# * Shell scripting - part 2 (lesson 16)

## * Variable

vim setup.sh (open file)

→ used to store data and can be referenced later.

→ Similar to variables in general programming language.

variable_name = value

file_name = config.yaml

reference the variable.

echo " using file $file_name to configure something "

variable_name = $(command)

Store the output of a command in a variable.

config_files = $(ls config)

echo " all the configuration files: $config_files "

## * Conditional Statements (if-else statements)

allows you to alter the control flow of the program

- eg: execute a command only when a certain condition is true

```
if [ condition ]
then
    statement
else
    statement
fi.
```

```
if [ -d "config" ]
then
    echo "reading config directory contents"
```

```
    config_files = $(ls config)
else
    echo "config dir not found. Creating one"
    mkdir config
fi
```

## * Basic operators

check if it is a ordinary file

```
if [ -f "config.yaml" ]
```

- -r → if file is readable
- -w → if file is writable
- -x → check if enecutable
- -s → check if empty
- -e → check if file or directory exist

number comparison

```
num_files = xx
```

relational operators.

- works only for numeric values.

```
if [ num_files -eq 10 ]
```

↳ if num_files is equal to 10.

- -lt 10 → less than 10
- -gt 10 → greater than 10
- -ge 10 → greater than equal to 10.
- -le 10 → less than equal to 10.
- -ne 10 → not equal to 10.

String operators.

```
user_group = xx
if [ user_group == "yash" ]
then
    echo "configure the server
elif [ user_group == "admin" ]
then
```

```
        echo "administer the server"
    else
        echo " no permission to configure
                the server".
    fi
```

* Passing arguments to a script

```
    user_group = $1
    if [ "$user_group" = = "nana" ]
    then
        echo
    elif [ " $user_group " = = "admin" ]
    then
        echo
    else
        echo
    fi
```

./setup admin

multiple parameters

```
    config_dir = $1
    if [ -d "$config_dir" ]
    then
        echo " reading config directory contents"
        config_files = $( ls "$config_dir" )
    else
        echo "config dir not found, creating one"
        mkdir "$config_dir"
        touch "$config_dir/config.sh"
    fi

    user_group = $2
    if [ "$user_group" = = "yash" ]
    then
```

```
        echo
    elif [ "$user_group" == "admin" ]
    then
        echo
    else
        echo
```

./setup.sh setup-scripts admin
(para1 = setup-scripts, para2 = admin)

* Read user input

```
#! /bin/bash
echo "Reading user input"
read -p "Please enter your
        password: " user_pwd
echo "Thanks for your password $user_pwd"
```

now we dont know how many parameters users will specify / Dont know parameters.

$* → represents all the arguments as a single string

```
echo "all param $*"
echo "user $1"
echo "user $2"
echo "all parameters : $#"
```

$# → Total no of arguments provided

Shell loop:

- enables you to execute a set of commands repeatedly.
- while, for, until, select loop.

for loop.

```
echo "all param $*"
echo "number of params: $#"
for param in $*
```

```
do
    echo $param
done
```

while loop.

run until certain condition is matched

eg- ping until service available

```
sum = 0
while true
    do
        read -p "enter a score: " score
        if [ "$score" = = "q" ]
        then
            break
        fi
        sum = $(($sum + $score))
        echo "total sum: $sum"
    done
```

BASH - complex syntax

alternative better:

python, ansible (configuration tool)