Part 3

→ WSL

→ Ubuntu

→ Visual studio code

Extentions

→ WSL

ubuntu cmd

→ code .

visual studio opens

cat /etc/shells

→ to view the shells available in the machine.

main.sh

```
#! /bin/bash
pwd
ls -l
```

→ chmod +x main.sh

→ sh main.sh

→ /home/mithran

→ main.sh

Giving all required commands in a file as above and executing it in one is shell scripting

```
main.sh
#! /bin/bash
echo "hello world"
```

→ ./main.sh (alternate of sh main.sh)

```
hello world
```

```
main.sh
#! /bin/bash
 a=10
 b=20
 c=30
 echo $a $b $c
```

→ ./main.sh

```
10 20 30
```

Rules for creating a variable

→ first letter should be a alphab
or underscore.

→ General form

→ Variable is linux are in form of
all all capital. (i.e A, A.B, VAR
can be in smalls also)

main.sh

```
#! /bin/bash

VAR_A =10
VAR_C =" Hello world"
echo $VAR_A $VAR_C
```

```
→ 10    Hello world
```

Variable types

→ local
→ System (shell variables)
→ Env variable

main.sh

```
#! /bin/bash

echo $AGE    # Local Variable
```

→ export AGE=26

O/P no output

main.sh

```
#! /bin/bash

echo $AGE # Env Variable.
```

O/P

26 (pick value from Env variable)

main.sh

```
echo $AGE    # Env variable

AGE = 50

echo $AGE  # local variable
```

(as value is readily available in local file it takes value from above AGE)

when local assignment is not available it goes check Env assigned variable.

O/P

```
26

50
```

~~Sh~~ main.sh

```
#! /bin/bash

echo $Home
echo $USER
echo $PWD
```

} System Variable (shell variables)

O/P

```
/home/mithran

mithran

/opt
```

main.sh

```
#!/bin/bash
echo "what is your Name"
read Myname
echo "hey hi, $Myname"
```

O/P ./main.sh

```
what is your Name
Hema (gets input from user)
hey hi, Hema.
```

read -p "what is your Name ?" Myname

O/P

what is your Name ? Hema

(no need of echo command)

Special Variables # Check (41:12) in 3rd viedo ↳ for code

$0 → filename

$1 → 1st Input Argument

$2 → 2nd " "

$# → tells the number of input arguments passed

$*, $@ → Show all passed i/p arguments

$? → ~~Ase~~ shows the exit status of the last command executed

O/P
0 → Success

$$ → telles the process id of the current shell

$! → process number of the last run command.

$\boxed{\text{Operators}}$

a = 10

b = 20

$a → value

$b → value

```
echo `expr $a + $b`
echo `expr $a - $b`
echo `expr $a \* $b`
```

↳ multiplication symbol is accompanied by \

```
echo `expr $b / $a`
echo `expr $b % $a`
```

o/p

```
30
-10
200
2
0
```

backtick symbol

a = $ b

o/p
20
20

~~a == b~~

~~o/p~~

~~true~~

```
echo $[$a==$b]
echo $[$a!=$b]
```

o/p

```
1
0
```

Relational operator

-eq	=	$a -eq $b
-ne	!=	$a -ne $b
-gt	>	$a -gt $b
-lt	<	$a -lt $b
-ge	>=	$a -ge $b
-le	<=	$a -le $b

string operation

a = "hema" b= "hema" c="char"

d = " "

=	equal	`$a = $b`
!=	not equal	`$a != $b`
-z	① checks for empty string	`-z $a`
-n	② checks for non-empty string	`-n $a`
str	Does ① and ② ① → ~~true~~ False ② → True	`$a`

File operation

Command	True/ False
-d	directory/not directory
-f	file/not file
-r	readable/not readable
-w	writeable/not writeable

-x	executable / not executable
-s	size >0 / not
-e	Exist / not Exist
[ie]	[-d $file] [-x $file]

-x — x——x——