BASH Shell

other shells are :- bourne, Korn, C, TC

BASH - bourne again shell available by default in most of the shell

Features of a shell

(i) Process control
(ii) create variables,
(iii) flow control (if-else, loop)
(iv) create functions
(v) file & command name completion (by pressing tab)
(vi) Command line editing
(we can edit the name of the command if written wrongly)
(vii) Command history

Shell executes commands in interactive and non-interactive mode.

Interactive mode means we directly write the command in the shell.

non-interactive mode means we execute the commands in using shell script or write in a file and executing of the file.

ex. $ cat > file1

date
pwd
ls

$ /bin/bash file1

ot → will execute all the above commands.

Internal command : those commands whose code is part of shell itself. No child process is executed. e.g. cd, echo, pwd, exit, type, alias, fg, unset, help

External commands are those whose code are written in binary file.

Some of the commands are implemented as both internal and external. e.g. pwd, cd echo, exit

$ type -a pwd

pwd is a shell builtin

Pwd is /bin/Pwd

=> to execute multiple commands ; can be used

$ sleep 5 ; echo "hi"

If & is used after a command then shell will not wait for the finish of the command, so, the command will execute in the background.

$ sleep 10 &

$ ls

when sleep 10 is finished, it will send a notification

=> $ command1 && command2

If command1 is successful then command2 will execute.

$ grep root /etc/passwd && echo "Found"

O/P Found.

$ grep daniel /etc/passwd && echo "Found"

O/P - no O/P.

$ command1 || command2

If command1 is success then command2 will not execute.

$ echo $?

If previous command is a success then it will return 0.

$ rm xyz-txt

$ echo $?

O/P 1 as previous command was failure.

is used to comment

$ ls # this is a comment.

$ echo hello & world

o/p hellow world

By using \, & is not interpreted as command and

$ echo this is a command \
which is split \
in parts

o/p :- this is a command which is split in parts.

=> user defined variables

$ name = "JNV"

$ echo $name

o/p JNV

$ n1 = 131

$ echo $n1

o/p 131

$ unset name

$ echo $name

o/p : no o/p

$ uname = (Jawahar lal nehru)

$ echo ${uname[0]}

o/p - Jawahar

$ echo ${uname[1]}

o/p - nehru

$ read var1
Jawahar

$ echo var1
Jawahar

$ read n1 n2 n3
Jawahar lal nehru university

$ echo $n1
Jawahar

$ echo $n2
lal

$ echo $n3
nehru university

:) env :- environment variables are used to configure the shell itself.

$ env
g.t will show all env variables

$ set

$ echo $PATH
this will give absolute path of all directories in which the shell will look for a executable file of a command.

$ echo $PWD
Present working directory

$ echo $CDPATH
It will give the name of the directory on which CD command will go. If here, the o/p is empty, it means it will see only in present directory to go to the requested directory.

$ echo $PS1
It will give the name of

$ PS1 = "JMW:-"
JMW:-

$ history

$ history 10
last 10 commands runned

$!2004

$ echo $ HISTSIZE
o/p 1000

$ echo $HISTFILE
/home/mukesh/.bash_history

If a command is type after a space then this command will not be recorded in history.

$ pwd

Date:

=) ls file.*

It will ~~access~~ list all files in present directory ~~whose file~~ which have any character after dot.

=) ls f0-- c

=) ls file?

It will display all files whose 1st 4 characters are file and ~~first~~ fifth character can be anything. Also, file name will be exactly 5.

=) ls file??

Flow chart of working of a shell