Basic Linux Shell Scripting for DevOps Engineers

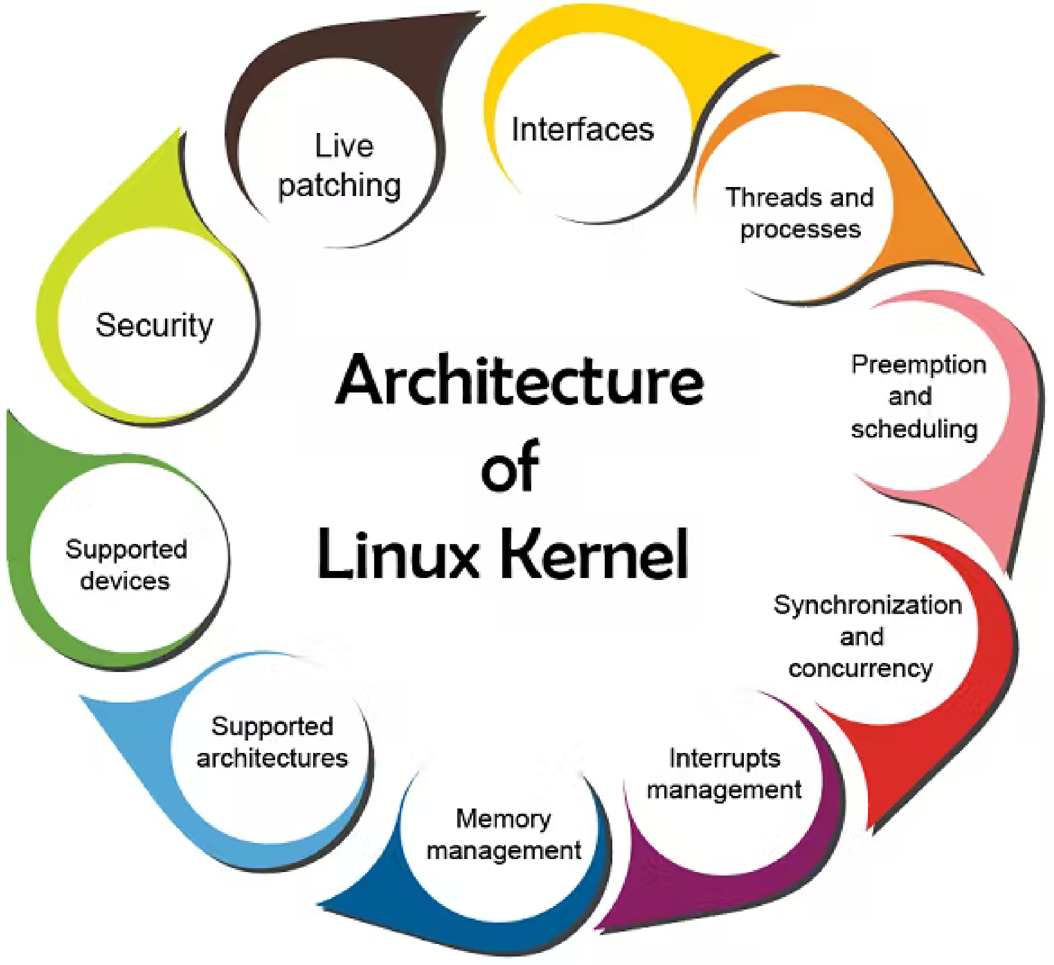
Day 4 : 90Days of DevOps Challenge

SS Shivraj Salunkhe · Mar 27, 2023 · 6 min read

What is Kernel ?

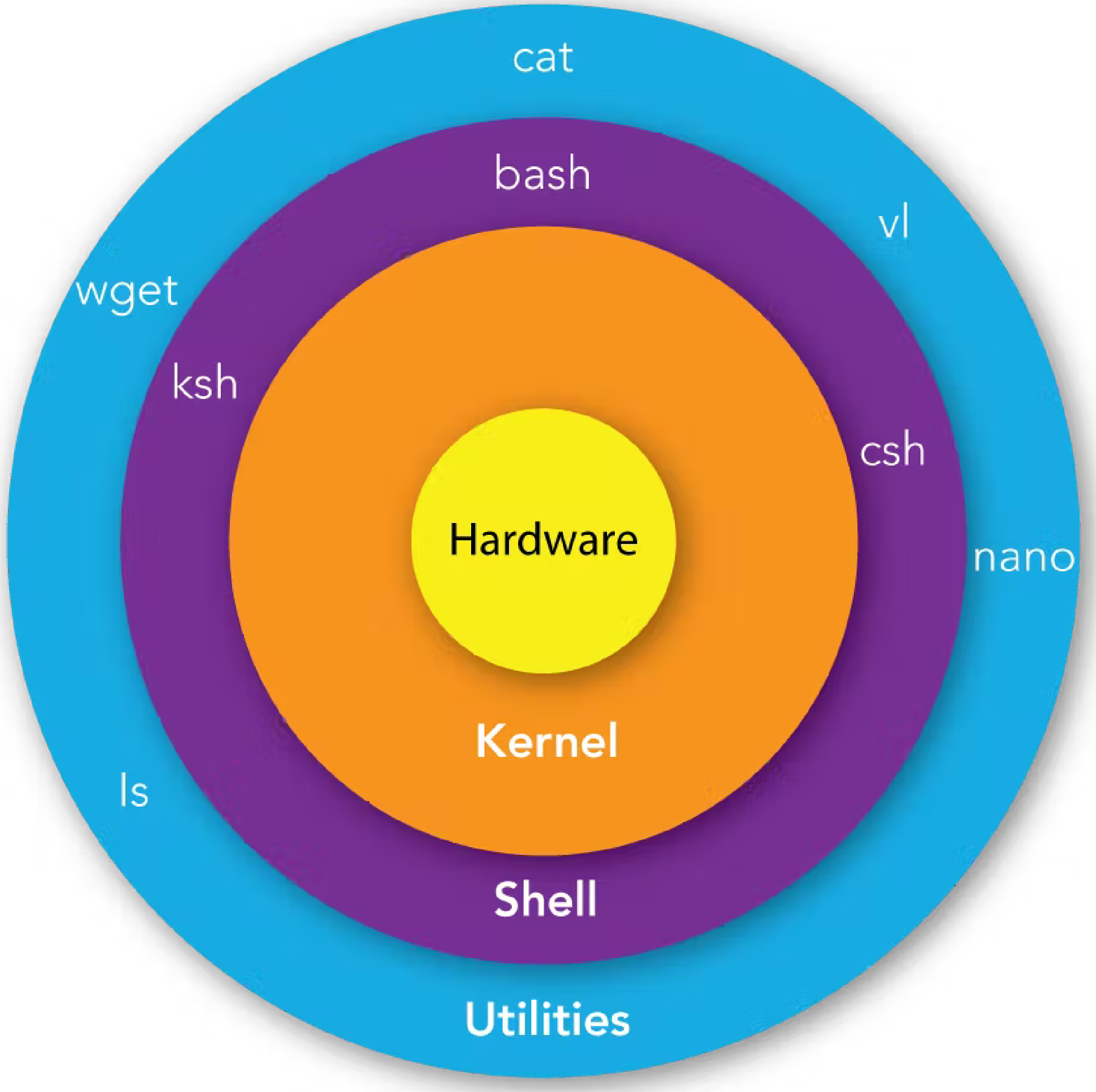
1. In computing, a kernel is the core component of an operating system that manages system resources, communicates with hardware devices, and provides fundamental services to other software running on the system.
2. It acts as a bridge between the hardware and software layers of a computer system, and is responsible for managing system memory, input/output operations, scheduling tasks, and handling errors and exceptions.
3. The kernel is loaded into memory when the operating system starts, and remains resident in memory until the system is shut down.

In shell scripting, the kernel is typically accessed through system calls, which are functions provided by the kernel that can be invoked by user-level applications. For example, a shell script might use the "mkdir" command to create a new directory, which in turn makes a system call to the kernel to allocate a new inode and directory entry in the file system.



What is Shell ?

- In computing, a shell is a user interface that allows a user to interact with a computer system. More specifically, a shell is a command-line interface (CLI) that provides a way for users to enter commands and execute them on a computer system.
- In a shell, users can type commands and the shell will interpret those commands and execute them on the underlying operating system. A shell also provides various features such as environment variables, command history, and command line editing that make it easier for users to work with the system.
- There are several different types of shells available, including the Bourne shell (sh), the C shell (csh), the Korn shell (ksh), the Bourne-Again shell (bash), and others. Each shell has its own set of features and capabilities, and users can choose the shell that best fits their needs and preferences.
- In addition to command-line shells, there are also graphical user interfaces (GUIs) that provide a visual way for users to interact with a computer system. However, command-line shells remain popular among system administrators, developers, and power users because of their flexibility and power.



What is Linux Shell Scripting?

- Linux shell scripting refers to the process of writing scripts, or programs, that are executed in a Linux shell. A shell script is a text file that contains a series of commands that can be executed together in sequence, similar to a batch file in Windows.
- In Linux, the shell is a command-line interface that allows users to enter commands and interact with the operating system. A shell script is a set of instructions that can be executed in the shell, automating tasks and making it easier for users to perform repetitive tasks.
- Shell scripts can be used for a wide range of purposes, such as system administration, software development, data analysis, and more. They can be used to perform tasks such as creating backups, installing software, automating system maintenance tasks, and processing data.
- Linux shell scripting is a powerful tool that can greatly improve productivity and streamline workflows. It is an essential skill for system administrators, developers, and anyone who works with Linux systems on a regular basis.

Task :

\>> Explain in your own words and examples, what is Shell Scripting for DevOps?

1. DevOps is a methodology that aims to bridge the gap between software development and operations teams, by promoting collaboration and automation. Shell scripting plays a key role in DevOps by enabling automation of various tasks such as building, testing, and deploying software.

2. For example, a DevOps engineer might write a shell script to automate the process of building and deploying a web application. The script might include commands to download the source code from a version control system, build the application, run tests, and deploy the application to a server.
3. Shell scripting can also be used to automate tasks such as provisioning and configuring infrastructure, monitoring system performance, and managing logs and backups. By automating these tasks, DevOps teams can save time and reduce the risk of human error, while improving the speed and reliability of the software development and deployment process.

\>> What is `#!/bin/bash?` can we write `#!/bin/sh` as well?

-> `#!/bin/bash` is called the shebang or hashbang, which is a special construct that is used at the beginning of a script file in **Linux/Unix** systems to tell the system which interpreter to use to execute the script. In this case, it tells the system to use the **Bash shell** to interpret and execute the script.

-> Similarly, `#!/bin/sh` would tell the system to use the **Bourne shell (sh)** to interpret and execute the script.

-> Both of these **shebangs** are valid and widely used, but they have some differences in behavior and features. **Bash** is a more advanced shell than **Bourne shell** and provides additional features such as command-line editing, command history, and job control that are not available in the basic Bourne shell.

-> Therefore, if your script uses Bash-specific features, it is better to use `#!/bin/bash` shebang. However, if your script only uses basic shell features that are compatible with both Bash and Bourne shell, then using `#!/bin/sh` would be more portable and compatible with a wider range of systems.

\>> Write a Shell Script which prints `I will complete #90DaysOofDevOps challenge.`

```
#!/bin/bash
echo "I will complete #90DaysOfDevOps challenge"
```

Here's how to use this script:

1. Open a text editor and paste the code above into a new file.
2. Save the file with a **.sh** extension (e.g. **devops_challenge.sh**).
3. Open a terminal or shell and navigate to the directory where the file is saved.
4. Make the file executable using the command: `chmod +x devops_challenge.sh`.
5. Execute the script using the command: `./devops_challenge.sh`.

When you run the script, it will print the message **"I will complete #90DaysOfDevOps challenge"** to the terminal.

\>> Write a Shell Script to take user input, input from arguments and print the variables.

```
#!/bin/bash
echo "Enter your name:"
read name
echo "Your name is: $name"

echo "The first argument is: $1"
echo "The second argument is: $2"
```

Here's how to use this script:

1. Open a text editor and paste the code above into a new file.
2. Save the file with a .sh extension (e.g. **input_script.sh**).
3. Open a terminal or shell and navigate to the directory where the file is saved.
4. Make the file executable using the command: `chmod +x input_script.sh`.
5. To run the script with arguments, use the command: `./input_script.sh argument1 argument2`.
6. To run the script and input from user, use the command: `./input_script.sh` and enter the name when prompted.

When you run the script, it will ask you to enter your name, and then print the name back to the terminal. It will also print the first and second arguments passed to the script, if any.

\>> Write an Example of If else in Shell Scripting by comparing 2 numbers.

```
#!/bin/bash

num1=10
num2=20

if [ $num1 -eq $num2 ]
then
  echo "The numbers are equal"
else
  echo "The numbers are not equal"
fi
```

- In this example, we define two variables `num1` and `num2`, and then use the if-else statement to compare them. The `-eq` operator is used to check if the two numbers are **equal**.
- If the two numbers are **equal**, the script will print "The numbers are equal" to the terminal. Otherwise, it will print "The numbers are not equal".
- Note that we have used the **square brackets** (`[` and `]`) to enclose the comparison expression. This is necessary in **shell scripting** for proper syntax.
- You can modify this example to use different values for `num1` and `num2` to compare any two numbers you want.

shell Devops Learning Journey Linux technology

WRITTEN BY

Shivraj Salunkhe

Follow

Dedicated and hardworking undergraduate student pursuing a degree in Information Technology with a passion for learning, leadership experience, and real-world skills through internships and part-time jobs in IT sector. Actively looking for new Opportunities and committed to do personal and professional growth.

MORE ARTICLES

Shivraj Salunkhe

Understanding package manager and systemctl

What is a package manager in Linux? In Linux, a package manager is a software tool that helps users...

Shivraj Salunkhe

File Permissions and Access Control Lists

In Linux, file permissions refer to the settings that determine who can read, write, or execute a fi...

Shivraj Salunkhe

Advanced Linux Shell Scripting for DevOps Engineers with User management

Advanced Linux Shell Scripting for DevOps Engineers covers a wide range of topics, including : Auto...