Introduction to Linux – Practical Exercises

Subject : Shell Scripts I

Exercise 1: User Management

As a system administrator, you need to create a list of all users on the system for auditing purposes. Write a shell script that accepts an output filename as a command line argument. This script uses functions to extract and sort usernames from the "*/etc/passwd*" file and save the sorted list to the specified output file. If no output filename is provided, it defaults to "*user_list.txt.*"

Exercise 2: Log Analysis

As a system administrator, you want to find and count the number of occurrences of a specific error message in a system log file. Write a shell script that accepts the error message and log file as command line arguments. This script uses functions to find and count occurrences of the provided error message in the specified log file and prints the count to the console.

Exercise 3: Process Monitoring

You need to monitor the system for a specific process and take action if it's consuming too much CPU. Write a shell script that accepts the process name and a CPU threshold as command line arguments. This script uses functions to monitor the specified process's CPU usage using `ps` and, if it exceeds the provided threshold, gracefully stops the process using `kill`. Personalized information, such as the process name and CPU threshold, is taken as command line input.

Exercise 4: Disk Usage Reporting

As a system administrator, you want to generate a report of disk usage for all mounted filesystems on the server. Write a shell script that accepts an output filename as a command line argument. This script uses functions to gather disk usage information with `df`, sorts the results by disk space usage, and appends this report to the specified output file. If no output filename is provided, it defaults to "*disk_usage_report.txt.*"

Exercise 5: Log Rotation

You want to create a simple log rotation script to keep the size of a log file in check. Write a shell script that accepts the log file name and a size threshold as command line arguments. This script uses functions to check the size of the specified log file and, if it exceeds the provided threshold, renames the file to include a version number and creates a new empty log file to continue logging. Personalized information, such as the log file name and size threshold, is taken as command line input.

Exercise 6: Shell + SSH and SCP

Write a shell script that uses SSH to connect to a remote server and execute a command of your choice, such as listing the contents of a directory on the remote server and save its output in a file. After, the script should use SCP to make a copy of this file in your computer. Make sure to accept the remote server, username, and command as command line arguments to the script.

Exercise 7: Automated Remote Command Execution

Develop a script that takes a list of IP addresses from a file as an input parameter. For each IP address, connect to the corresponding remote server using SSH and execute a predefined command (e.g., "uptime"). Display the results for each server.

Exercise 8: Backup Script

Create a backup script that copies specified directories from a local machine to a remote server. Allow the user to input the local directories, remote server IP address, username, and the destination path on the remote server.

Exercise 9: IP List Parameter

Modify the script from Exercise 7 to accept an IP list file as a parameter. The script should read the IP addresses from the file and execute commands on each remote server.

Exercise 10: Remote System Information

Write a script that connects to a remote server using SSH and gathers information about the system, such as OS version, available memory, and disk usage. Display the collected information.

Exercise 11: Automated Software Deployment

Develop a script that deploys a software package to multiple remote servers. Allow the user to input the local path of the software package, a list of remote server IP addresses, username, and the destination path on each remote server.

Exercise 12: Dynamic SSH Configuration

Create a script that dynamically generates an SSH configuration file based on a provided list of remote servers. The script should accept an IP list file and generate SSH configurations for each server, making it easier to connect to them in the future.

Exercise 13: Remote Log Analysis

Write a script that connects to multiple remote servers using SSH, retrieves specified log files, and performs analysis (e.g., counting occurrences of specific events). Display the results for each server.