

**shell script to check tomcat is running or not.if not restart it.**

bash

Copy code

```
#!/bin/bash
```

```
# Define the path to your Tomcat installation
```

```
TOMCAT_HOME="/path/to/your/tomcat"
```

```
# Function to check if Tomcat is running
```

```
is_tomcat_running() {
```

```
    # Check if the process ID of Tomcat is running
```

```
    ps aux | grep catalina | grep -v grep
```

```
}
```

```
# Check if Tomcat is running
```

```
if is_tomcat_running; then
```

```
    echo "Tomcat is running."
```

```
else
```

```
    echo "Tomcat is not running. Restarting..."
```

```
# Restart Tomcat
```

```
$TOMCAT_HOME/bin/shutdown.sh
```

```
sleep 5
```

```
$TOMCAT_HOME/bin/startup.sh
```

```
# Check again if Tomcat is now running
```

```
if is_tomcat_running; then
```

```
    echo "Tomcat has been successfully restarted."
```

```
else
```

```
    echo "Failed to restart Tomcat. Please check the logs for more information."
```

```
fi
```

```
fi
```

Make sure to replace `"/path/to/your/tomcat"` with the actual path to your Tomcat installation directory. Save this script as, for example, `check_and_restart_tomcat.sh`, and then make it executable using the following command:

**bash**

**Copy code**

```
chmod +x check_and_restart_tomcat.sh
```

You can then run this script manually or set it up as a cron job to run at regular intervals. Keep in mind that you may need to adjust the script based on your specific Tomcat installation and environment.

-----

**User**

**What are the shell scripts you have written to automate the tasks using any scripting language?**

2 / 2

ChatGPT

I can provide you with examples of various shell scripts for different tasks. Below are a few examples:

1. Backup Script:

bash

Copy code

```
#!/bin/bash
```

```
# Backup script to archive files and directories
```

```
# Define variables
```

```
backup_dir="/path/to/backup"
```

```
source_dir="/path/to/source"
```

```
# Create backup directory if it doesn't exist
```

```
mkdir -p "$backup_dir"
```

```
# Create a timestamp for the backup file
```

```
timestamp=$(date +"%Y%m%d%H%M%S")
```

```
backup_file="$backup_dir/backup_$timestamp.tar.gz"
```

```
# Archive and compress files
```

```
tar -czf "$backup_file" "$source_dir"
```

```
echo "Backup completed: $backup_file"
```

2. File Cleanup Script:

```
bash
```

Copy code

```
#!/bin/bash
```

```
# Script to clean up files older than a certain number of days
```

```
# Define variables
```

```
target_dir="/path/to/target"
```

```
days_threshold=30
```

```
# Find and remove files older than the threshold
```

```
find "$target_dir" -type f -mtime +$days_threshold -exec rm {} \;
```

```
echo "File cleanup completed."
```

3. Website Availability Checker:

```
bash
```

Copy code

```
#!/bin/bash
```

# Script to check website availability

website="https://www.example.com"

timeout\_seconds=5

# Use curl to check the website

if curl --output /dev/null --silent --head --fail --max-time \$timeout\_seconds "\$website"; then

echo "Website is accessible."

else

echo "Website is not accessible. Taking action..."

# Add actions here, such as restarting services or sending notifications

fi

4. User Account Management:

bash

Copy code

#!/bin/bash

# Script to create a new user account

# Define variables

username="newuser"

password="password123"

# Check if the user already exists

if id "\$username" &>/dev/null; then

echo "User already exists."

else

# Create a new user

useradd -m -s /bin/bash "\$username"

echo "\$username:\$password" | chpasswd

echo "User \$username created with password \$password."

fi

Remember to customize these scripts according to your specific requirements and test them thoroughly in a safe environment before deploying them in a production environment.

---

User

- 1. write a shell script for fibonacci series?**
- 2. write a shell script to swap two variable values?**
- 3. write a shell script to to change the suffix of files?**
- 4. write a shell script to take dialy backup of files and store on s3?**
- 5. linux command to list files of only 4kb?**

ChatGPT

Certainly! Here are shell scripts for the tasks you've mentioned, along with a Linux command to list files of only 4KB:

1. Fibonacci Series Script:

bash

Copy code

```
#!/bin/bash
```

```
# Script to generate Fibonacci series
```

```
echo "Enter the number of terms for Fibonacci series:"
```

```
read n
```

```
a=0
```

```
b=1
```

```
echo "Fibonacci series up to $n terms:"
```

```
for ((i=0; i<n; i++)); do
```

```
    echo -n "$a "
```

```
    temp=$((a + b))
```

```
    a=$b
```

```
b=$temp
```

```
done
```

```
echo
```

## 2. Swap Two Variable Values Script:

```
bash
```

Copy code

```
#!/bin/bash
```

```
# Script to swap two variable values
```

```
echo "Enter the value of first variable (a):"
```

```
read a
```

```
echo "Enter the value of second variable (b):"
```

```
read b
```

```
echo "Before swapping: a = $a, b = $b"
```

```
# Swapping values
```

```
temp=$a
```

```
a=$b
```

```
b=$temp
```

```
echo "After swapping: a = $a, b = $b"
```

## 3. Change Suffix of Files Script:

```
bash
```

Copy code

```
#!/bin/bash
```

```
# Script to change the suffix of files in a directory
```

```
echo "Enter the directory path:"
```

```
read directory
```

```
echo "Enter the current suffix to replace:"
```

```
read current_suffix
```

```
echo "Enter the new suffix:"
```

```
read new_suffix
```

```
# Change suffix of files
```

```
for file in "$directory"/*"$current_suffix"; do
```

```
    mv "$file" "${file%"$current_suffix"}$new_suffix"
```

```
done
```

```
echo "Suffix changed successfully."
```

4. Daily Backup to S3 Script:

```
bash
```

Copy code

```
#!/bin/bash
```

```
# Script to take daily backup of files and store on S3
```

```
# Set AWS credentials and S3 bucket information
```

```
export AWS_ACCESS_KEY_ID="your_access_key_id"
```

```
export AWS_SECRET_ACCESS_KEY="your_secret_access_key"
```

```
bucket_name="your_s3_bucket_name"
```

```
# Set backup directory
```

```
backup_dir="/path/to/backup"
```

```
# Create a timestamp for the backup file
```

```
timestamp=$(date +"%Y%m%d%H%M%S")  
backup_file="$backup_dir/backup_$(date +"%Y%m%d%H%M%S").tar.gz"
```

# Archive and compress files

```
tar -czf "$backup_file" "/path/to/source"
```

# Upload the backup to S3

```
aws s3 cp "$backup_file" "s3://$bucket_name/"
```

```
echo "Backup completed and stored on S3: $backup_file"
```

Make sure to replace placeholder values (e.g., `your_access_key_id`, `your_secret_access_key`, `your_s3_bucket_name`) with your actual AWS credentials and S3 bucket information.

## 5. Linux Command to List Files of Only 4KB:

To list files of only 4KB in the current directory, you can use the `find` command:

```
bash
```

Copy code

```
find . -type f -size 4k
```

This command will list all regular files in the current directory (and its subdirectories) that are exactly 4KB in size. Adjust the path (`.` in this case) as needed.

User

Write Shell script for CPU memory usage

ChatGPT

Certainly! Below is a simple shell script that prints CPU and memory usage information:

```
bash
```

Copy code

```
#!/bin/bash
```

# Shell script to display CPU and memory usage



```
# Get CPU usage
```

```
cpu_usage=$(top -bn1 | grep "Cpu(s)" | awk '{print $2 + $4}')
```

```
# Get memory usage
```

```
memory_usage=$(free | awk '/Mem/{printf("%.2f"), $3/$2 * 100}')
```

```
echo "CPU Usage: $cpu_usage%"
```

```
echo "Memory Usage: $memory_usage%"
```

Save this script to a file, for example, `cpu_memory_usage.sh`. Make it executable using the following command:

```
bash
```

Copy code

```
chmod +x cpu_memory_usage.sh
```

You can then run the script:

```
bash
```

Copy code

```
./cpu_memory_usage.sh
```

This script uses `top` and `free` commands to obtain CPU and memory usage information, respectively. The CPU usage is calculated by summing the percentage of user and system CPU usage. The memory usage is calculated as a percentage of used memory out of the total available memory.

Note: Keep in mind that the specific commands and their output format might vary slightly depending on your system. It's always a good idea to test and adapt the script based on your system's characteristics.

---

**User**

**how will you create a 100 text files in a single linux command**

```
bash
```

Copy code

```
touch file{1..100}.txt
```

This command uses brace expansion {1..100} to generate a sequence of numbers from 1 to 100 and creates files named file1.txt, file2.txt, ..., file100.txt using the touch command.

After running this command, you'll have 100 empty text files in the current directory. Adjust the file names or extension as needed.

---

## User

**14. Write a shell script to list and show process resource utilization of 3rd and 6th position.**

**15. shell Script to delete the duplicate numbers in a set of numbers?**

14. Show Process Resource Utilization for 3rd and 6th Position:

bash

Copy code

```
#!/bin/bash
```

```
# Shell script to list and show process resource utilization of 3rd and 6th position
```

```
# Get the PIDs of the processes in 3rd and 6th position
```

```
pid3=$(ps aux --sort=-%cpu | awk 'NR==4 {print $2}')
```

```
pid6=$(ps aux --sort=-%cpu | awk 'NR==7 {print $2}')
```

```
# Display resource utilization for the processes
```

```
echo "Process in 3rd position (PID: $pid3):"
```

```
ps -p $pid3 -o pid,%cpu,%mem,cmd
```

```
echo "Process in 6th position (PID: $pid6):"
```

```
ps -p $pid6 -o pid,%cpu,%mem,cmd
```

This script uses the ps command to list processes sorted by CPU usage (--sort=-%cpu) and then extracts the PIDs of processes in the 3rd and 6th positions. Finally, it displays the resource utilization for those processes.

## 15. Delete Duplicate Numbers in a Set:

bash

Copy code

```
#!/bin/bash
```

```
# Shell script to delete duplicate numbers in a set of numbers
```

```
echo "Enter a set of numbers separated by spaces:"
```

```
read input_numbers
```

```
# Use awk to print unique numbers
```

```
unique_numbers=$(echo "$input_numbers" | awk '{for(i=1;i<=NF;i++)if(!a[$i]++)printf"%s ",$i}')
```

```
echo "Set after removing duplicates: $unique_numbers"
```

This script reads a set of numbers from the user, and then it uses awk to print unique numbers, effectively removing duplicates from the input set.

Make sure to save these scripts in separate files, make them executable using `chmod +x script_name.sh`, and then run them using `./script_name.sh`.

=====

**User**

**1. Write a shell script of hashmap?**

**2. Write a shell script to find max no. In sequence of number**

ChatGPT

Shell scripts are not the most suitable for implementing data structures like a hashmap. However, you can simulate a simple hashmap using associative arrays, which are available in some shells like Bash. Here's an example of a script simulating a hashmap:

## 1. Shell Script Simulating Hashmap:

bash

Copy code

```
#!/bin/bash
```

# Shell script simulating a simple hashmap using associative arrays

declare -A hashmap

# Add key-value pairs to the hashmap

hashmap["key1"]="value1"

hashmap["key2"]="value2"

hashmap["key3"]="value3"

# Access values by key

echo "Value for key1: \${hashmap["key1"]}"

echo "Value for key2: \${hashmap["key2"]}"

echo "Value for key3: \${hashmap["key3"]}"

# Print all keys and values

echo "All keys: \${!hashmap[@]}"

echo "All values: \${hashmap[@]}"

Note: Associative arrays are available in Bash version 4.0 and later.

2. Shell Script to Find Maximum Number in a Sequence:

bash

Copy code

#!/bin/bash

# Shell script to find the maximum number in a sequence of numbers

echo "Enter a sequence of numbers separated by spaces:"

read numbers

# Use awk to find the maximum number

```
max_number=$(echo "$numbers" | awk '{max=$1; for(i=2;i<=NF;i++) if($i>max) max=$i} END{print max}')
```

```
echo "Maximum number in the sequence: $max_number"
```

This script reads a sequence of numbers from the user, uses awk to find the maximum number, and then prints the result.

Make sure to save these scripts in separate files, make them executable using `chmod +x script_name.sh`, and then run them using `./script_name.sh`.

=====

**User**

**what is process substitution in shell script?**

ChatGPT

Process substitution is a feature in shell scripting that allows you to use the output of a command (or commands) as if it were a file or a set of files. It provides a convenient way to pass the output of a command directly to another command or program that expects input from a file.

In Bash and other compatible shells, process substitution is denoted by the use of `<()` or `>()` syntax. The basic idea is to treat the output of the enclosed command(s) as if it were a file that can be read from or written to.

Here are examples of both input and output process substitution:

Input Process Substitution (`<()`):

bash

Copy code

```
#!/bin/bash
```

```
# Example of input process substitution
```

```
# Compare two files using the 'diff' command
```

```
diff <(command1) <(command2)
```

In this example, `command1` and `command2` are executed, and their outputs are passed as input to the `diff` command for comparison.

Output Process Substitution (>()):

bash

Copy code

```
#!/bin/bash
```

```
# Example of output process substitution
```

```
# Redirect the output of a command to a file
```

```
command1 > >(tee output_file)
```

In this example, the output of `command1` is redirected to a file (`output_file`) using process substitution. The `tee` command is used to both display the output on the console and write it to the file.

Process substitution is particularly useful when you want to treat the output of a command as a file without the need to create a temporary file explicitly. It provides a cleaner and more efficient way to handle command output in certain scenarios.

=====

**User**

### **1. linux Command to display bottom 20 lines of a file?**

ChatGPT

To display the bottom 20 lines of a file in Linux, you can use the `tail` command with the `-n` option. Here's an example:

bash

Copy code

```
tail -n 20 filename
```

Replace `filename` with the actual name of the file you want to display. This command will show the last 20 lines of the specified file.

If you want to display the bottom 20 lines of a file without specifying the filename, you can use a pipe with the `tail` command:

bash

Copy code

```
some_command | tail -n 20
```

Replace some\_command with the command whose output you want to display the bottom 20 lines of.

=====

**User**

**What is grep and find command?**

ChatGPT

grep and find are two powerful command-line utilities in Unix-like operating systems, including Linux. They serve different purposes and are commonly used for searching and manipulating text files and directories.

grep Command:

grep (Global Regular Expression Print) is a command-line utility that searches for patterns in text files. It allows you to specify a pattern (regular expression) and then searches for lines in one or more files that match that pattern. Here's a basic example:

bash

Copy code

```
grep "pattern" filename
```

This command searches for the specified "pattern" in the contents of the "filename" and prints the matching lines.

Example:

bash

Copy code

```
grep "error" log.txt
```

This command would display all lines containing the word "error" in the file log.txt.

find Command:

The find command is used for searching files and directories in a directory hierarchy based on various criteria such as name, size, type, and more. It is a versatile tool for locating files that meet specific conditions. Here's a basic example:

bash

Copy code

```
find /path/to/search -name "filename"
```

This command searches for files with the specified name within the specified directory and its subdirectories.

Example:

bash

Copy code

```
find /home/user -name "*.txt"
```

This command would find all files with a .txt extension in the /home/user directory and its subdirectories.

Both grep and find are powerful tools that are commonly used in combination with other commands and utilities to perform complex text processing and file system operations on the command line.