with Return Values: The return value is 7

2)Example of Functions with Return Values:

#!/bin/bash

myfun(){

return $(($1+$2))

}

add\_two\_num 2 3

echo The sum is $?

2)Output of Functions with Return Values: The sum is 5

* Variable Scope :

Scope in a program or script is a region where the variables have their existence.

* Overriding Commands

It is possible to have a function with the same name as that of a command. It is helpful if we want to run a command with specific options or want to have a customized edition of it.

Example of Overriding Commands:

#!/bin/bash

#overriding command

echo(){

builtin echo "The name is : $1"

}

echo "Ashok"

Output of Overriding Commands: The name is : "Ashok"

Here in this example, we have overridden the echo command. Now before printing the given string, it appends the string with some more words and then prints.