What is Operating System

An operating system (OS) is a software program that acts as an intermediary between computer hardware and user applications. It manages the computer's hardware resources, provides common services for computer programs, and enables users to interact with the computer.

What is Unix

Unix is a family of multitasking, multiuser computer operating systems that derive from the original AT&T Unix, developed in the 1970s at the Bell Labs research center by Ken Thompson, Dennis Ritchie, and others.

What is Linux

Linux is a Unix-like operating system kernel initially developed by Linus Torvalds in 1991

Kernel

The kernel is the core component of an operating system that manages the system's hardware resources and provides essential services for other software running on the system. It acts as an intermediary between applications and the hardware, handling tasks such as process scheduling, memory management, device input/output, and system calls.

Shell

A shell is a command-line interface (CLI) program that provides users with a way to interact with the operating system and execute commands. It interprets user commands and executes them by invoking the corresponding system programs or utilities.

Process

A process is an instance of a program that is currently being executed by the computer's CPU. It includes the program's code, data, and resources allocated by the operating system, such as memory, file descriptors, and CPU time

Program

A program is a set of instructions or code written in a programming language that performs a specific task when executed by a computer. Programs can range from simple scripts to complex applications and can be stored as files on a computer's storage device.

Monolithic Kernel:

In a monolithic kernel architecture, all the operating system services run in kernel space. This means that the entire operating system, including device drivers and system call interface, resides in the kernel space. Monolithic kernels are known for their simplicity and high performance but can be less modular and harder to extend or customize.

Microkernel:

A microkernel architecture aims to minimize the kernel's size and complexity by delegating most operating system services, such as device drivers and file systems, to user-space processes known as

servers. The microkernel itself only provides essential services like inter-process communication and memory management. Microkernels are known for their modularity, flexibility, and reliability but can suffer from performance overhead due to increased inter-process communication.

What is sed

SED (Stream EDitor) is a powerful command-line utility used for text manipulation and transformation in Unix-like operating systems. It reads text from standard input or files, performs operations on that text (such as search and replace), and then outputs the modified text to standard output or files. SED is particularly useful for tasks like search and replace, text filtering, and text processing in shell scripts and command pipelines.

List of Basic commands

ls: Lists files and directories in the current directory.
cd: Changes the current directory. Usage: cd [directory].
pwd: Prints the current working directory.
mkdir: Creates a new directory. Usage: mkdir [directory].
rmdir: Removes an empty directory. Usage: rmdir [directory].
touch: Creates an empty file or updates the timestamp of an existing file. Usage: touch [filename].
rm: Removes files or directories. Usage: rm [filename] (for files) or rm -r [directory] (for directories).
cp: Copies files and directories. Usage: cp [source] [destination].
mv: Moves or renames files and directories. Usage: mv [source] [destination].
cat: Displays the contents of a file. Usage: cat [filename].
grep: Searches for patterns in text. Usage: grep [pattern] [filename].
chmod: Changes file permissions. Usage: chmod [permissions] [filename].
chown: Changes file ownership. Usage: chown [owner]:[group] [filename].
history: Displays a list of previously executed commands. Usage: history.
man: Displays the manual pages for commands. Usage: man [command].
date: Displays the current date and time. Usage: date.
uptime: Displays the system's uptime. Usage: uptime.
echo: Prints text or variables to the terminal. Usage: echo [text].
find: Searches for files and directories in a directory hierarchy. Usage: find [directory] -name [filename].
ps: Displays information about active processes. Usage: ps.
kill: Terminates processes by their process IDs (PIDs). Usage: kill [PID].
killall: Terminates processes by their names. Usage: killall [processname].
top: Displays real-time information about system resource usage, including CPU, memory, and processes. Usage: top.
df: Displays disk space usage for file systems. Usage: df.
du: Displays disk space usage for files and directories. Usage: du [directory].
uname: Displays system information such as the operating system name, kernel version, and hardware architecture. Usage: uname [options].
cal: Displays a calendar for the current month or a specified month and year. Usage: cal [month] [year].
clear: Clears the terminal screen. Usage: clear.
whoami: Displays the username of the current user. Usage: whoami.

Networking commands in linux

ifconfig: Displays information about network interfaces, such as IP addresses, MAC addresses, and network configuration. Usage: ifconfig.

ip: A versatile command for managing network interfaces, addresses, routes, and tunnels. Usage: ip [options].

ping:Sends ICMP echo requests to a specified network host to test network connectivity. Usage: ping [hostname or IP address].

traceroute (or tracepath): Traces the route packets take to reach a destination host, showing the IP addresses of intermediate routers. Usage: traceroute [hostname or IP address].

netstat: Displays network connections, routing tables, interface statistics, and more. Usage: netstat [options].

ss: A replacement for netstat with similar functionality, displaying socket statistics. Usage: ss [options].

route: Displays or modifies the kernel's IP routing table. Usage: route [options].

nslookup: Performs DNS queries to retrieve information about domain names, IP addresses, and mail exchange records. Usage: nslookup [hostname].

dig: A flexible DNS lookup utility that provides detailed DNS information, including DNS records and query options. Usage: dig [options] [hostname].

host: Provides DNS lookup and name resolution functionality. Usage: host [hostname].

arp: Displays or manipulates the system's ARP cache, which maps IP addresses to MAC addresses. Usage: arp [options].

iwconfig: Displays and configures wireless network interfaces. Usage: iwconfig [interface].

iw: A more advanced tool for configuring and monitoring wireless network interfaces. Usage: iw [options].

ssh: Connects to a remote system securely using the SSH protocol. Usage: ssh [username@hostname].

wget: Downloads files from the internet via HTTP, HTTPS, or FTP protocols. Usage: wget [URL].

curl: Transfers data from or to a server using various protocols such as HTTP, HTTPS, FTP, etc. Usage: curl [URL].

tcpdump: A packet analyzer that captures and displays TCP/IP packets transmitted or received on a network interface. Usage: tcpdump [options] [filter expression].

route: Displays or modifies the kernel's IP routing table. Usage: route [options].

host: Provides DNS lookup and name resolution functionality. Usage: host [hostname].

nmap: A powerful network scanning tool that discovers hosts and services on a computer network, thus creating a map of the network. Usage: nmap [options] [target].

Hostname: The hostname command in Linux is used to view or set the hostname of the system.

What is nano

Nano is a simple and user-friendly text editor for Unix-like operating systems, including Linux. It is often used in terminal environments as a lightweight alternative to more complex text editors like Vim or Emacs.

Ctrl + G: Display the help menu, which provides a list of available commands and keyboard shortcuts.

Ctrl + X: Exit the editor. If changes have been made to the file, Nano will prompt to save them before exiting.

Ctrl + O: Write the current contents to a file (save).
Ctrl + R: Read a file into the current buffer. Useful for inserting the contents of another file into the current one.
Ctrl + W: Search for text within the file.
Ctrl + K: Cut the current line (delete the line and copy it to the clipboard).
Ctrl + U: Paste the contents of the clipboard at the cursor position.
Ctrl + A: Move the cursor to the beginning of the current line.
Ctrl + E: Move the cursor to the end of the current line.
Ctrl + Space: Set a mark (start selection).
Alt + A: Set the mark (start selection).
Alt + 6: Cut text from the current cursor position to the beginning of the line.
Ctrl + K: Cut text from the current cursor position to the end of the line.
Ctrl + ^: (Ctrl + Shift + 6) Copy the current line to the clipboard.
Alt + 6: Copy the current line to the clipboard.
Ctrl + J: Justify the current paragraph.
Ctrl + _: Move the cursor to a specific line and column.
Ctrl + N: Move to the next line.
Ctrl + P: Move to the previous line.
Alt + W: Find and replace text within the file.
Ctrl + T: Invoke the spell checker if available.
Alt + U: Undo the last action.
Alt + E: Redo the last undone action.

Job Management Commands
jobs: Displays a list of current jobs running in the background or suspended in the current shell session.
bg: Resumes a suspended job in the background. Usage: bg [job ID].
fg: Brings a background job to the foreground. Usage: fg [job ID].
Ctrl + Z: Suspends the currently running foreground process and puts it in the background.
kill: Terminates a process by sending a signal. Usage: kill [options] [PID].
killall: Terminates processes by name. Usage: killall [processname].
pkill: Sends signals to processes based on name or other attributes. Usage: pkill [options] [pattern].
pgrep: Searches for processes based on name or other attributes and displays their PIDs. Usage: pgrep [options] [pattern].
ps: Displays information about active processes. Usage: ps [options].
top: Displays real-time information about system resource usage, including CPU, memory, and processes. Usage: top.
timeout: Runs a command with a time limit. If the command doesn't complete within the specified time, it's terminated. Usage: timeout [options] [duration] [command].
renice: Changes the priority of a running process. Usage: renice [priority] [PID].

Process management commands
ps: Displays information about active processes. Usage: ps [options].
top: Displays real-time information about system resource usage, including CPU, memory, and processes. Usage: top.

htop: An interactive process viewer that provides a more user-friendly and feature-rich alternative to top. Usage: htop.
kill: Terminates a process by sending a signal. Usage: kill [options] [PID].
killall: Terminates processes by name. Usage: killall [processname].
pkill: Sends signals to processes based on name or other attributes. Usage: pkill [options] [pattern].
pgrep: Searches for processes based on name or other attributes and displays their PIDs. Usage: pgrep [options] [pattern].
kill -9 (SIGKILL): Forces termination of a process. Usage: kill -9 [PID].
kill -15 (SIGTERM): Sends a termination signal to a process, allowing it to exit gracefully. Usage: kill -15 [PID].
kill -STOP (SIGSTOP): Suspends a process. Usage: kill -STOP [PID].
kill -CONT (SIGCONT): Resumes a suspended process. Usage: kill -CONT [PID].
nice and renice: Adjusts the priority of a process. Usage: nice [command] or renice [priority] [PID].
nohup: Runs a command immune to hangups, with output to a non-tty. Usage: nohup [command].
disown: Removes jobs from the shell's job table, allowing them to continue running even after the shell session ends. Usage: disown [options] [job ID].
jobs: Displays a list of current jobs running in the background or suspended in the current shell session. Usage: jobs.

Shell scripting
Variables:
Variables are used to store data temporarily. They can be created and assigned values using the syntax variable_name=value. Example: name="John"

Comments:
Comments in shell scripts start with the # symbol and are ignored by the shell. They are used to add explanations or documentation to the script.

Command Substitution:
Command substitution allows the output of a command to replace the command itself within a script. It is denoted by enclosing the command in backticks (`) or $() syntax. Example: current_date=$(date +%Y-%m-%d)

Arithmetic Operations:
Shell supports basic arithmetic operations such as addition, subtraction, multiplication, and division using the (( )) syntax or expr command. Example: result=$((2 + 3)) or result=$(expr 2 + 3)

Conditional Statements:
Conditional statements allow executing different code blocks based on the evaluation of a condition. The if, elif, and else keywords are used for conditional branching. Example:

```
if [ condition ]; then
   # Code block to execute if condition is true
elif [ another_condition ]; then
   # Code block to execute if another_condition is true
```

```
else
    # Code block to execute if none of the conditions are true
Fi
```

Looping Constructs:
Looping constructs allow executing a block of code repeatedly. The for, while, and until loops are commonly used in shell scripting. Example:

```
for item in list; do
    # Code block to execute for each item in the list
done
```

Functions:
Functions allow encapsulating a block of code for reuse. They are defined using the function keyword or simply by their name followed by parentheses. Example:

```
function_name() {
    # Function body
}
```

Positional Parameters:
Positional parameters are variables that hold arguments passed to a shell script or function. They are referenced using $1, $2, etc., where $1 refers to the first argument, $2 refers to the second argument, and so on.

echo:
The echo command is used to display messages or values of variables.
Flags:
-n: Suppresses the trailing newline.
read:
The read command is used to read input from the user or from a file.
Flags:
-p: Displays a prompt message before reading input.
-r: Prevents backslashes from being interpreted as escape characters.
-s: Suppresses user input from being echoed to the terminal.
grep:
The grep command is used to search for patterns in text.
Flags:
-i: Ignores case sensitivity.
-v: Inverts the match to display lines that do not match the pattern.
sed:
The sed command is used for text stream processing or editing.
Flags:
-i: Performs in-place editing of the file.
-e: Specifies a script or command to be applied to the input.

Comparison Operators:
-eq: Equal to
-ne: Not equal to
-lt: Less than
-le: Less than or equal to
-gt: Greater than
-ge: Greater than or equal to
Logical Operators:
-a or &&: Logical AND
-o or ||: Logical OR
!: Logical NOT
String Comparisons:
=: Equal to (for string comparison)
!=: Not equal to (for string comparison)
-z: Checks if a string is empty
-n: Checks if a string is not empty
File Tests:
-e: Checks if a file exists
-f: Checks if a file exists and is a regular file
-d: Checks if a file exists and is a directory
-r: Checks if a file is readable
-w: Checks if a file is writable
-x: Checks if a file is executable
Arithmetic Evaluation:
-lt, -le, -gt, -ge, -eq, -ne: Used for numerical comparison within an arithmetic expression.
Pattern Matching:
*: Matches any string of characters
?: Matches any single character

echo
-n:
Suppresses the trailing newline character, causing echo not to output a newline after the specified string.
Example: echo -n "Hello, World"
-e:
Enables interpretation of backslash escapes in the string. This allows you to include special characters like newline (\n), tab (\t), etc.
Example: echo -e "Hello\nWorld"
-E:
Disables interpretation of backslash escapes in the string. This is useful if you want to print literal backslashes or avoid interpretation of special characters.
Example: echo -E "Hello\nWorld"

Pipe commands
grep:

Filters input based on a specified pattern or regular expression.
Example: command1 | grep "pattern"
sort:
Sorts lines of text alphabetically or numerically.
Example: command1 | sort
awk:
A powerful text processing tool that allows for pattern scanning and text manipulation.
Example: command1 | awk '{print $1}'
cut:
Extracts specific columns or fields from each line of input.
Example: command1 | cut -d ' ' -f 2
sed:
Stream editor for filtering and transforming text.
Example: command1 | sed 's/pattern/replacement/'
tr:
Translates or deletes characters in the input.
Example: command1 | tr '[:lower:]' '[:upper:]'
uniq:
Filters out adjacent matching lines from the input.
Example: command1 | uniq
wc:
Counts lines, words, or characters in the input.
Example: command1 | wc -l
tee:
Reads from standard input and writes to standard output and files simultaneously.
Example: command1 | tee output.txt
xargs:
Converts input from standard input into arguments to a command.
Example: command1 | xargs command2

sed command example

Search and Replace:
Replace "old_word" with "new_word" in a file named "file.txt" and output the result:
sed 's/old_word/new_word/g' file.txt

In-Place Editing:
Perform an in-place edit (modify the file directly) to replace "old_word" with "new_word" in a file named "file.txt":
sed -i 's/old_word/new_word/g' file.txt

Delete Lines Matching a Pattern:
Delete all lines containing the word "pattern" from a file named "file.txt":
sed '/pattern/d' file.txt

Print Specific Lines:
Print only lines 5 to 10 from a file named "file.txt":
sed -n '5,10p' file.txt

Print Lines Matching a Pattern:
Print only lines containing the word "pattern" from a file named "file.txt":
sed -n '/pattern/p' file.txt

Insert Text Before a Pattern:
Insert the text "new_line" before each line containing the word "pattern" in a file named "file.txt":
sed '/pattern/i\new_line' file.txt

Insert Text After a Pattern:
Insert the text "new_line" after each line containing the word "pattern" in a file named "file.txt":
sed '/pattern/a\new_line' file.txt

Delete Blank Lines:
Delete all blank lines from a file named "file.txt": sed '/^$/d' file.txt

Extract IP Addresses:
Extract all IP addresses from a file named "file.txt":sed -nE 's/.*([0-9]{1,3}\.){3}[0-9]{1,3}.*/\0/p' file.txt

Convert Tabs to Spaces:
Convert all tabs to spaces in a file named "file.txt":
sed 's/\t/ /g' file.txt

What is Perl
Perl is a high-level, general-purpose programming language that was originally developed by Larry Wall in the late 1980s. It is known for its powerful text-processing capabilities and is widely used for system administration, web development, network programming, and various other tasks. Perl is interpreted, dynamic, and supports multiple programming paradigms, including procedural, object-oriented, and functional programming.