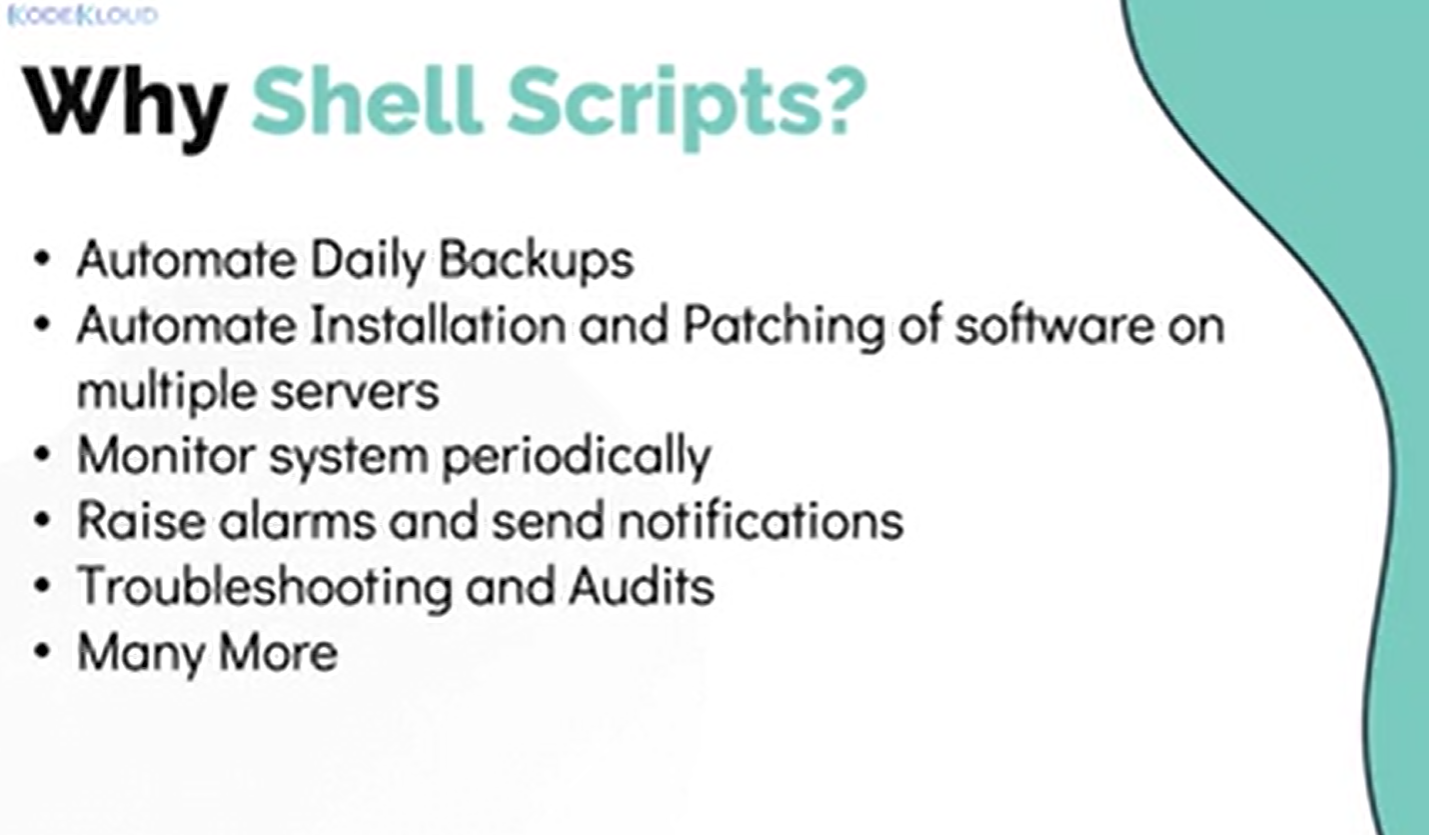
SHELL SCRIPTING

<https://devopscube.com/linux-shell-scripting-for-devops/?utm_source=chatgpt.com>

A **shell** is a program that acts as an interface between the user and the operating system. It allows users to interact with the OS by entering commands in a **command-line interface (CLI)**.

Shell scripting is **crucial in DevOps** because it allows for automation of repetitive tasks, system configuration, application deployment, monitoring, and more. It's widely used to streamline workflows, improve efficiency, and minimize human error.

**Common Types of Shells**:

They allow users to execute commands, run scripts, and interact with the operating system.

Bash: Bourne Again Shell): **Bash** is the most common shell in Linux

Zsh (Z Shell)

Csh (C Shell)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Feature** | **Bash 🟢** | **Ksh 🟠** | **Sh 🔵** | **Dash 🔴** | **Zsh 🟣** | **Csh 🔵** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Common Use** | Default in Linux, DevOps scripting | UNIX scripting | System scripts (Old UNIX) | Fast startup scripts (Debian/Ubuntu) | Interactive shell | C-like shell (Old UNIX) |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Scripting Power** | ✅ Best | ✅ Good | ❌ Basic | ❌ Minimal | ✅ Advanced | ❌ Not great |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Auto-Completion** | ✅ Yes | ✅ Yes (Limited) | ❌ No | ❌ No | ✅ Best | ❌ No |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Performance** | ⚡ Fast | ⚡ Fast | 🐢 Slow | 🚀 Very Fast | ⚡ Fast | ⚡ Fast |

***Command to get which shell using: echo $SHELL***

What is Shebang (#!)

A **shebang (#!)** is the first line of a shell script. It tells the system which program (like bash or python) should run the script.

Eg: #!/path/to/interpreter

      bash: #!/bin/bash

       sh: #!/bin/sh

Echo:

The echo command in shell is used to display text or variables on the terminal. It is commonly used in scripts for printing messages

Eg:   echo “my name is chinju”

How to create a file for a shell script?

touch filename.sh

How to Execute a Shell Script?

./filename

Sh filename

Bash filename

Permission to a file

 chmod command in Unix/Linux is used to change the permissions of a file or directory. "chmod" stands for change mode. It defines who can read, write, or execute a file.

Permissions are classified into three categories:

1. **Read (r)**: Permission to view the file.
2. **Write (w)**: Permission to modify or delete the file.
3. **Execute (x)**: Permission to run the file as a program.

r = 4

w = 2

x = 1

These permissions are assigned to three groups:

* **User (u)**: The owner of the file.
* **Group (g)**: Users who are part of the file's group.
* **Others (o)**: Everyone else.

The sum of numbers defines the permission:

* **7 (4+2+1)** = read, write, and execute.
* **6 (4+2)** = read and write.
* **5 (4+1)** = read and execute.
* **4** = read only.
* **3 (2+1)** = write and execute.
* **2** = write only.
* **1** = execute only.
* **0** = no permission.

set -x

The set -x command in shell scripting is used for **debugging** and helps in **tracing the commands** that are executed within a script. When you include set -x in your script, it will print each command (with expanded arguments) to the terminal before executing it.

#!/bin/bash

#Author: Chinju

#Date: 15-Feb-25

#

#

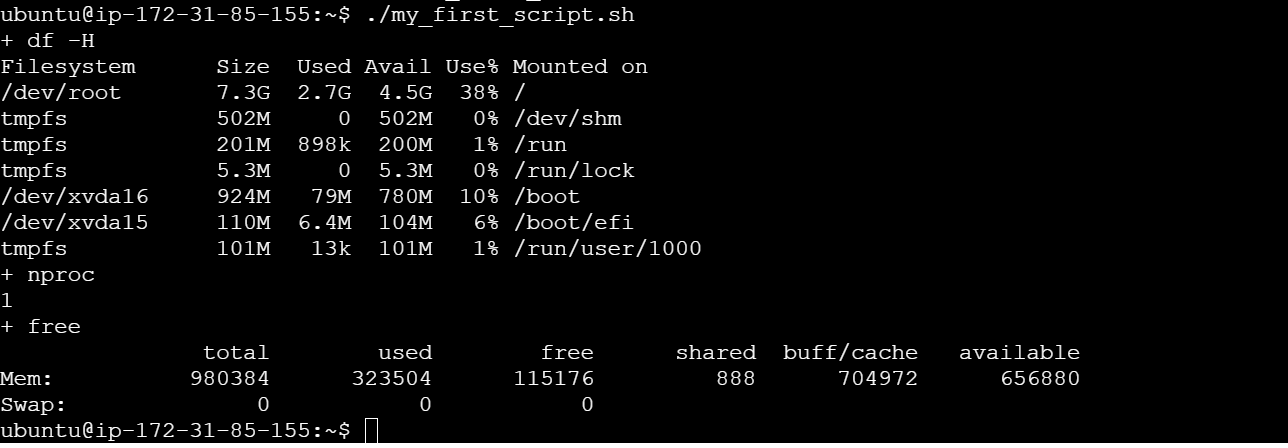
set -x

df -H

nproc

Free

OUTPUT:



set -x enables the debug mode (with the + prefix).

set +x disables it.

set -e command in a shell script is used to make the script **exit immediately** if any command within the script returns

**Without set -o pipefail**, a pipeline of commands will return the exit status of the **last command** in the pipeline, even if earlier commands in the pipeline fail.

Basic commands

* Ls : List Directory
* Pwd: Print Working Directory
* Cd : Change Directory
* Cd ..  : (Move Up One Directory)
* Cd ../.. :Move Up Two Directorie
* Ls -ltr : The ls -ltr command is used to list files with detailed information, sorted by modification time in reverse order (oldest first).

      -l: Detailed listing.

   -t: Sort by modification time.

   -r: Reverse the order.

* Vi : text editor which can use to create and edit file
* Touch : Create an Empty File

              touch filename

* :wq! : Save and Quit
* :q! : quit not saving
* Cat : cat command is used to display the content of a file.
* Mkdir  :Make Directory
* rm : Remove Files
* rm -rf : remove directories and their contents
* man : (Manual Pages)
* Clear: command is used to clear the terminal screen.
* Cp: copy, original files will be there in main location

       cp -r→ copying directory

       Cp → copying files

* Mv: move or rename

     mv oldname.txt newname.txt

     mv myfolder /home/user/Documents/

* Nproc: to get processor core number
* Top:  Provides a real-time, dynamic view of system performance (CPU, memory, processes).
* Free : to get memory details

**Default (free)**: Memory is displayed in **kilobytes (KB)**.

**free -g**: Memory is displayed in **gigabytes (GB)**.

* df -h : disk
* ps -ef : display a detailed list of all running processes on the system

            Use top when you **want real-time monitoring** of system resources.

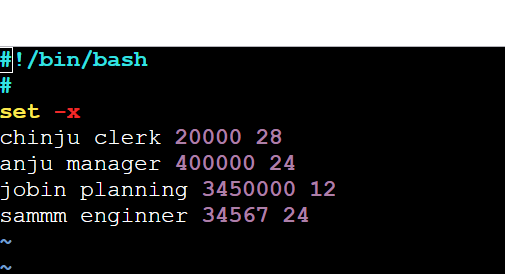
            Use ps -ef when you **want a one-time snapshot** of all running processes

ps -ef | grep "Amazone":   Filters the output of ps -ef to show only lines that contain      the word **"Amazone"**.

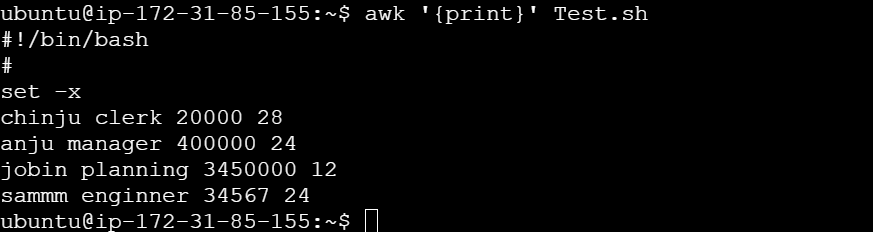
Awk: **AWK** is a powerful text-processing tool used in Unix/Linux for pattern scanning and processing

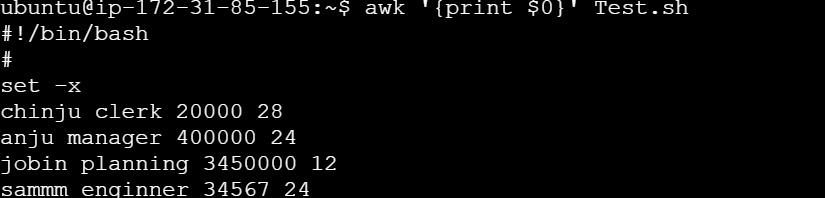
Scan file line by line

Eg: Create a file with below contents and saved as Test.sh

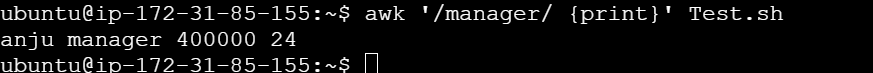


1. awk ‘{print}’ Test.sh or awk ‘{print $0}’



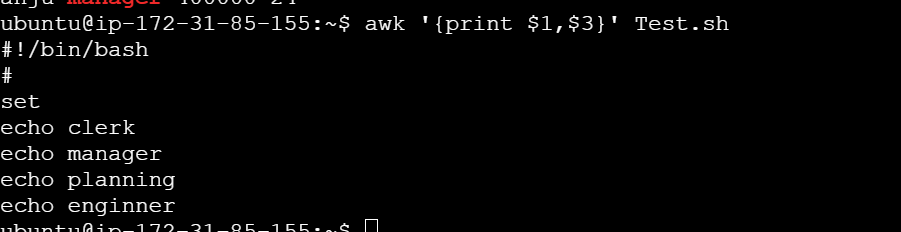


1. Awk ‘/manager/ {print}’ Tesh.sh

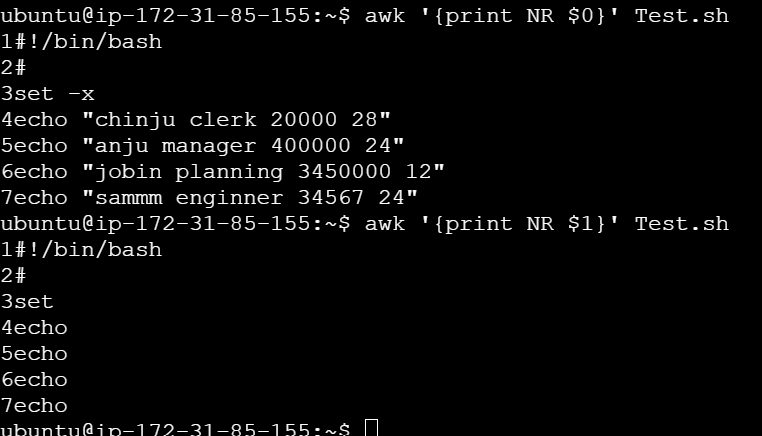


1. To print only selected columns

Awk ‘{print $1,$3}’ Tesh.sh



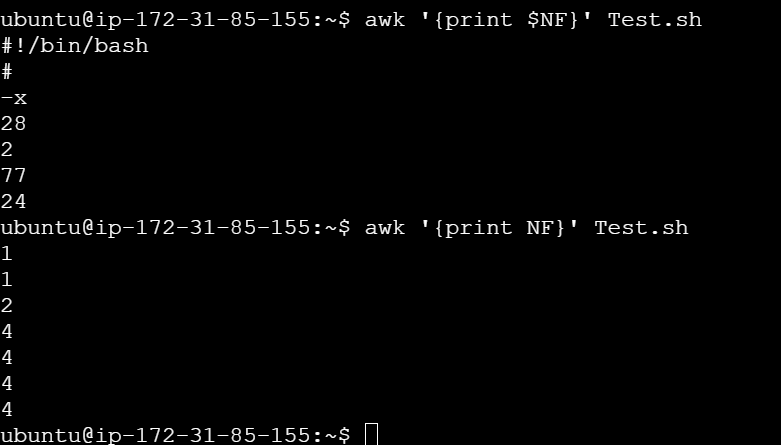
1. NR: Number of Records



1. NF number of fields

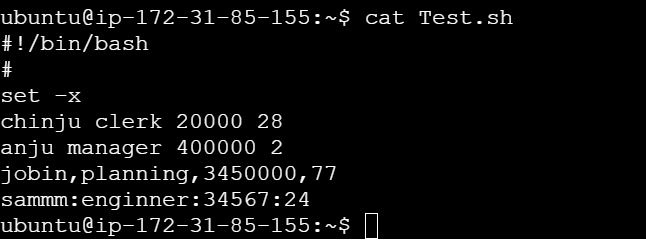
Two ways we can use NF

1. Awk ‘{print NF}’
2. Awk ‘{print,$NF}’



5)  F : **Field Separator**. F" ", inside the quotes we can give comma or space or colun…..

   Create an Test.sh file

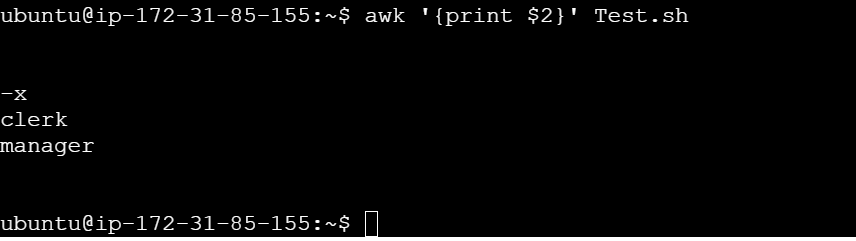


1. By default awk taken space as filed separator

jobin,planning,3450000,77

Sammm:enginner:34567:24

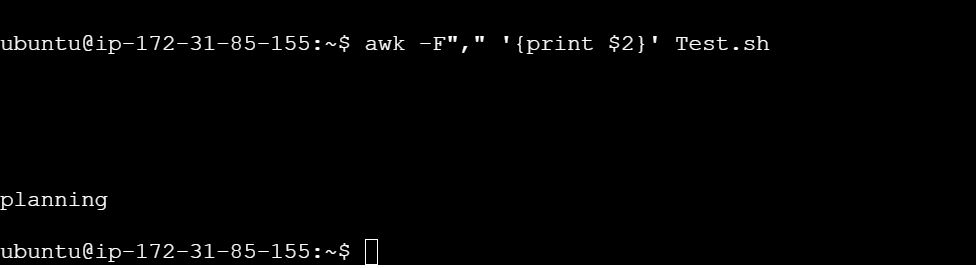
there are no spaces between the words, so the entire line is treated as **one field**.



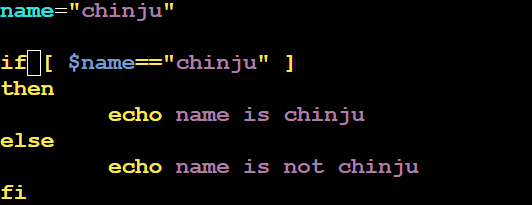
Here also inside filed separator we provided space so apart from space it will consider as sink=gle filed

1. 

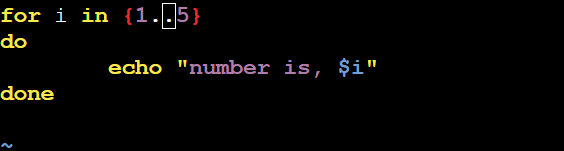
Filed separator with comma



If else



For loop



What Are Signals in Linux

They are used to notify processes about events such as termination, interruptions, or the need for special handling.

SIGINT: Interrupt signal (CTRL+C)

SIGTERM : Termination signal (default for kill)

SIGALRM:Alarm clock

Trap:

It helps prevent unwanted termination of a script.

trap 'commands' SIGNAL

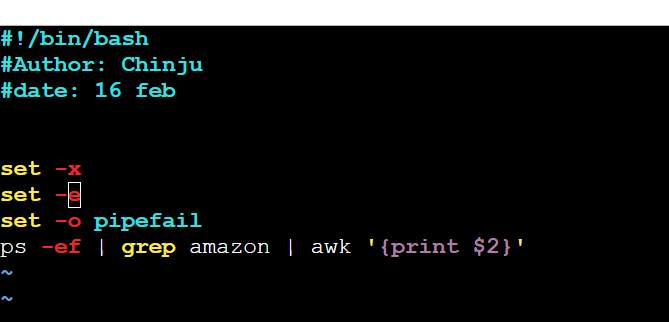
#!/bin/bash

trap 'echo "Don't press CTRL+C!"' SIGINT

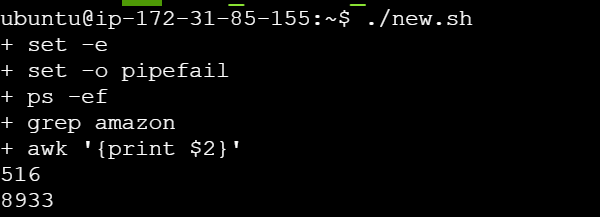
sleep 10  # Just a simple command to keep the script running for a while

PROGRAM

1. Write a program to fetch all the process running, in that get amazon PID



Output:



2) If u have a log file how to check the errors.

1. Copy the log file to any git and get its url then give command

      Curl provide url | grep ERROR

1. If u want to download the content from then use wget

wget command is used to download files from the web through HTTP, HTTPS, and FTP protocols

Wget provide url | grep ERROR

Wget <https://raw.githubusercontent.com/iam-veeramalla/sandbox/refs/heads/main/log/dummylog01122022.log> | grep ERROR

3) script for finding any files .

  Sudo find / -name filenameto search

3) Write a shell script to create an folder with name chinju and inside this folder create one file

#!/bin/bash

mkdir chinju

cd chinju

touch file1

If someone asking why u need shell scripting for devops?  
Explain we have 1000 server and daily we need to monitor its node health check then we can use shell scripting

Write and shell script to check node health check of a server

#!/bin/bash

# Author: Chinju

#  Date: 15-Feb

#  Description: Node Health Check for Memory and CPU Usage

#

# #Set the log file for health check results

 LOG\_FILE="/home/ubuntu/node\_health\_check.log"

 echo "Node Health Check - $(date)" >>$LOG\_FILE

#

# Check CPU usage

 echo "CPU Usage:" >> $LOG\_FILE

 top -bn1 | grep "Cpu(s)" >> $LOG\_FILE

#

# Check Memory usage

 echo "Memory Usage:" >> $LOG\_FILE

 free -h >> $LOG\_FILE

# # Display the health status in the terminal

 echo "CPU and Memory health check completed. Results are saved in $LOG\_FILE."

Without the $, it would just be treated as the string LOG\_FILE, not the path to the log file.

**>**: Redirects output to a file, **overwriting** the file's content if it exists.

**>>**: Redirects output to a file, **appending** the output to the end of the file without overwriting it.