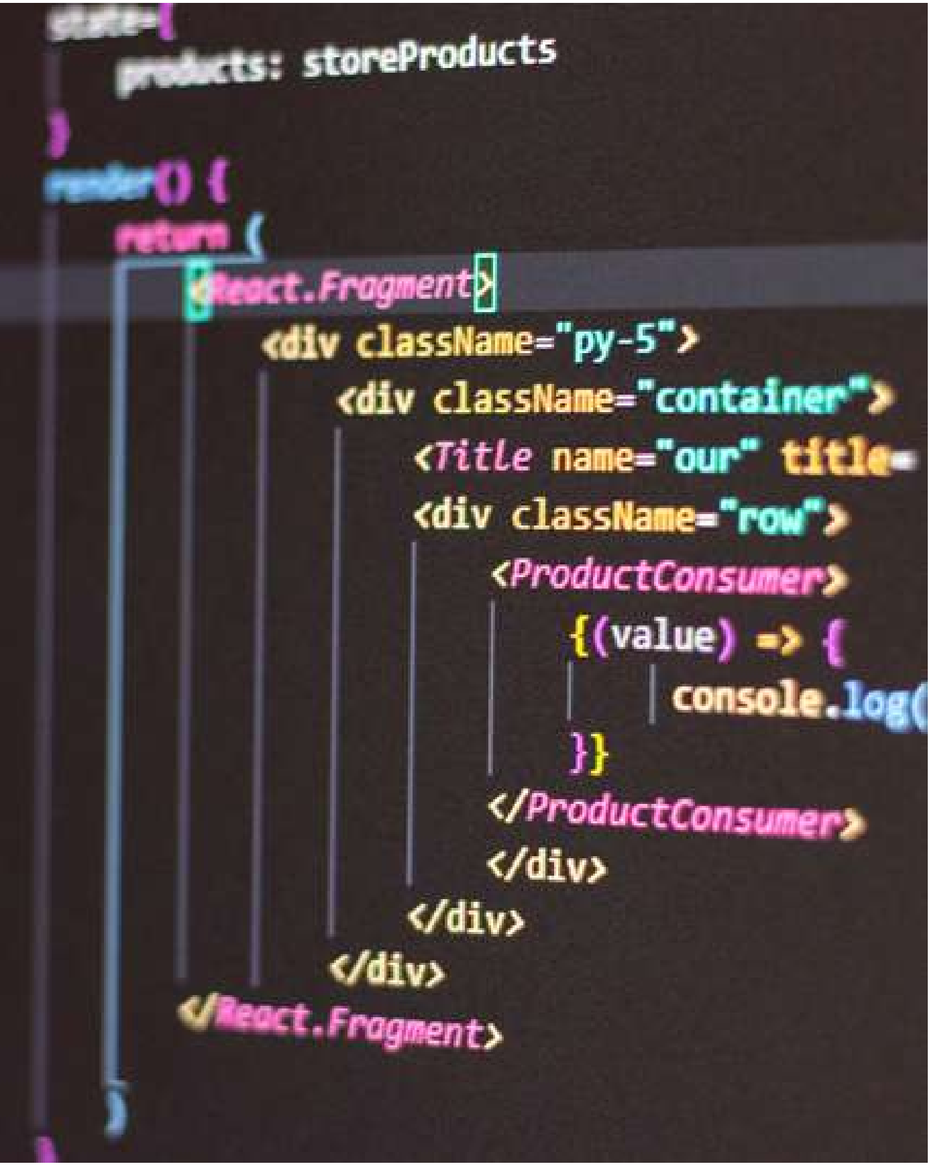
Linux Learning Centre

BASH SHELL SCRIPTING

LLC 500 CYBERSECURITY AND ETHICAL HACKING

www.linuxlearningcentre.co.ke

DEFINITION



Bash scripting refers to the execution of a series of commands written in a file.

Bash

Bash is a command language interpreter. It is widely available on various operating systems and is a default command interpreter on most Linux systems.

Shell

Shell is a macro processor which allows for an interactive or non-interactive command execution.

Scripting

Scripting allows for an automatic commands execution that would otherwise be executed interactively one-by-one.

Bash is used for system administration, data crunching, web application deployment, automated backups, creating custom scripts for various pages, etc.

BASH USAGE EXAMPLE

Bash script that periodically checks the accessibility of a web server

```
#!/bin/bash

# Define the target Apache server URL
target_server="www.google.com"

# Define the threshold (in seconds) for considering the server down
threshold=5

# Log file to store downtime information
log_file="server_downtime.log"

# Function to log downtime
log_downtime() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] Server was down for $downtime_duration seconds." >> "$log_file"
}

# Create the log file if it doesn't exist
if [ ! -f "$log_file" ]; then
    touch "$log_file"
fi

while true; do
    # Measure the time before sending the request
    start_time=$(date +%s)

    # Send a HEAD request to the target server
    response_code=$(curl -sL -w "%{http_code}" -o /dev/null "$target_server")

    # Measure the time after receiving the response
    end_time=$(date +%s)

    # Calculate downtime duration
    downtime_duration=$((end_time - start_time))

    if [ "$response_code" != "200" ]; then
        echo "[$(date '+%Y-%m-%d %H:%M:%S')] Server is not accessible. Response code: $response_code"

        if [ "$downtime_duration" -ge "$threshold" ]; then
            log_downtime
        fi
    else
        echo "[$(date '+%Y-%m-%d %H:%M:%S')] Server is accessible."

        if [ -s "$log_file" ]; then
            echo "[$(date '+%Y-%m-%d %H:%M:%S')] Server is back online." >> "$log_file"
        fi
    fi

    sleep 10  # Wait for 1 minute before the next check
done
```

BASH USAGE EXAMPLE

Shell Script to Backup Files in Directory

Open terminal and create an empty shell script using the following command.

```
#create a directory for the files you need to backup

$ sudo mkdir /home/myfiles
$ cd /home/myfiles
#create sample test files
$ touch testfile{1..10}
#change your current working directory
$ cd /home
#create the backup script file
$ sudo nano backup.sh
```

Now add the following to your shell script.

```
#!/bin/sh
timestamp="$(date +'%b-%d-%y')"
sudo tar -cvpzf
/home/myfiles-${timestamp}.tar.gz
/home/myfiles

#Save and close the file.
```

In the above command,
c - compression
v - verbose
p - retain file permissions
z - create gzip file
f - regular file

```
#Make Script Executable

sudo chmod +x backup.sh
```

Verify the script Run the script with following command.

```
$ sudo /home/backup.sh
```

#reference https://fedingo.com/shell-script-to-backup-files-in-directory/

PRACTICAL BASH USAGE - EXAMPLE 2

Download a sample log file for use
Google:
inurl:access.log filetype:log

$ wget https://npcassoc.org/log/access.log

We are given an Apache HTTP server log that contains evidence of an attack. Our task is to use simple Bash commands to inspect the file and discover various pieces of information, such as who the attackers were, and what exactly happened on the server.
We first use the head command to look at the log file to understand its structure

Sample log file
http://www.offensive-security.com/pwk- files/access_log.txt.gz

PRACTICAL BASH USAGE - EXAMPLE 2

```
$ head access.log
```

```
#sort out the ip addresses
```

```
$ cat access.log | cut -d " " -f 1 | sort -u
```

```
#export the ip addresses
cat access.log | cut -d " " -f 1 | sort > ipaddress.txt
```

```
tail -f 50 access.log
```

ENVIRONMENT VARIABLES

We can view the contents of a given environment variable with the echo command followed by the “$” character and an environment variable name. For example, let’s take a look at the contents of the PATH environment variable:

```
kali@kali:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

Some other useful environment variables include USER, PWD, and HOME, which hold the values of the current terminal user’s username, present working directory, and home directory respectively

```
kali@kali:~$ echo $USER
kali
kali@kali:~$ echo $PWD
/home/kali
kali@kali:~$ echo $HOME
/home/kali
```

PIPING AND REDIRECTION

Piping (using the | operator) and redirection (using the > and < operators) connects these streams between programs and files to accommodate a near infinite number of possible use cases.

Redirecting to a New File

```
kali@kali:~$ ls
Desktop Documents Downloads Music Pictures Public
Templates Videos
kali@kali:~$ echo "test"
test
kali@kali:~$ echo "test" > redirection_test.txt
kali@kali:~$ ls
Desktop Documents Downloads Music Pictures Public
redirection_test.txt Template
kali@kali:~$ cat redirection_test.txt
test
kali@kali:~$ echo "Kali Linux is an open source project" >
redirection_test.txt
kali@kali:~$ cat redirection_test.txt
Kali Linux is an open source project
```

TOOLS

https://github.com/Leviathan36/kaboom.git

https://github.com/Leviathan36/trigmap

THANK YOU!!

BASH SCRIPTING