Advance Bash Scripting- Part:2

1.Regular Expression in Bash Scripting

- **Mainly used in Pattern Matching**

```
#!/bin/bash
read -p "Enter an Email": email

if[[ $email=~ ^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$ ]]; then
echo "Valid Email"
else
echo "Invalid Email"
fi
```

1. =~ operator (Regex Matching Bash)
    a. =~ used to check if a variable matches a regular expression (regex)\
2. ^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$
    a. ^ start of the string
    b. a-zA-Z0-9._%+-
        i. (a-z) lower cases allowed
        ii. (A-Z_ upper cases allowed
        iii.(0-9)digits allowed
        iv.(._%+-)allowed special characters
    c. @ (Mandatory Symbol)
    d. (**a-zA-Z0-9.-]+**) for Domain Name
    e. \. (Dot Before TLD)
    f. {2} must matches atleast 2 letters of the domain
    g. $ end of the string

2. Bash Array & Associative Array Example

- indexed Array

```
names=("Alice""Bob""Charlie")
echo "First Name:${names[0]}";
echo "All Name:${names[@]}";
```

- Associative Arrays

```
declare -A user_info
user_info[name]="Nikunj Soni"
user_infor[role]="DevOps Trainer"

echo "User : ${user_info[name]}, Role: ${user_info[role]}"

# print entire array
for key in "${!user_info[@]}"; do
echo "$key:${user_info[$key]}"
done
```

3. Bash control Structure

1. if -else

```
#!/bin/bash
read -p "Enter a Number" num

if((num > 10)); then
echo "Number is Greater than 10"
elif((num==10)); then
echo "Number is exactly 10"
else
echo "Number is less than 10"
fi
```

TASK: Write a Program to check the validity of user to vote in India by taking the age input from user

```
#!/bin/bash
read -p "Enter a Number" num

if((num > 18)); then
echo "User is Allowed to Vote"
else
echo "User is Not Allowed to Vote"
fi
```

2. For Loop

```
for i in {1..5}; do
```

```
echo "Iteration $i"
done
```

3. While Loop

```
count=1
while((count < =5)); do
echo "Count: $count"
((count++))
done
```

4. Until Loop

```
until [[ -f "/tmp/file.txt" ]]; do
echo "Waiting for File..."
sleep 5
done
```

example :2 Waiting for Service to Strat

```
#!/bin/bash

SERVICE="nginx"
echo "Waiting for $SERVICE to strat....."

until systemctl is-active --quiet "$SERVICE"; do
echo "$SERVICE is not running yet..."
sleep 3
done

echo "$SERVICE is now running"
```

exmaple:3 waiting for internet connection

```
#!/bin/bash

echo "Checking for Internet Connection....."

until ping -c 1 google.com &>/dev/null; do
echo "No internet , retrying in 5 seconds..."
sleep 5
done
```

```
echo "Internet is Available!"
```

4. Bash Variables and Parameters

- **Positional Parameters**

```
#!/bin/bash

echo "Script name: $0"
echo "First Argument: $1"
echo "Second Argument: $2"
```

- **Default Values Using ${VAR:-default}**

```
name=${1:-"Guest"}
echo "Hello , $name"
```

TASK:

Scenario:A system administrator wants to monitor log file (/var/log/syslog) for a specific keyword (eg."ERROR").

The Script Should:

1. **Wait for the log file to exist : before monitoring**
2. **continuously check for the keyword (ERROR) using LOOP**
3. **Send the notification(echo message) when an error is detected.**

Solution:

```
GNU nano 7.2 error.sh #!/bin/bash

LOG_FILE="/var/log/syslog"
KEYWORD="ERROR"

echo "Waiting for log file: $LOG_FILE..."
until [ -f "$LOG_FILE" ]; do
echo "Log file not found, waiting..."
sleep 3
done
echo "Log file detected!"
```

```
tail -Fn0 "$LOG_FILE" | while read line; do
if [[ "$line" =~ $KEYWORD ]]; then
echo "ALERT: Error detected in logs!"
echo "Log Entry: $line"
fi
done
```