Notes :

CENTURYPLY®

Ubuntu Linux Operating System

Qs) What is shell?

Ans- A UNIX shell program interprets user commands which are either directly entered by the user or which can be read from a file called shell script or shell program.

Shell scripts are interpreted, not compiled.

When you write a shell script, it interpreted about your operating system and you don't need to compile your shell script in order to execute it.

In your editor, you can write shell script and you awake or execute it without compiling.

Different types of shells are there:-

- We will be learning bash shell scripting.

CENTURYPLY®

Notes :

Qs) What shall types your OS support.

cat /etc/shells.

→ /bin/sh
/bin/dash
~~/bin/bash~~
~~/usr~~ /usr/bin/bash

sh stands for bourne shell, which is original shell used on UNIX system or UNIX-like environment.

Bash stands for bourne again shell. It is standard GNU shell, which is intuitive and flexible.

which bash → to see where bash is located

Notes :

∞ → Shell script starts
∞ → " " ends
∞ → " " continues

§ ∞ → terminal

L2 Open bash file which you created, in VS Code

#! refers to hashbang

∞ #! /bin/bash → 1st line of script

echo "Hello World"

∞

§ ls -al

§ chmod +x hello.sh → to give execute permissions

∞ # this is a comment

Theory :-

System Variables → maintained by LINUX or UNIX operating system

Some variables which are capital most likely they are system variables.

∞ echo $BASH ↵
/bin/bash

System Variable

∞ echo $BASH_VERSION ↵
5.0.17(1) — release

∞ echo $PWD ↵
/mnt/c/practise-linux

∞ echo $HOME ↵
/home/akash-kinkar-pandey

∞ name = Mark (no space before =
(no space after =)

variable

∞ echo the name is $name
$ name

∞ echo "the name is $name"

∞ VALUE = 10
$VALUE

L3 Read user input

∞ echo "Center name:"
read name
echo "Centered name: $name"

Center name
Akash
Centered name: Akash

Input in same line

∞ read -p 'username: ' user_var

echo "username: $user_var"

→ While taking input, nothing is displayed

∞ read (-sp) 'password: ' pass

echo "password: $pass"

→ Take many inputs

∞ read (-a) names

echo "Names: ${names[0]}"

∞ read

echo $REPLY

(If you don't write anything input is stored in inbuilt variable REPLY

Pass Arguments

∞ echo $1 $2 $3 'hi'

./yoo.sh hello akash pandey

∞ echo $0

→ Prints file name

Take all command line arguments into an array format

```
args=("$@")
echo ${args[0]} ${args[1]} ${args[2]}
echo $@   ← Prints array or all command line arguments
echo $#   ← Prints number of cmd line arguments
```

L5 if statements

```
count=10   (no spaces here)
if [ $count -eq 9 ]
then
    echo "condition is true"
fi
```

-ne ~~to~~ not equal to

```
~~if [ $count > 9 ]~~
if (( $count > 9 ))
then
    echo "cond is true"
fi
```

∞ word = abc → single = works

```
if [ $word == "abc" ]
then
  echo "hi"
fi
```

for String :~

= , == , != , < , >

If using angular(<,>) brackets use [[ ]] outside in case of strings.

∞ word = a

```
if [[ $word < "b" ]]
then
  echo "hi
else
  echo "hello"
then
fi
```

```
if [[ $word == "c" ]]
then
echo "b is true"
elif [[ $word == "a" ]]
then
echo "a is true"
else
echo "hello"
fi
```

L6 File Test Operators

```
echo -e "enter name of file \c"
read filename
if [ -e $filename ]
then
echo "file found"
else
echo "file not found"
fi
```

-e → file exists or not

-f → check if file exists and if it regular file or not

-d → ~~for~~ check if directory exists or not

-s → checks if file not empty

```
if [ -s $filename ]
then
echo "file not empty"
else
echo "file empty"
fi
```

returns true if file is not empty

-b → check if block file or not
-c → " " character " " "

-r → check if file has read permission
-w → " " " " write "
-x → " " " " execute "

Block special

Character special file → contains text or data
Block special file → Binary special file → picture or video

L7 Append at end of file

```
echo -e "enter name of file"
read filename
if [ -e $filename ]
then
    if [ -w $filename ]
    then
    echo "type some data. Press Ctrl D to exit"
    cat >> $filename
    else
    echo "file do not have write permission"
    fi
else
    echo "$filename file does not exist"
fi
```

Logical AND

```
age=25
if ["$age" -gt 18] && ["$age" -lt 30]
then
    echo "valid age"
else
    echo "not valid age"
fi
```

```
age=15
if ((($age > 14 || $age < 15)))
then
  echo "Hi"
else
  echo "hello"
fi
```

OR

```
age=25
if [ "$age" -eq 18 -o "$age" -eq 30 ]
then
    echo "valid age"
else
    echo "invalid"
fi
```

```
if [[ "$age" -eq 18 || "$age" -eq 30 ]]
```

L10

```
a=1
b=3
echo $((a+b))
echo $((a*b))
echo $(expr $num1 + $num2)
echo $(expr $num1 \* $num2)
```

L11

```
echo "5 + 20.2" | bc          → 25.2
echo "scale=2 24/5" | bc      → 4.8
echo "24/5" | bc              → 4
```

```
a=1
b=3.2
echo "scale=2; $a+$b" | bc    → 4.2
echo "scale=2; sqrt($num)" | bc -l
                              → 5.19
```

```
echo "scale = 2; 3^3" | bc -l
```

→ 27.00

L12 Switch Case

```
vehicle = $1

case $vehicle in
    "car" )
    echo "Carrr" ;;
    "van" )
    echo "van" ;;
    "bicycle" )
    echo "bicycle" ;;
    "truck"
    echo "truckk" ;;
    * )
    echo "unknown vehicle" ;;
esac
```

L13

```
echo -e "Enter some character"
read value

case $value in
    [a-z])
        echo "Small letters..." ;;
    [A-Z])
        echo "Capital letters..." ;;
    [0-9])
        echo "Digits" ;;
    ?)
        echo "Special character" ;;
    *)
        echo "Unknown $value" ;;
esac
```

Search Regular expression Patterns wikipedia

DO LANG=C if wrong output comes for uppercase letters

14 Arrays

os=('ubuntu' 'windows' 'kali')

echo "${os[@]}" → full array

echo "${os[1]}" → windows

echo "${!os[@]}"

→ 0 1 2

echo "${#os[@]}"

→ 3 (Array length)

os[3]='mac' # you can added at 6th index

echo "${os[3]}"

unset os[2] → delete index's element

∞ string = abcdefghi

echo = "${string[0]}" → abcdefghi

echo = "${string[1]}" → abcdefghi

echo = "${#string[@]}" → 1

L15 while loop

```
n=1
while (($n<10))
do
  echo $n
  n=$((n+1))
done
```

((n++)) → increments

L16

sleep 1 ← wait for 1 second

gnome-terminal & ← to open terminal

L17 Read a file

```
while read p
do
  echo $p
done < hello.sh
```

```
cat hello.sh | while read up
do
  echo $p
done
```

```
ls /etc
```

IFS part skipped

L18 until loop

```
n=1
until [ $n -ge 10 ]
do
    echo $n
    n=$((n+1))
done
```

(until (($n>10)))

```
1
2
3
4
5
6
7
8
9
```

L19

```
for i in 1 2 3 4 5
do
  echo $i
done

for i in {1..10}
do
  echo $i
done

for i in {1..10..2}   # increment
:
:

echo ${BASH_VERSION}

for ((i=0; i<5; i++))
do
    echo $i
done
```

L20

```
for i in `pwd date`
do
    $i
done
```

```
for cmd in * → iterate on all files, directories
do
    if [ -d $item ] → only directories printed
    then
        echo $item
    fi
done
```

L21

```
select name in hey ho
do
    echo "$name selected"
done
```

1) hey
2) ho

} menu structure

#? ← write 2

2 selected

```
select name in mark john tom ben
do
    case $name in
    mark)
        echo mark selected;;
    john)
        echo john selected;;
    a) ~~wrong~~
        echo Wrong;;
    esac
done
```

L22 Break

```
for(( ))
do
    if [ $i -gt 5 ]
    then
        break
    fi
done
```

} inside for loop.

Continue

```
if [ $i -eq 3 -o $i -eq 6 ]
then
    continue
fi
```

L23 functions

```
function Hello() {
    echo "Hello"
}

quit() { exit }

quit        # Call functions
Hello
```

```
function print () {
    echo $1 $2 }
~~echo "foo"~~
```

print Hello hey

L 24

```
function print() {
local name = $1
    echo " the name is $name"
                }
name = "Tom"
echo "The name is $name: Before"
print Man
echo " The name is $name: After"
```

↳ Tome

↳ Man

(4) we didn't write local, Tom would be printed)