**Raffle Doc:**

https://www.youtube.com/watch?v=zsajhz2\_50g&list=PLdpzxOOAlwvIKMhk8WhzN1pYoJ1YU8Csa&index=8

<https://medium.com/@devopsshack/30-real-time-shell-scripts-usecases-with-example-codes-8c778368ba2d>

**Project URL**: <https://www.youtube.com/watch?v=DZIb0lXJ2Mk&t=507s>

**Sell-Scripting:**

**Use-cases With Example Codes: -** <https://medium.com/@devopsshack/30-real-time-shell-scripts-usecases-with-example-codes-8c778368ba2d>

**What is shell Scripting in Linux?**

🡺 It is process of automating your day -to-day activities or regular activities on your Linux computer.

**A screenshot of a computer

Description automatically generated**

**Over- View Ans:**

🡺Shell scripting in Linux is a way to automate tasks and manage system operations by writing scripts that execute a series of commands in a Unix-like operating system. These scripts are written for a shell, which is a command-line interpreter that provides a user interface to the operating system.

Here’s a basic overview:

1. **Shell Types**: Linux has several types of shells, including:
   * **Bash (Bourne Again Shell):** The most common default shell for Linux.
   * **sh (Bourne Shell):** An older shell that's more basic.
   * **csh (C Shell):** Known for its C-like syntax.
   * **ksh (Korn Shell):** A shell with features from both the Bourne and C shells.
2. **Writing a Shell Script:**
   * **File Creation**: Create a file with a .sh extension or no extension at all.
   * **Shebang Line:** The first line of the script should specify the interpreter. For example, #!/bin/bash tells the system to use the Bash shell to execute the script.
   * **Commands:** Inside the script, you can include any shell commands you would normally type at the command line.
   * **Permissions:** Make the script executable with chmod +x scriptname.sh.
3. **Basic Components:**
   * **Variables**: Store data to be used later in the script. E.g., name="Alice".
   * **Control Structures:** Include conditionals (if, else, elif) and loops (for, while, until) to control the flow of the script.
   * **Functions**: Define reusable blocks of code within the script.
4. **Example Script:**

**#!/bin/bash**

**echo "Hello, World!"**

**name="Alice"**

**echo "Hello, $name"**

1. **Common Uses:**
   * **System Administration:** Automate backup tasks, user management, and system monitoring.
   * **Batch Processing:** Execute multiple commands or scripts sequentially.
   * **Task Scheduling:** Use with tools like cron for scheduled tasks.

Command’s:

Ls, touch, Man , cat chmod, vim,

df, free, nproc, top

**Variable** – This nothing but a place in the memory in which some values is stored so that you can reuse whenever you want, or you can change the value whatever is needed.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Special Variables:

A screenshot of a computer

Description automatically generated

**set -vx:**  It enables verbose and debug modes, printing each command and its expanded arguments before execution, aiding in debugging

**Operators:**

A close up of black text

Description automatically generated

A close-up of a computer screen

Description automatically generated

A computer screen shot of text

Description automatically generated

A screen shot of a computer

Description automatically generated

A black and white image of a sign

Description automatically generated

A screenshot of a computer

Description automatically generated

**EX:**

A screenshot of a computer program

Description automatically generated

A diagram of a statement

Description automatically generated

A computer screen with white text

Description automatically generated

A screenshot of a computer

Description automatically generated

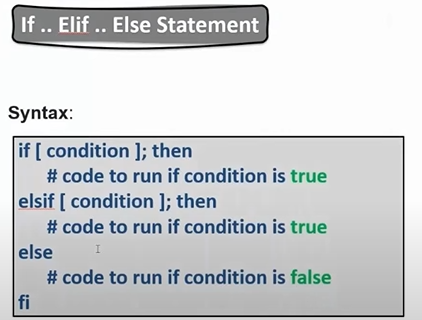
A black background with white text

Description automatically generated

**OUTPUT:**

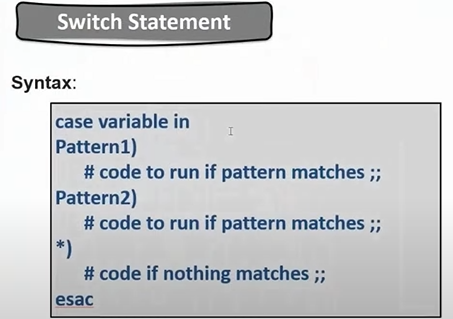
A screen shot of a computer code

Description automatically generated



A screen shot of a computer

Description automatically generated



A computer screen with white text

Description automatically generated

**Loops**: It allows you to execute a block of code repeatedly based on a condition or a set of values, enabling automation and iteration through data efficiently.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

EX:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A black screen with white text

Description automatically generated

A screenshot of a computer

Description automatically generated

A black background with white text

Description automatically generated

A screenshot of a computer

Description automatically generated

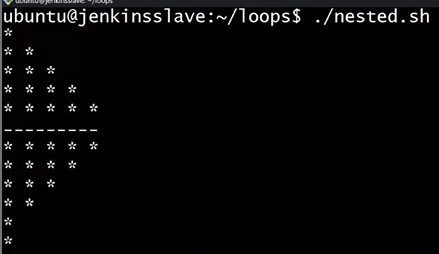
A screen shot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

**Nested Loop:**



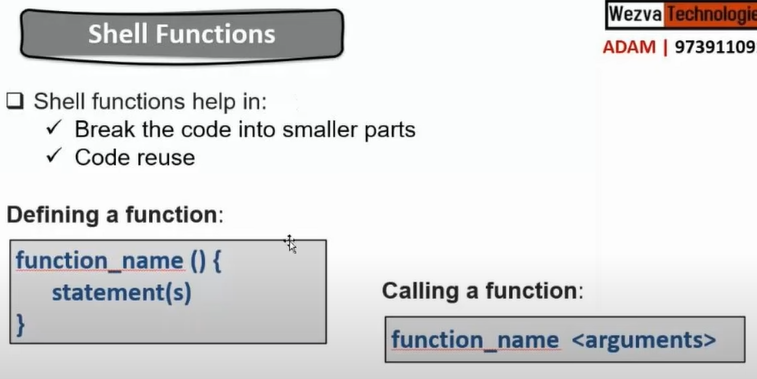
A screen shot of a computer code

Description automatically generated

A screen shot of a computer

Description automatically generated

Function:



A close up of a white background

Description automatically generated

A computer screen with white text

Description automatically generated

A screenshot of a computer program

Description automatically generated

1. **Write a shell script creating one folder and inside that folder created two different files?**

**#!/bin/bash**

**#Created a folder**

**mkdir myfolder\_ABC**

**#created two file’s inside the new folder**

**cd myfolder\_ABC**

**Touch Firstfile Secondfile**

**2.What is purpose of shell scripting using in Devops?**

**🡺** Shell scripting is a key tool in DevOps for automating and streamlining various aspects of the development and operations workflow. Here are some key purposes and benefits of using shell scripting in DevOps:

1. **Automation of Repetitive Tasks**: Shell scripts can automate repetitive tasks such as code deployment, system monitoring, backups, and configuration changes, reducing manual effort and minimizing human error.
2. **Configuration Management**: Shell scripts are used to automate the setup and configuration of servers and environments. This includes installing software, setting environment variables, and configuring system services.
3. **Continuous Integration and Deployment (CI/CD)**: Shell scripts play a crucial role in CI/CD pipelines. They are used to automate build processes, run tests, package applications, and deploy them to various environments.
4. **System Monitoring and Maintenance**: Scripts can be used to monitor system health, check resource usage, and perform routine maintenance tasks. They can alert administrators to potential issues before they become critical problems.
5. **Environment Setup**: Scripts can quickly set up development, staging, and production environments by automating the provisioning and configuration of resources. This helps ensure consistency across environments.
6. **Custom Automation**: For tasks that are specific to the organization's needs or not covered by existing tools, shell scripts provide a flexible way to create custom automation solutions.
7. **Integration with Other Tools**: Shell scripts can be used to interface with other tools and services, such as version control systems (e.g., Git), monitoring tools (e.g., Nagios), and cloud providers (e.g., AWS CLI), enabling a more integrated workflow.
8. **Error Handling and Logging**: Scripts can include error handling and logging mechanisms to track the success or failure of automated tasks, making it easier to diagnose and troubleshoot issues.
9. **Batch Processing**: Shell scripts can process large volumes of files or data in batch mode, which is useful for tasks like data migration, log analysis, and bulk updates.
10. **Simplification of Complex Commands**: Shell scripts allow you to bundle complex commands and sequences of commands into a single script, simplifying their execution and making them easier to manage.

**3. Write a custom node script and this node script will detect the node health of your virtual machine so let’s we will write the a shell script.**

**Step: 1 – vi nodeHealth.sh**

**Step: 2**

#!/bin/bash

#####################################

# Author: Jagadeesh

# Date: 28/08/24

#

# This Script outputs the node health

#

# Version: V1

###########################################

Set -x # Debug Mode

df -h

free -g

nproc

**Step:3 chmod 777 nodeHealth.sh**

**Step:4 ./nodeHealth.sh**

**--------**

**Ps -ef 🡪 It Printing all the information about the processes that are running on the virtual Machine.**

**Grep command 🡪 All the Output that is generated it only fetches the information that want or required.**

**4.Fine out the processes that are running by Amazon and get me the process ID’s?**

**🡺 ps -ef | grep “amazon”**

**A screen shot of a computer

Description automatically generated**

**awk – It can filter out the information from your output.**

**5.what is different between grep and awk command?**

**grep - This command directly give you the entire statements or inter sentences.**

**Awk – This very powerful that it can also give you a specific columns form the output.**

**A screenshot of a computer program

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**set -e – Exit the script when there is an error.**

**What is different between Wget and curl command**

**Curl – It is a command line tool that is used to send and receive data from a server by simply specifying the URL (location of the server) and data.**

**wget**

wget is also a command line tool. However, it is used to **“get”** content from web servers using the URL of the content.

**If -else condition:**

A = 5

B = 10

If [ $a > $b ]

then

echo “a is great than b”

else

echo “b is great than a”

fi

A computer screen with colorful text

Description automatically generated

Trap command:

**Shell Scripting Interview Questions:**

**1. List some of the commonly used shell commands?**

A) Ls, cp, mv, mkdir, touch, vim, grep, netcat...ect

🡺 I used shell scripting for listing the files for that I used the

- **ls command**

**🡺** we have to fine some file using **find command**

**🡺** For debug commands **– Top, sar**(System Activity Reporter),**df** (disk spaces)

**2. Write a simple shell script to list all processes?**

**A) ps -ef**

Ex: I Just want the process’s ID which is displaying

🡺 **ps -ef | awk -F “ ” ‘{print $2}’**

1**. Awk command** – It filter the out put of specific line or file.

2. **F (Field separator)** – It will be separating a string a from a string that is a white space

**\*3. write a script to print only error from a remote log.**

A) curl pipe grep

**Ex:** 1. curl google.com | grep error

A screenshot of a computer screen

Description automatically generated

2. curl <https://raw.githubusercontent.com/iam-veeramalla/sandbox/main/log/dummylog01122022.log> | grep trace

**Curl** is retrieving output (or) getting your entire logfile

**Grep** it getting a section of output, like if you want error only error output we come.

**Pipe(|) –** It will send a enter output of first command and second command.

**4. Write a shell script to print number divided by 3 & 5 and not 15.**

**(OR) Print even number ---> Divisible by 2**

**(OR) Print Odd number -----> Not Divisible by 3**

**(OR) Print num divisible by 3, (OR) prime numbers.**

**Ans Hint:**

Divisible by 3 (IF - condition) ----> 3,6, 9, 12, 15

Divisible by 5 (IF- condition) ----> 5,10, 15

And not divisible by 15 (Not-if)

1 to 100 num (for Loop)

**Step1:** Vim simple.sh

**Step2:**

**#!/bin/bash**

**for i in {1…100}; do**

**if ([ `expr $i % 3` == 0 ] || [ `expr $i % 5` == 0] ) && [ `expr $i % 15` != 0];**

**then**

**echo $i**

**fi;**

**done**

**Step3:** chmod 777 simple.sh

**Step4:** ./simple.sh

**5. Write a script to print number of “S” in Mississippi**

**A)**

**Step1:** Vim num.sh

**Step2:**

**#!/bin/bash**

**X= mississippi**

**grep -o “s” <<<”$x” | wc -l**

**Step3:** chmod 777 num.sh

**Step4:** ./num.sh

**6. How will you debug the shell Script?**

A**) set -x**

7. What is crontab in Linux? Can you provide an example of usage?

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

**7. How to open a read-only file?**

A) **vim -r test.txt**

**8. what is different b/w soft and hard link?**

**A)**

* A hard link always points a filename to data on a storage device.
* A soft link always points a filename to another filename, which then points to information on a storage device.

**A screenshot of a computer

Description automatically generated**

**9. What is the difference B/w break and continue statements?**

🡺 The Break statement stops the **entire loop process**.

🡺 The Continue statement only stops the **current iteration of**

**the loop.**

**10. What are some disadvantages of shell scripting?**

**A black text on a white background

Description automatically generated**

**11. Is bash dynamic or statically typed and why?**

**🡺**Bash is considered **dynamically typed**. This means that variables in Bash do not have a fixed type, and their type can change at runtime based on the value assigned to them. For example, you can assign a string to a variable and later assign a number to the same variable without any type declaration or error.

**12. How will you replace multiple occurrences of a string in a file?**

**🡺**Replace multiple occurrences of a string in a file using Bash, you can use the **sed** command, which is a stream editor.

**example:**

**sed -i 's/old\_string/new\_string/g' filename**

**Breakdown of the command:**

* **sed**: The command to invoke the stream editor.
* **-i:** This option edits the file in place (modifies the original file). You can add a suffix (e.g., -i.bak) to create a backup of the original file.
* **s/old\_string/new\_string/g:** This is the substitution commandwhere:
  + **s:** Stands for "substitute."
  + **old\_string:** The string you want to replace**.**
  + **new\_string:** The string you want to replace it with.
  + **g:** Stands for "global," meaning it will replace all occurrences in each line**.**

**Example:**

**If you want to replace all occurrences of "apple" with "orange" in a file called fruits.txt, you would run:**

**🡺 sed -i 's/apple/orange/g' fruits.txt**

**13. Explain about a network troubleshooting utility?**

**🡺 Traceroute** is a network diagnostic tool used to track the route that packets take from one networked device to another. It provides information about each hop (router or switch) the packets encounter along the way, including the response times for each hop.

**14. How will you sort list of names in a file?**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

**15 how will you manage logs of system that generate huge log files every day?**

**🡺 Log Rotation:**

* Use tools like logrotate to automatically rotate log files. This means old log files are compressed and archived, while new logs continue to be written to a fresh file.
* Configuration examples for logrotate include setting daily, weekly, or monthly rotations, and specifying how many old logs to keep.

**A screen shot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generatedDay-7**

Write a script to report the usage of AWS in your project?

**Step1: -** vim aws\_resources.sh

**Step2: -**

**#!/bin/bash**

#########################

# Author: Jagadeesh

# Date: 20-09-24

# Version: v1

# This script will report the aws resource usage.

###########################

**Set -x**

# AWS S3

# AWS EC2

# AWS Lambda

# AWS IAM Users

# List s3 buckets

**echo “print list of s3 buckets”**

**aws s3 ls**

# list Ec2 Instances

**echo “print list of ec2 buckets”**

**aws ec2 describe-instances**

# list lambda

**echo “print list of lambda functions”**

**aws lambda list-functions**

# list IAM users

**echo “print list of IAM Users”**

**aws iam list-users**

**:wq!**

**Step3:** chmod 777 aws\_resources.sh

**Step4:** ./aws\_resources.sh

A screenshot of a computer

Description automatically generated

**Note:**

**Docker Scenario**

1.Docker can use it for various purposes in your organization now there could be a scenario where you have Kubernetes and inside the Kubernetes you have the docker engine running so that your docker engine helps the Kubernetes to create the containers for deployment.

2. It may be some situation where you are not using Kubernetes, and you want to create a container, or you want to deploy application on a standalone machine that means you are going to use some tool like docker engine or docker compose on single machine even in that case you need to use docker engine.

3. we are using docker for building an image because that is a popular use case for your build pipelines whether your building a microservice based out of java or spring boot or node or react whatever it could be one of the common thing which we need is how do we build the Image as well.

**Shell script Project**

**URL:** <https://gitlab.com/learndevopseasy/scripting>