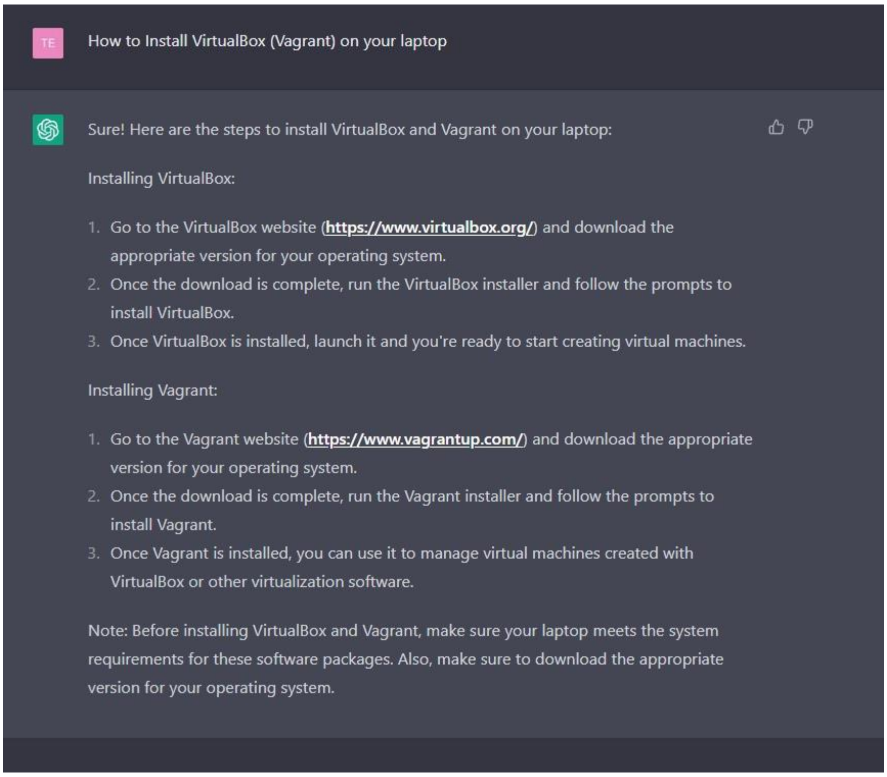
Linux Assignment Report

1. 1. Install VirtualBox (Vagrant) on your laptop, spin up centos 7, and Ubuntu 18.04 Machine.
   centos /7 and ubuntu/bionic64 are the Vagrant boxes that you need to download.



How to Install VirtualBox (Vagrant) on your laptop

Sure! Here are the steps to install VirtualBox and Vagrant on your laptop:

Installing VirtualBox:

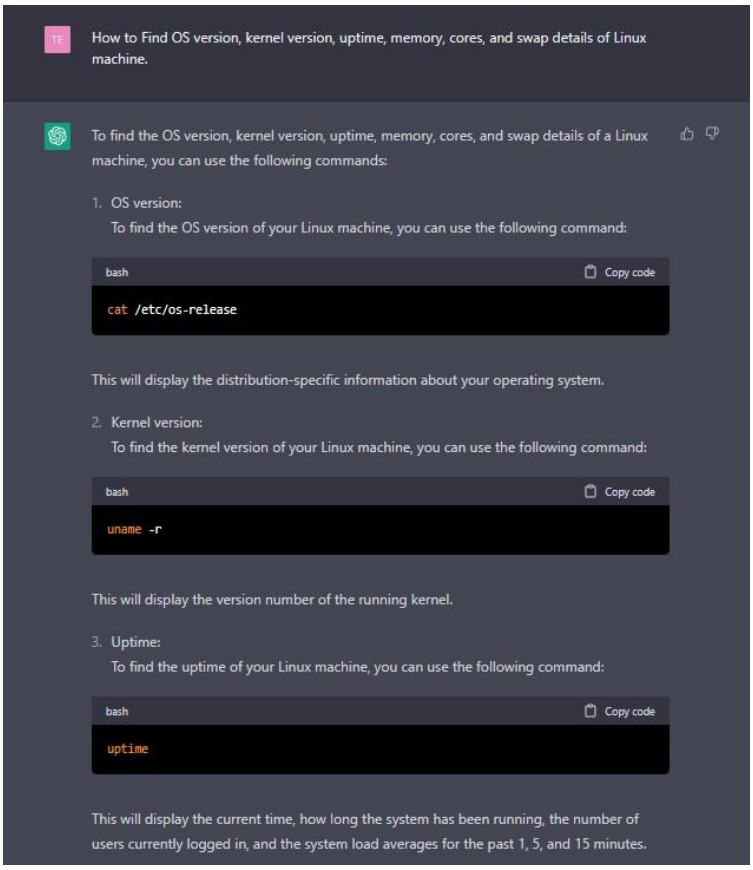
1. Go to the VirtualBox website (**https://www.virtualbox.org/**) and download the appropriate version for your operating system.
2. Once the download is complete, run the VirtualBox installer and follow the prompts to install VirtualBox.
3. Once VirtualBox is installed, launch it and you're ready to start creating virtual machines.

Installing Vagrant:

1. Go to the Vagrant website (**https://www.vagrantup.com/**) and download the appropriate version for your operating system.
2. Once the download is complete, run the Vagrant installer and follow the prompts to install Vagrant.
3. Once Vagrant is installed, you can use it to manage virtual machines created with VirtualBox or other virtualization software.

Note: Before installing VirtualBox and Vagrant, make sure your laptop meets the system requirements for these software packages. Also, make sure to download the appropriate version for your operating system.

2. Find OS version, kernel version, uptime, memory, cores, and swap details of Linux machine.



How to Find OS version, kernel version, uptime, memory, cores, and swap details of Linux machine.

To find the OS version, kernel version, uptime, memory, cores, and swap details of a Linux machine, you can use the following commands:

1. OS version:
   To find the OS version of your Linux machine, you can use the following command:

```bash
cat /etc/os-release
```

This will display the distribution-specific information about your operating system.

2. Kernel version:
   To find the kernel version of your Linux machine, you can use the following command:

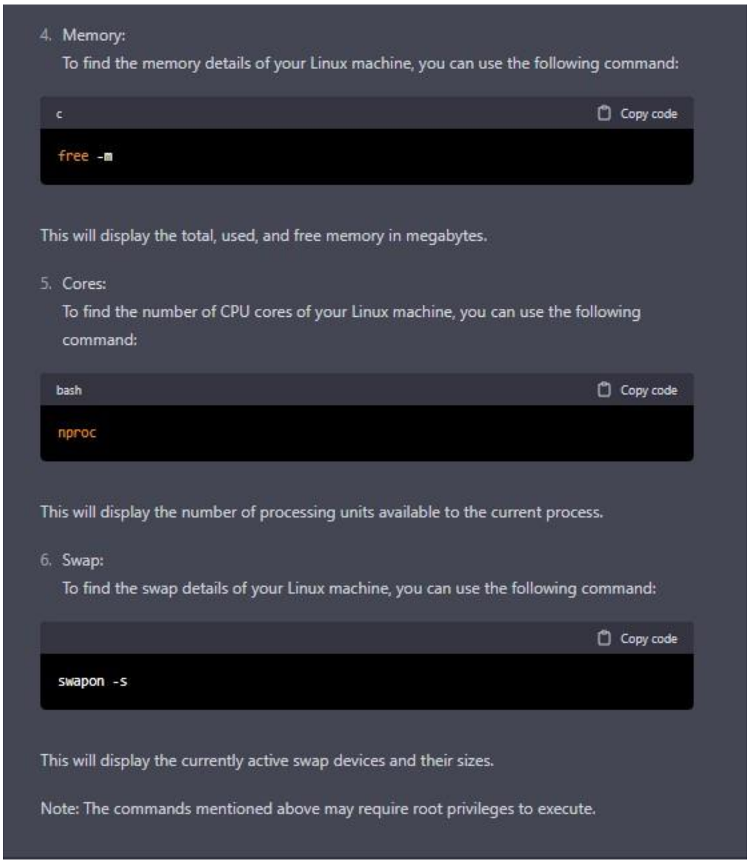
```bash
uname -r
```

This will display the version number of the running kernel.

3. Uptime:
   To find the uptime of your Linux machine, you can use the following command:

```bash
uptime
```

This will display the current time, how long the system has been running, the number of users currently logged in, and the system load averages for the past 1, 5, and 15 minutes.



4. Memory:
   To find the memory details of your Linux machine, you can use the following command:

```c
free -m
```

This will display the total, used, and free memory in megabytes.

5. Cores:
   To find the number of CPU cores of your Linux machine, you can use the following command:

```bash
nproc
```

This will display the number of processing units available to the current process.

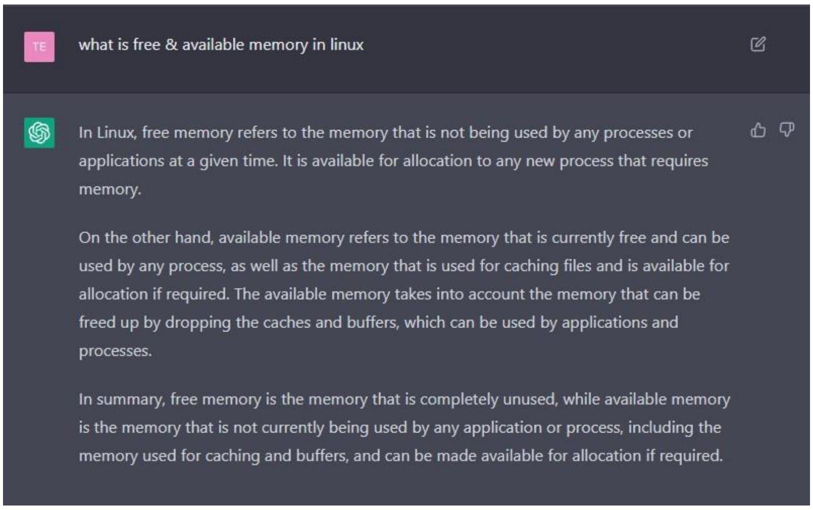
6. Swap:
   To find the swap details of your Linux machine, you can use the following command:

```
swapon -s
```

This will display the currently active swap devices and their sizes.

Note: The commands mentioned above may require root privileges to execute.

a. Be ready to explain what free & available memory is.



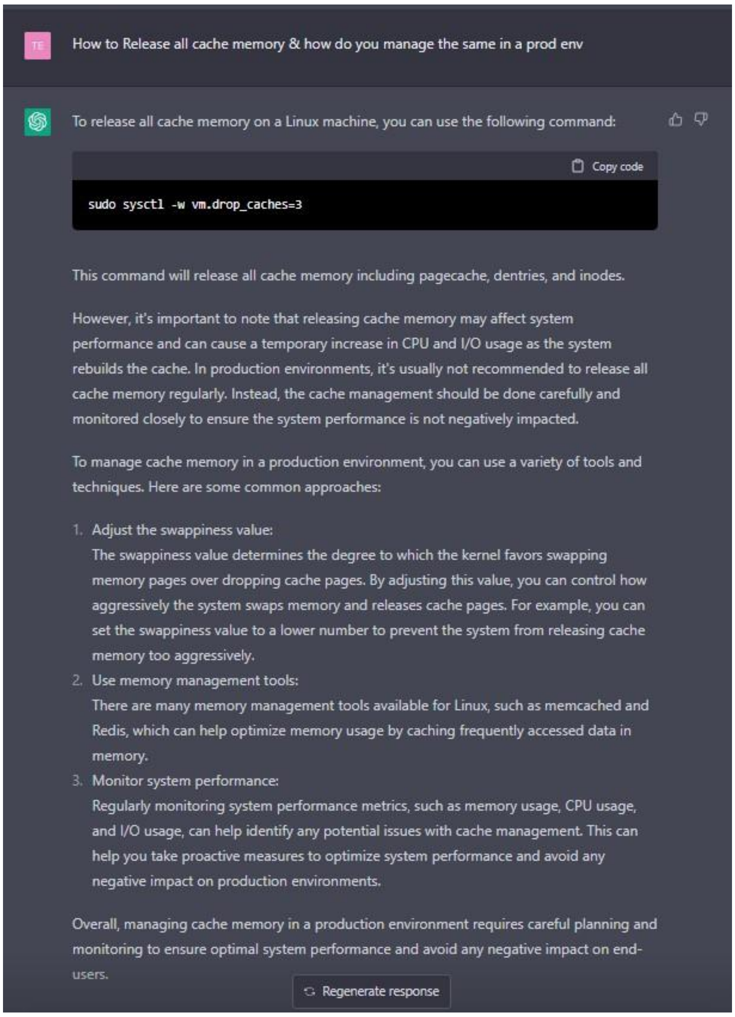
what is free & available memory in linux

In Linux, free memory refers to the memory that is not being used by any processes or applications at a given time. It is available for allocation to any new process that requires memory.

On the other hand, available memory refers to the memory that is currently free and can be used by any process, as well as the memory that is used for caching files and is available for allocation if required. The available memory takes into account the memory that can be freed up by dropping the caches and buffers, which can be used by applications and processes.

In summary, free memory is the memory that is completely unused, while available memory is the memory that is not currently being used by any application or process, including the memory used for caching and buffers, and can be made available for allocation if required.

b. Release all cache memory & how do you manage the same in a prod env.



How to Release all cache memory & how do you manage the same in a prod env

To release all cache memory on a Linux machine, you can use the following command:

```
sudo sysctl -w vm.drop_caches=3
```

This command will release all cache memory including pagecache, dentries, and inodes.

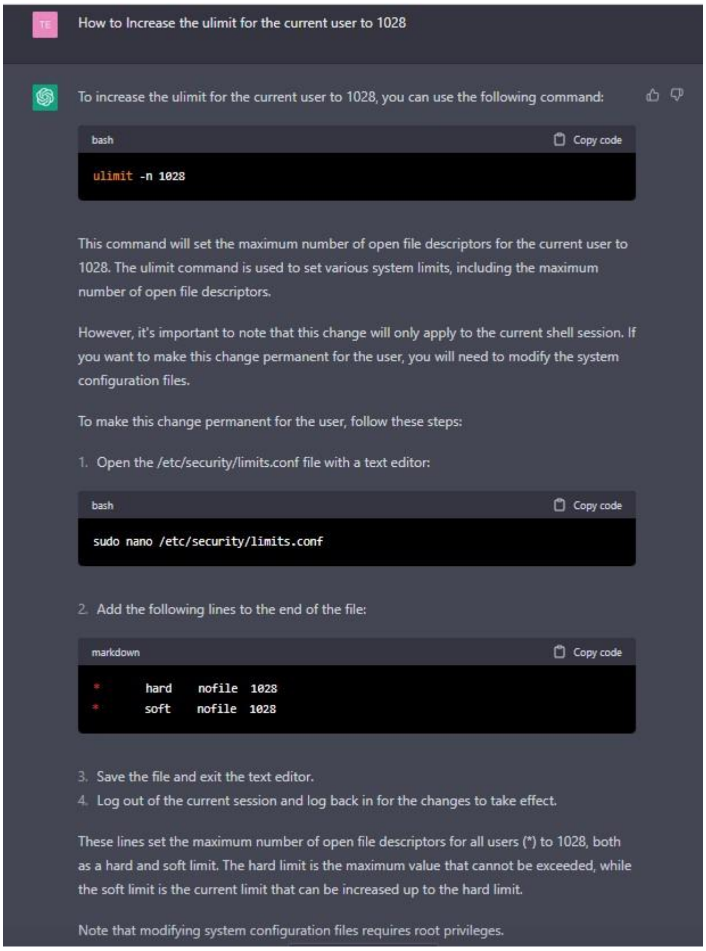
However, it's important to note that releasing cache memory may affect system performance and can cause a temporary increase in CPU and I/O usage as the system rebuilds the cache. In production environments, it's usually not recommended to release all cache memory regularly. Instead, the cache management should be done carefully and monitored closely to ensure the system performance is not negatively impacted.

To manage cache memory in a production environment, you can use a variety of tools and techniques. Here are some common approaches:

1. Adjust the swappiness value:
   The swappiness value determines the degree to which the kernel favors swapping memory pages over dropping cache pages. By adjusting this value, you can control how aggressively the system swaps memory and releases cache pages. For example, you can set the swappiness value to a lower number to prevent the system from releasing cache memory too aggressively.
2. Use memory management tools:
   There are many memory management tools available for Linux, such as memcached and Redis, which can help optimize memory usage by caching frequently accessed data in memory.
3. Monitor system performance:
   Regularly monitoring system performance metrics, such as memory usage, CPU usage, and I/O usage, can help identify any potential issues with cache management. This can help you take proactive measures to optimize system performance and avoid any negative impact on production environments.

Overall, managing cache memory in a production environment requires careful planning and monitoring to ensure optimal system performance and avoid any negative impact on end-users.

c. Increase the ulimit for the current user to 1028.



How to Increase the ulimit for the current user to 1028

To increase the ulimit for the current user to 1028, you can use the following command:

```bash
ulimit -n 1028
```

This command will set the maximum number of open file descriptors for the current user to 1028. The ulimit command is used to set various system limits, including the maximum number of open file descriptors.

However, it's important to note that this change will only apply to the current shell session. If you want to make this change permanent for the user, you will need to modify the system configuration files.

To make this change permanent for the user, follow these steps:

1. Open the /etc/security/limits.conf file with a text editor:

```bash
sudo nano /etc/security/limits.conf
```

2. Add the following lines to the end of the file:

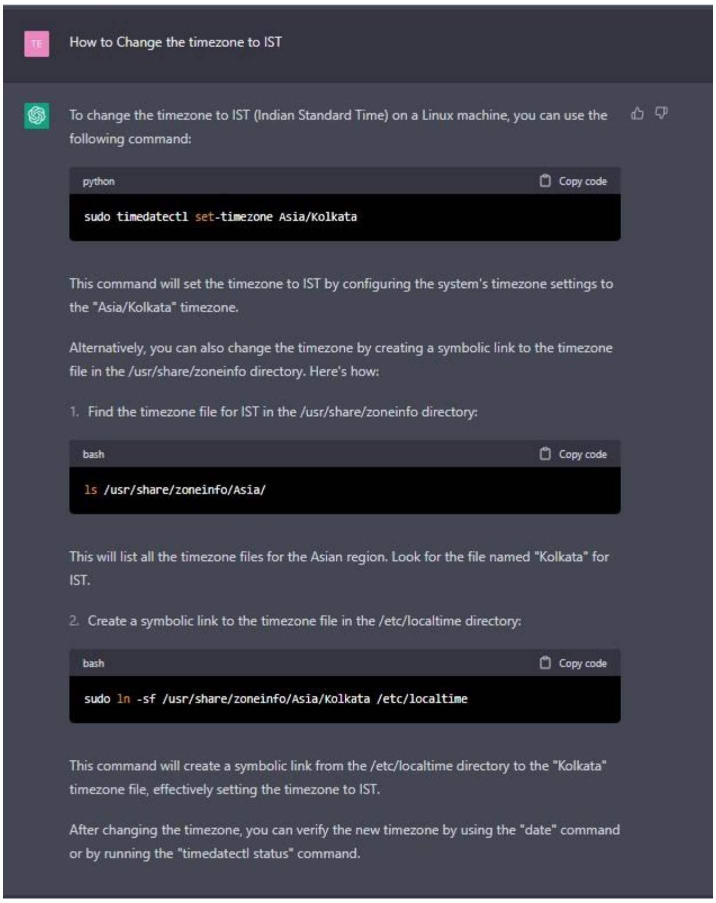
```markdown
*       hard    nofile  1028
*       soft    nofile  1028
```

3. Save the file and exit the text editor.
4. Log out of the current session and log back in for the changes to take effect.

These lines set the maximum number of open file descriptors for all users (*) to 1028, both as a hard and soft limit. The hard limit is the maximum value that cannot be exceeded, while the soft limit is the current limit that can be increased up to the hard limit.

Note that modifying system configuration files requires root privileges.

d. Change the timezone to IST



How to Change the timezone to IST

To change the timezone to IST (Indian Standard Time) on a Linux machine, you can use the following command:

```python
sudo timedatectl set-timezone Asia/Kolkata
```

This command will set the timezone to IST by configuring the system's timezone settings to the "Asia/Kolkata" timezone.

Alternatively, you can also change the timezone by creating a symbolic link to the timezone file in the /usr/share/zoneinfo directory. Here's how:

1. Find the timezone file for IST in the /usr/share/zoneinfo directory:

```bash
ls /usr/share/zoneinfo/Asia/
```

This will list all the timezone files for the Asian region. Look for the file named "Kolkata" for IST.

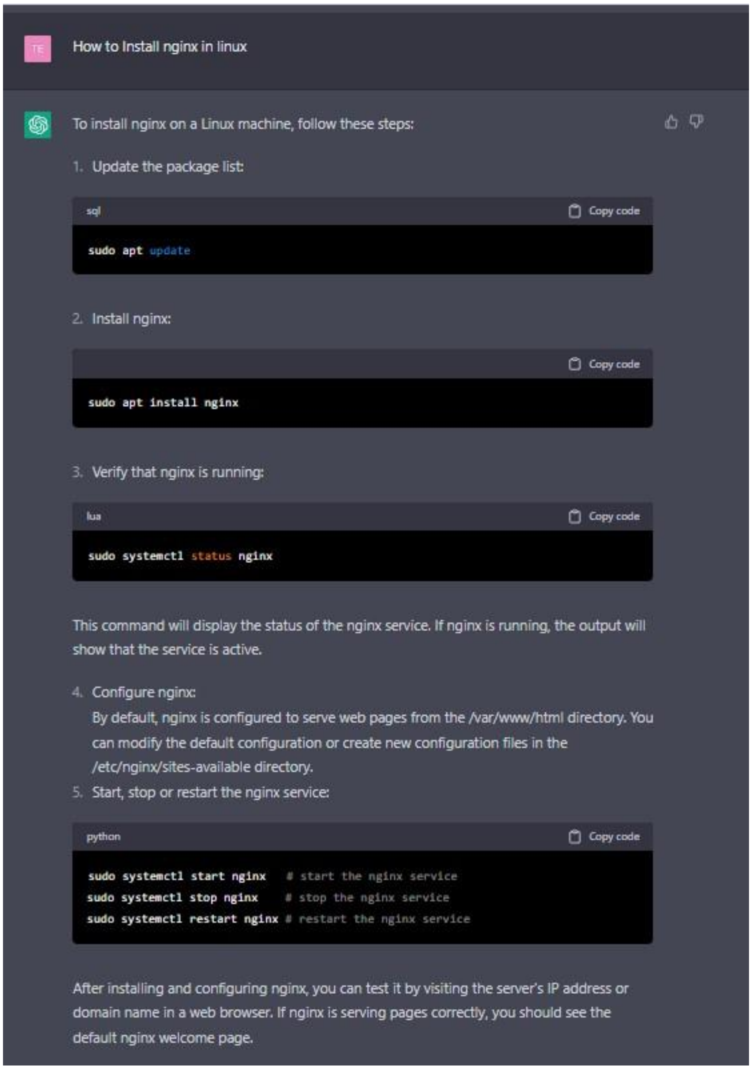
2. Create a symbolic link to the timezone file in the /etc/localtime directory:

```bash
sudo ln -sf /usr/share/zoneinfo/Asia/Kolkata /etc/localtime
```

This command will create a symbolic link from the /etc/localtime directory to the "Kolkata" timezone file, effectively setting the timezone to IST.

After changing the timezone, you can verify the new timezone by using the "date" command or by running the "timedatectl status" command.

3. Install nginx



How to Install nginx in linux

To install nginx on a Linux machine, follow these steps:

1. Update the package list:

```sql
sudo apt update
```

2. Install nginx:

```
sudo apt install nginx
```

3. Verify that nginx is running:

```lua
sudo systemctl status nginx
```

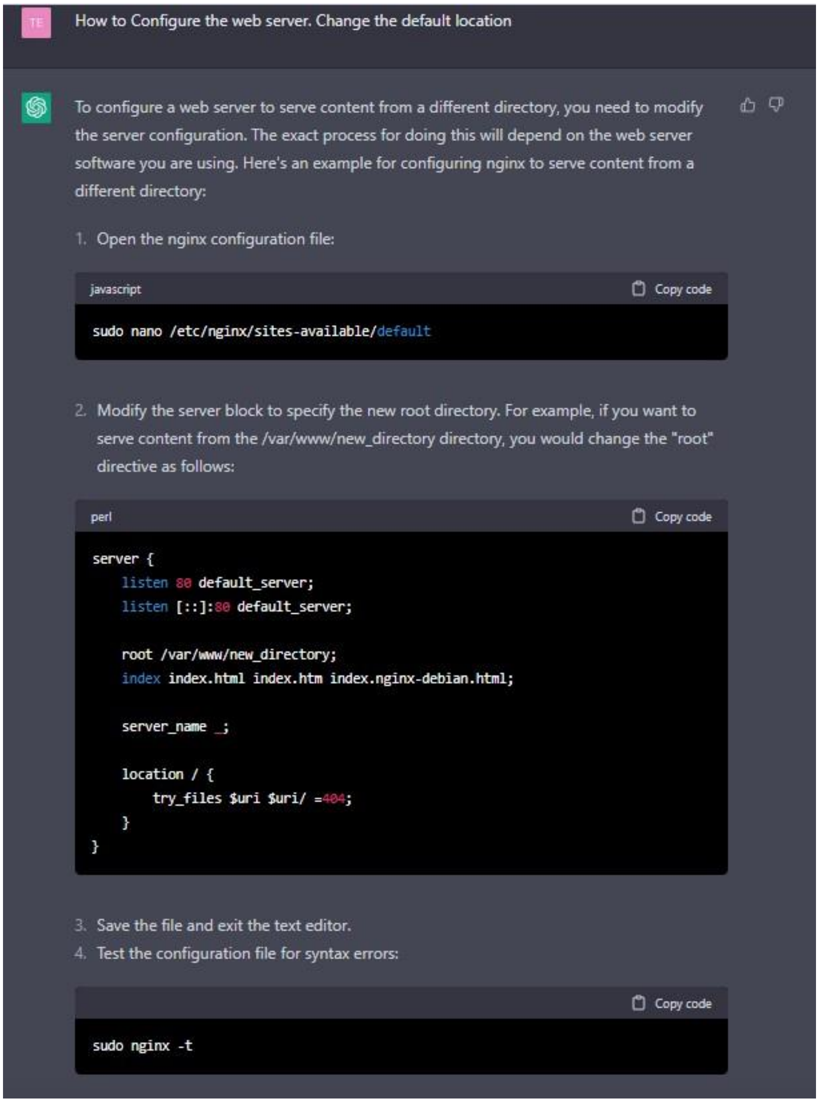
This command will display the status of the nginx service. If nginx is running, the output will show that the service is active.

4. Configure nginx:
   By default, nginx is configured to serve web pages from the /var/www/html directory. You can modify the default configuration or create new configuration files in the /etc/nginx/sites-available directory.
5. Start, stop or restart the nginx service:

```python
sudo systemctl start nginx   # start the nginx service
sudo systemctl stop nginx    # stop the nginx service
sudo systemctl restart nginx # restart the nginx service
```

After installing and configuring nginx, you can test it by visiting the server's IP address or domain name in a web browser. If nginx is serving pages correctly, you should see the default nginx welcome page.

a. Configure the web server. Change the default location



How to Configure the web server. Change the default location

To configure a web server to serve content from a different directory, you need to modify the server configuration. The exact process for doing this will depend on the web server software you are using. Here's an example for configuring nginx to serve content from a different directory:

1. Open the nginx configuration file:

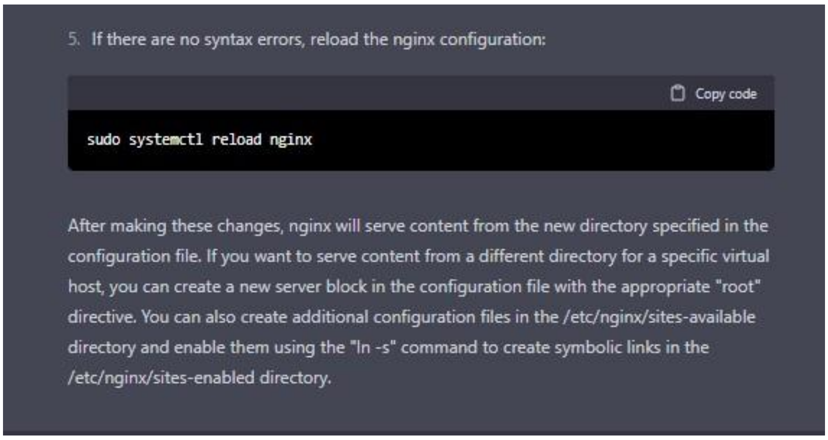
```
sudo nano /etc/nginx/sites-available/default
```

2. Modify the server block to specify the new root directory. For example, if you want to serve content from the /var/www/new_directory directory, you would change the "root" directive as follows:

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;

    root /var/www/new_directory;
    index index.html index.htm index.nginx-debian.html;

    server_name _;

    location / {
        try_files $uri $uri/ =404;
    }
}
```

3. Save the file and exit the text editor.
4. Test the configuration file for syntax errors:

```
sudo nginx -t
```

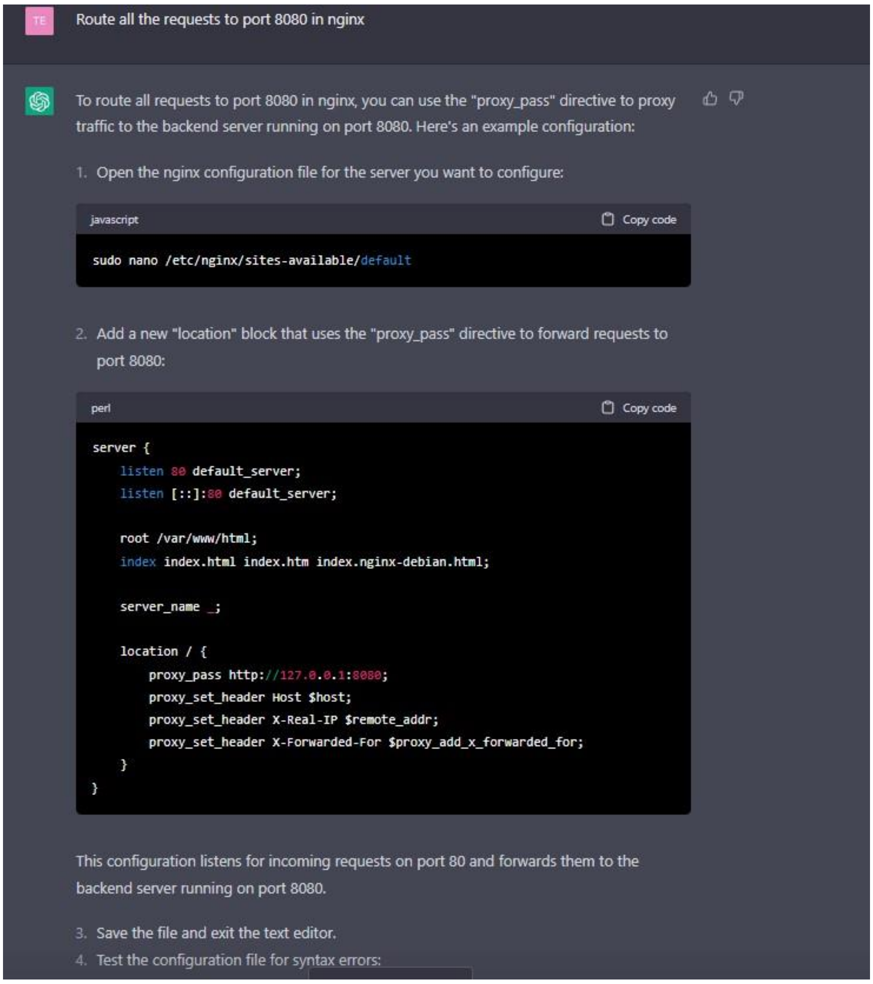


5. If there are no syntax errors, reload the nginx configuration:

```
sudo systemctl reload nginx
```

After making these changes, nginx will serve content from the new directory specified in the configuration file. If you want to serve content from a different directory for a specific virtual host, you can create a new server block in the configuration file with the appropriate "root" directive. You can also create additional configuration files in the /etc/nginx/sites-available directory and enable them using the "ln -s" command to create symbolic links in the /etc/nginx/sites-enabled directory.

b. Route all the requests to port 8080.



Route all the requests to port 8080 in nginx

To route all requests to port 8080 in nginx, you can use the "proxy_pass" directive to proxy traffic to the backend server running on port 8080. Here's an example configuration:

1. Open the nginx configuration file for the server you want to configure:

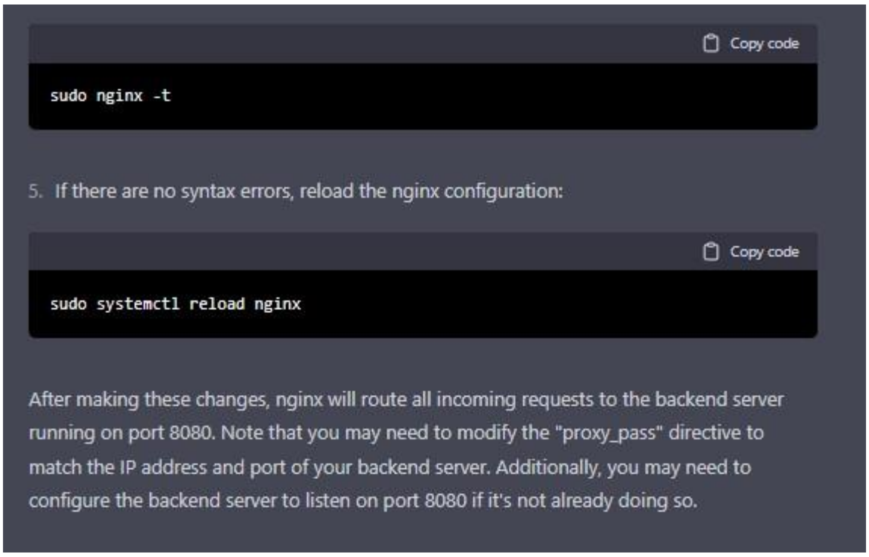
```javascript
sudo nano /etc/nginx/sites-available/default
```

2. Add a new "location" block that uses the "proxy_pass" directive to forward requests to port 8080:

```perl
server {
    listen 80 default_server;
    listen [::]:80 default_server;

    root /var/www/html;
    index index.html index.htm index.nginx-debian.html;

    server_name _;

    location / {
        proxy_pass http://127.0.0.1:8080;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }
}
```

This configuration listens for incoming requests on port 80 and forwards them to the backend server running on port 8080.

3. Save the file and exit the text editor.
4. Test the configuration file for syntax errors:



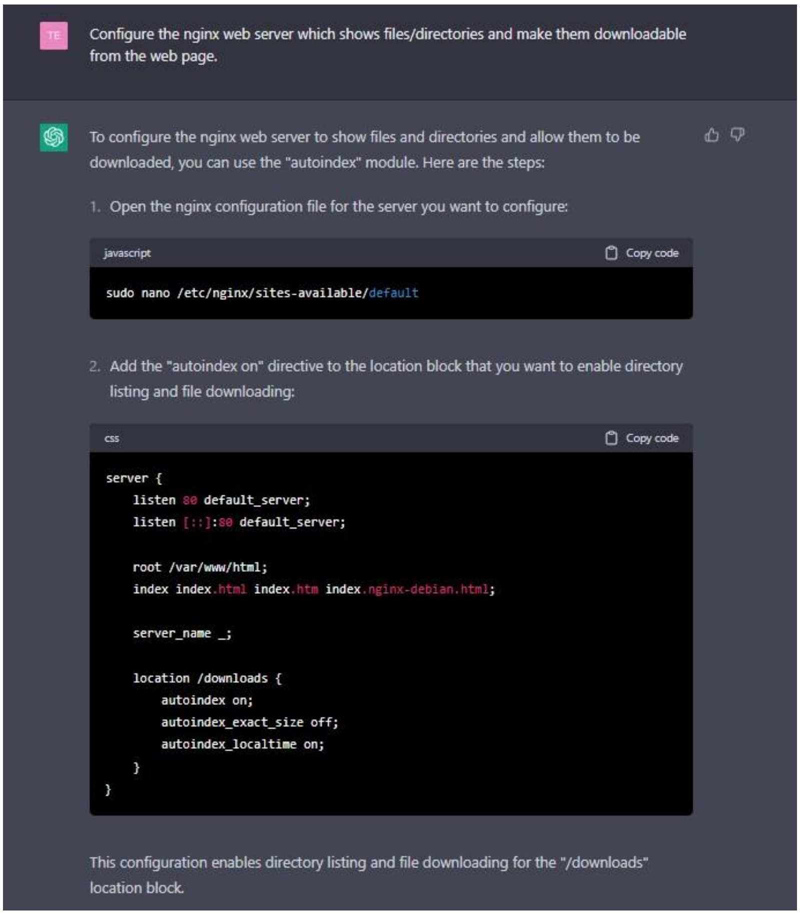
```
sudo nginx -t
```

5. If there are no syntax errors, reload the nginx configuration:

```
sudo systemctl reload nginx
```

After making these changes, nginx will route all incoming requests to the backend server running on port 8080. Note that you may need to modify the "proxy_pass" directive to match the IP address and port of your backend server. Additionally, you may need to configure the backend server to listen on port 8080 if it's not already doing so.

c. Configure the web server which shows files/directories and make them downloadable from the web page.



Configure the nginx web server which shows files/directories and make them downloadable from the web page.

To configure the nginx web server to show files and directories and allow them to be downloaded, you can use the "autoindex" module. Here are the steps:

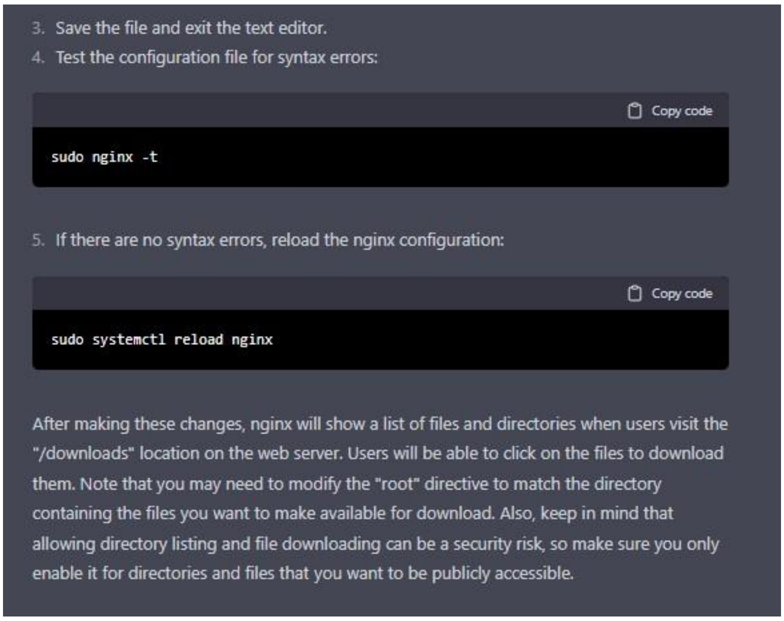
1. Open the nginx configuration file for the server you want to configure:

```javascript
sudo nano /etc/nginx/sites-available/default
```

2. Add the "autoindex on" directive to the location block that you want to enable directory listing and file downloading:

```css
server {
    listen 80 default_server;
    listen [::]:80 default_server;

    root /var/www/html;
    index index.html index.htm index.nginx-debian.html;

    server_name _;

    location /downloads {
        autoindex on;
        autoindex_exact_size off;
        autoindex_localtime on;
    }
}
```

This configuration enables directory listing and file downloading for the "/downloads" location block.



3. Save the file and exit the text editor.
4. Test the configuration file for syntax errors:

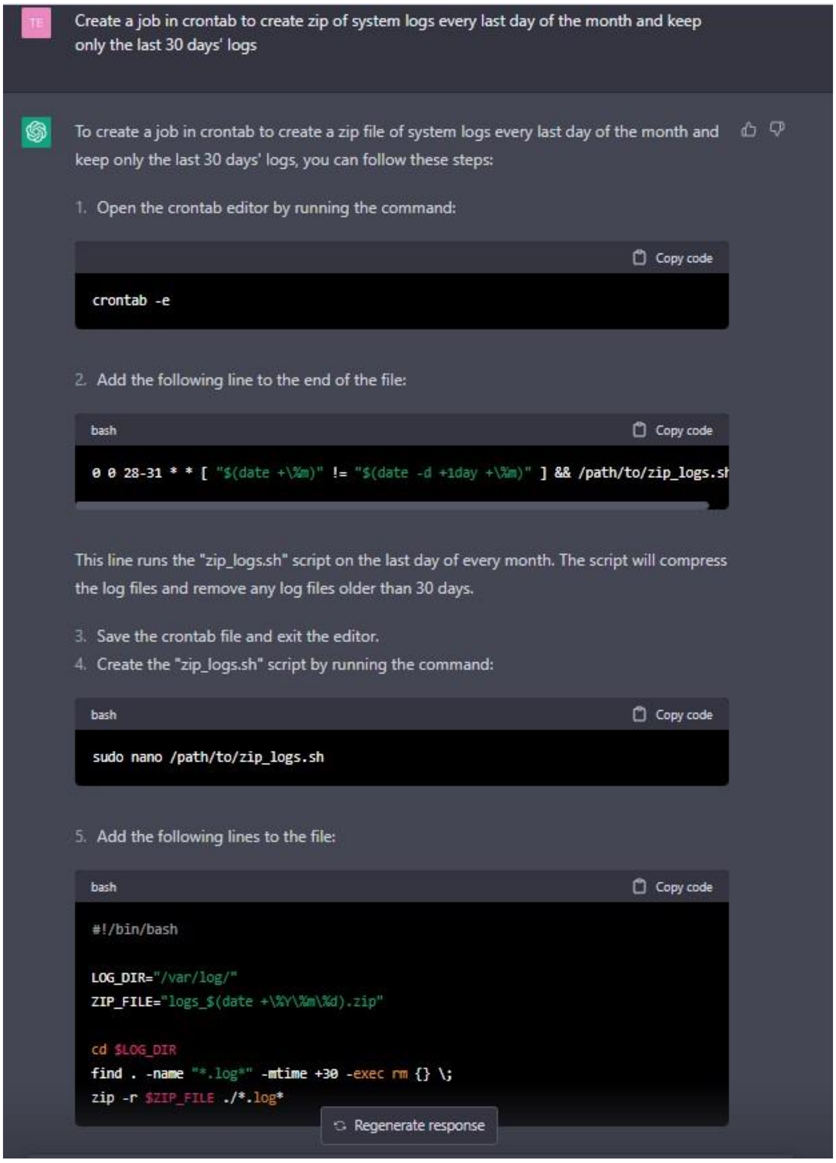
```
sudo nginx -t
```

5. If there are no syntax errors, reload the nginx configuration:

```
sudo systemctl reload nginx
```

After making these changes, nginx will show a list of files and directories when users visit the "/downloads" location on the web server. Users will be able to click on the files to download them. Note that you may need to modify the "root" directive to match the directory containing the files you want to make available for download. Also, keep in mind that allowing directory listing and file downloading can be a security risk, so make sure you only enable it for directories and files that you want to be publicly accessible.

4. Create a job in crontab to create zip of system logs every last day of the month and keep only the last 30 days’ logs.



Create a job in crontab to create zip of system logs every last day of the month and keep only the last 30 days’ logs

To create a job in crontab to create a zip file of system logs every last day of the month and keep only the last 30 days' logs, you can follow these steps:

1. Open the crontab editor by running the command:

```
crontab -e
```

2. Add the following line to the end of the file:

```
0 0 28-31 * * [ "$(date +\%m)" != "$(date -d +1day +\%m)" ] && /path/to/zip_logs.s
```

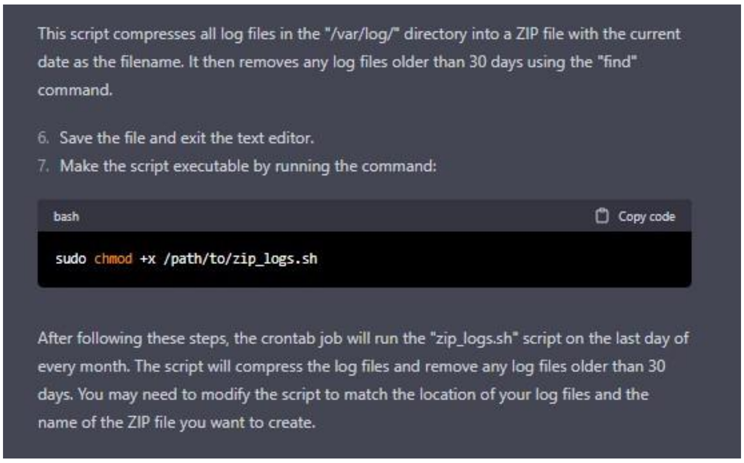
This line runs the "zip_logs.sh" script on the last day of every month. The script will compress the log files and remove any log files older than 30 days.

3. Save the crontab file and exit the editor.
4. Create the "zip_logs.sh" script by running the command:

```
sudo nano /path/to/zip_logs.sh
```

5. Add the following lines to the file:

```
#!/bin/bash

LOG_DIR="/var/log/"
ZIP_FILE="logs_$(date +\%Y\%m\%d).zip"

cd $LOG_DIR
find . -name "*.log*" -mtime +30 -exec rm {} \;
zip -r $ZIP_FILE ./*.log*
```



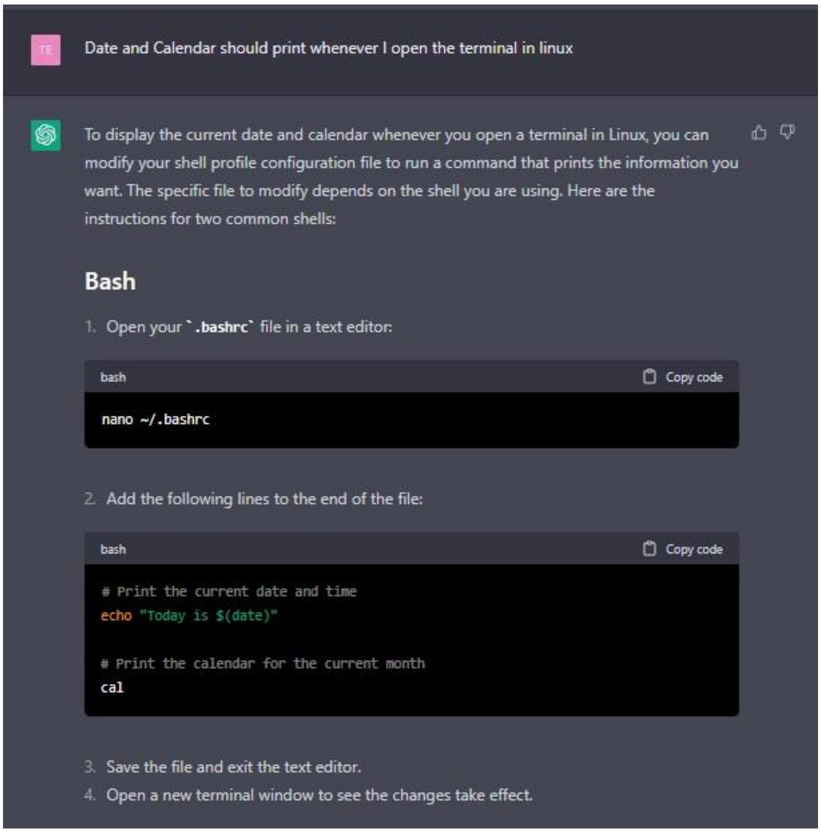
This script compresses all log files in the "/var/log/" directory into a ZIP file with the current date as the filename. It then removes any log files older than 30 days using the "find" command.

6. Save the file and exit the text editor.
7. Make the script executable by running the command:

```
sudo chmod +x /path/to/zip_logs.sh
```

After following these steps, the crontab job will run the "zip_logs.sh" script on the last day of every month. The script will compress the log files and remove any log files older than 30 days. You may need to modify the script to match the location of your log files and the name of the ZIP file you want to create.

5. Date and Calendar should print whenever I open the terminal



Date and Calendar should print whenever I open the terminal in linux

To display the current date and calendar whenever you open a terminal in Linux, you can modify your shell profile configuration file to run a command that prints the information you want. The specific file to modify depends on the shell you are using. Here are the instructions for two common shells:

Bash

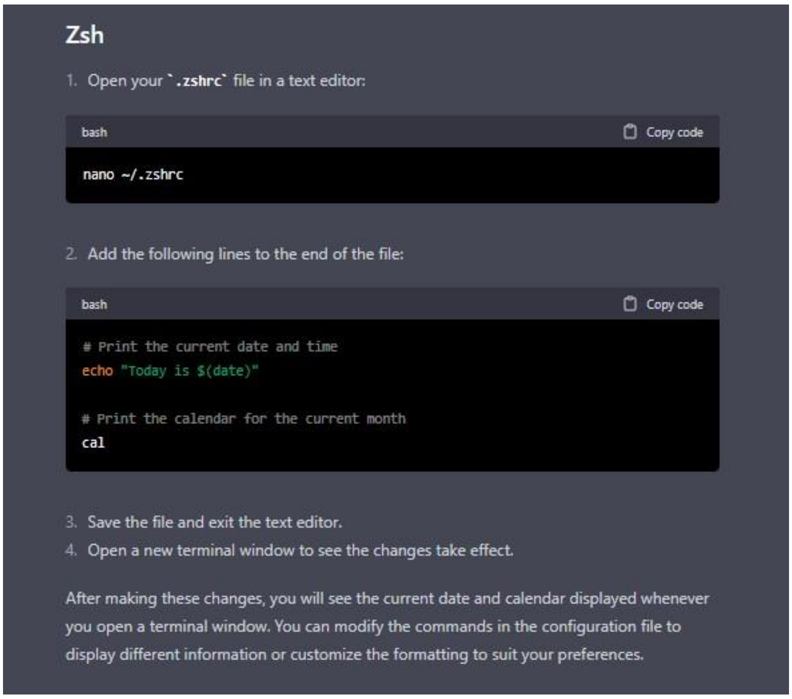
1. Open your `.bashrc` file in a text editor:

```bash
nano ~/.bashrc
```

2. Add the following lines to the end of the file:

```bash
# Print the current date and time
echo "Today is $(date)"

# Print the calendar for the current month
cal
```

3. Save the file and exit the text editor.
4. Open a new terminal window to see the changes take effect.



Zsh

1. Open your `.zshrc` file in a text editor:

```bash
nano ~/.zshrc
```

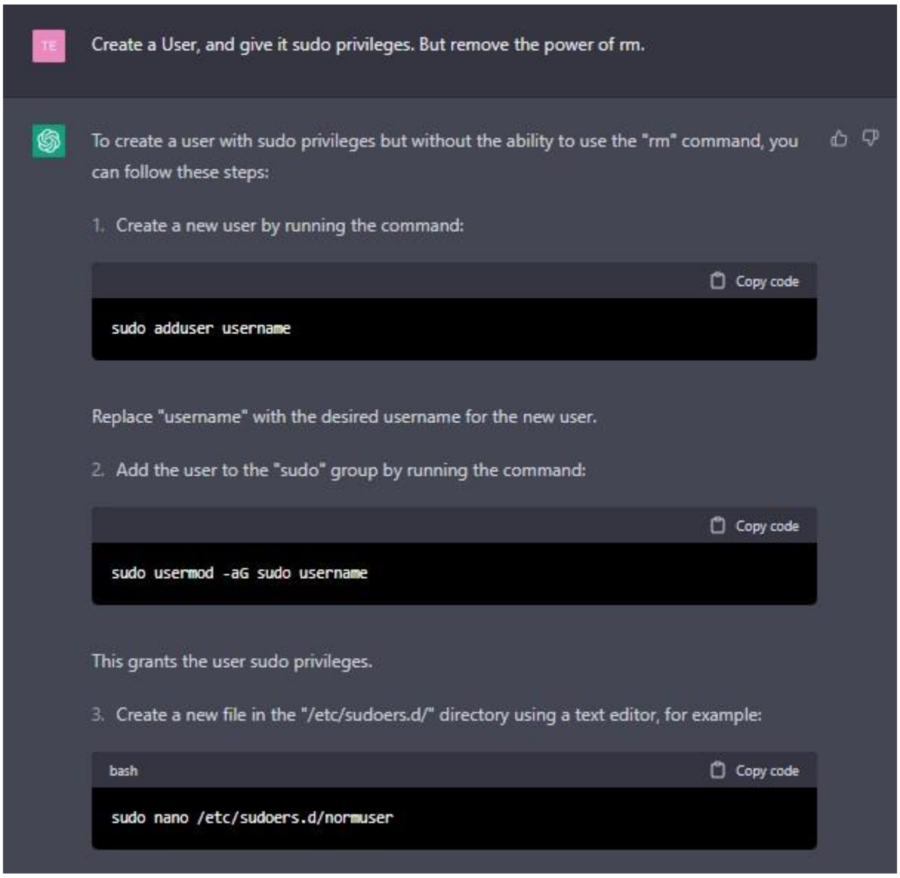
2. Add the following lines to the end of the file:

```bash
# Print the current date and time
echo "Today is $(date)"

# Print the calendar for the current month
cal
```

3. Save the file and exit the text editor.
4. Open a new terminal window to see the changes take effect.

After making these changes, you will see the current date and calendar displayed whenever you open a terminal window. You can modify the commands in the configuration file to display different information or customize the formatting to suit your preferences.

6. Create a User, and give it sudo privileges. But remove the power of rm.



Create a User, and give it sudo privileges. But remove the power of rm.

To create a user with sudo privileges but without the ability to use the "rm" command, you can follow these steps:

1. Create a new user by running the command:

```
sudo adduser username
```

Replace "username" with the desired username for the new user.

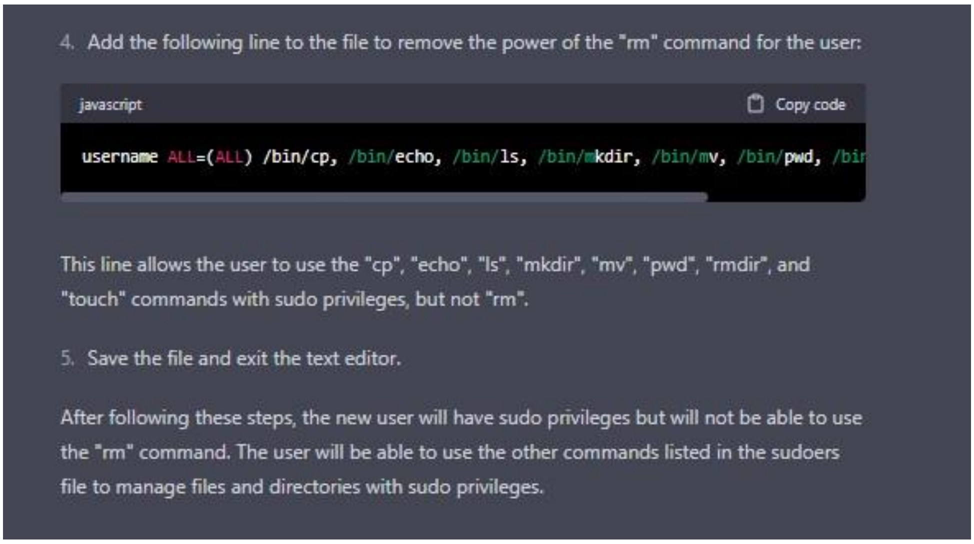
2. Add the user to the "sudo" group by running the command:

```
sudo usermod -aG sudo username
```

This grants the user sudo privileges.

3. Create a new file in the "/etc/sudoers.d/" directory using a text editor, for example:

```bash
sudo nano /etc/sudoers.d/normuser
```



4. Add the following line to the file to remove the power of the "rm" command for the user:

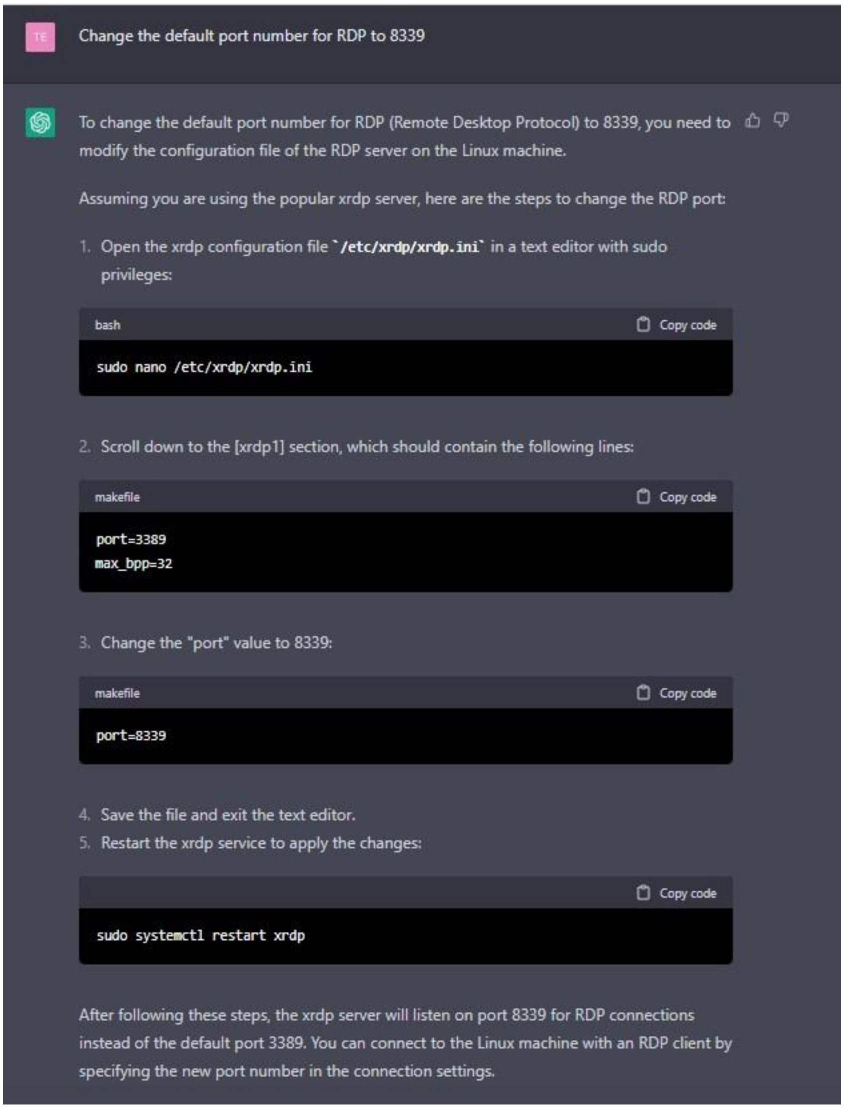
```javascript
username ALL=(ALL) /bin/cp, /bin/echo, /bin/ls, /bin/mkdir, /bin/mv, /bin/pwd, /bi
```

This line allows the user to use the "cp", "echo", "ls", "mkdir", "mv", "pwd", "rmdir", and "touch" commands with sudo privileges, but not "rm".

5. Save the file and exit the text editor.

After following these steps, the new user will have sudo privileges but will not be able to use the "rm" command. The user will be able to use the other commands listed in the sudoers file to manage files and directories with sudo privileges.

7. Change the default port number for RDP to 8339 and document on the same.



Change the default port number for RDP to 8339

To change the default port number for RDP (Remote Desktop Protocol) to 8339, you need to modify the configuration file of the RDP server on the Linux machine.

Assuming you are using the popular xrdp server, here are the steps to change the RDP port:

1. Open the xrdp configuration file `/etc/xrdp/xrdp.ini` in a text editor with sudo privileges:

```bash
sudo nano /etc/xrdp/xrdp.ini
```

2. Scroll down to the [xrdp1] section, which should contain the following lines:

```makefile
port=3389
max_bpp=32
```

3. Change the "port" value to 8339:

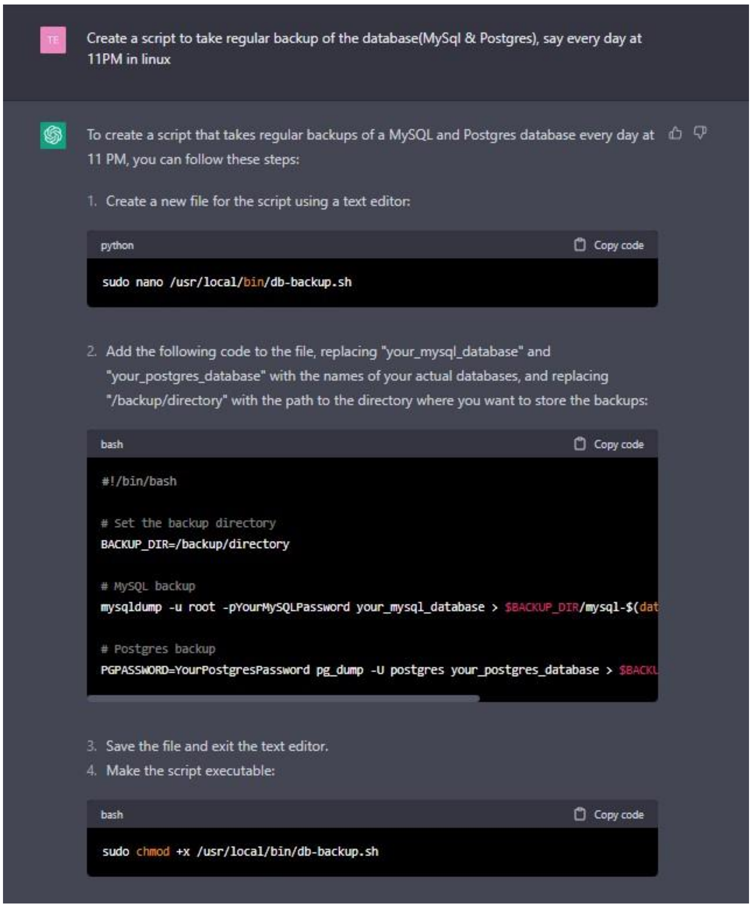
```makefile
port=8339
```

4. Save the file and exit the text editor.
5. Restart the xrdp service to apply the changes:

```
sudo systemctl restart xrdp
```

After following these steps, the xrdp server will listen on port 8339 for RDP connections instead of the default port 3389. You can connect to the Linux machine with an RDP client by specifying the new port number in the connection settings.

8. Create a script to take regular backup of the database(MySql & Postgres), say every day at 11PM.



Create a script to take regular backup of the database(MySql & Postgres), say every day at 11PM in linux

To create a script that takes regular backups of a MySQL and Postgres database every day at 11 PM, you can follow these steps:

1. Create a new file for the script using a text editor:

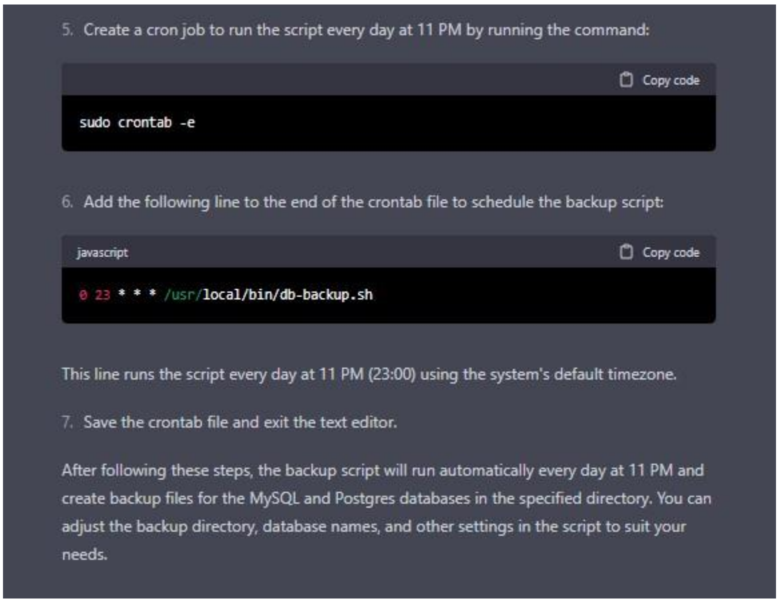
```
sudo nano /usr/local/bin/db-backup.sh
```

2. Add the following code to the file, replacing "your_mysql_database" and "your_postgres_database" with the names of your actual databases, and replacing "/backup/directory" with the path to the directory where you want to store the backups:

```
#!/bin/bash

# Set the backup directory
BACKUP_DIR=/backup/directory

# MySQL backup
mysqldump -u root -pYourMySQLPassword your_mysql_database > $BACKUP_DIR/mysql-$(dat

# Postgres backup
PGPASSWORD=YourPostgresPassword pg_dump -U postgres your_postgres_database > $BACKU
```

3. Save the file and exit the text editor.
4. Make the script executable:

```
sudo chmod +x /usr/local/bin/db-backup.sh
```



5. Create a cron job to run the script every day at 11 PM by running the command:

```
sudo crontab -e
```

6. Add the following line to the end of the crontab file to schedule the backup script:

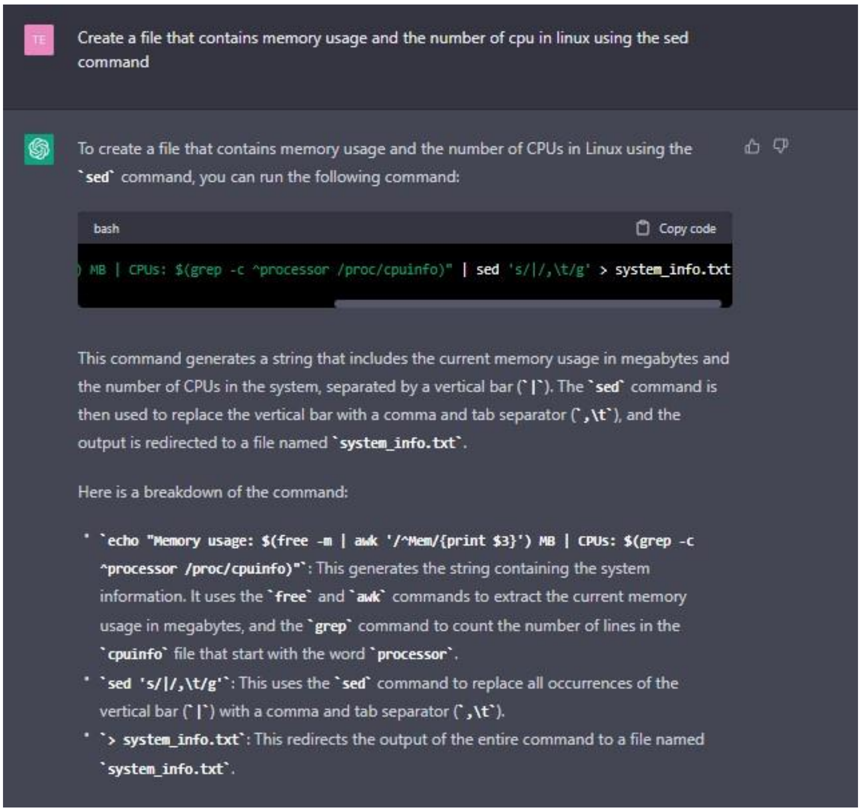
```
0 23 * * * /usr/local/bin/db-backup.sh
```

This line runs the script every day at 11 PM (23:00) using the system's default timezone.

7. Save the crontab file and exit the text editor.

After following these steps, the backup script will run automatically every day at 11 PM and create backup files for the MySQL and Postgres databases in the specified directory. You can adjust the backup directory, database names, and other settings in the script to suit your needs.

9. Create a file that contains memory usage and the number of cpu in linux using the sed command.



TE

Create a file that contains memory usage and the number of cpu in linux using the sed command

To create a file that contains memory usage and the number of CPUs in Linux using the `sed` command, you can run the following command:

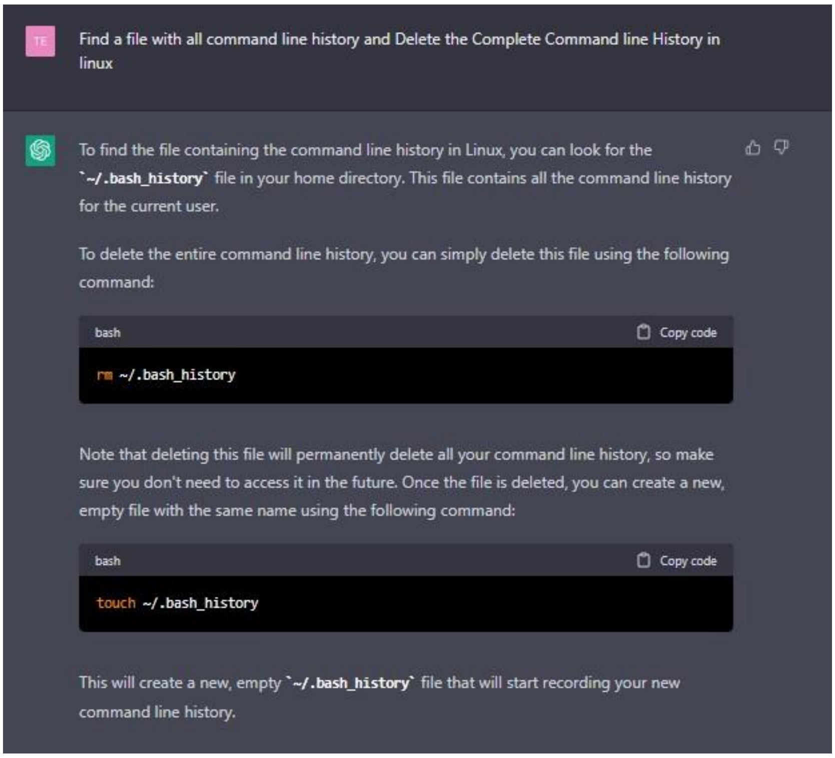
```bash
) MB | CPUs: $(grep -c ^processor /proc/cpuinfo)" | sed 's/|/,\t/g' > system_info.txt
```

This command generates a string that includes the current memory usage in megabytes and the number of CPUs in the system, separated by a vertical bar (`|`). The `sed` command is then used to replace the vertical bar with a comma and tab separator (`,\t`), and the output is redirected to a file named `system_info.txt`.

Here is a breakdown of the command:

- `echo "Memory usage: $(free -m | awk '/^Mem/{print $3}') MB | CPUs: $(grep -c ^processor /proc/cpuinfo)"`: This generates the string containing the system information. It uses the `free` and `awk` commands to extract the current memory usage in megabytes, and the `grep` command to count the number of lines in the `cpuinfo` file that start with the word `processor`.
- `sed 's/|/,\t/g'`: This uses the `sed` command to replace all occurrences of the vertical bar (`|`) with a comma and tab separator (`,\t`).
- `> system_info.txt`: This redirects the output of the entire command to a file named `system_info.txt`.

10. Find a file with all command line history and Delete the Complete Command line History.



TE

Find a file with all command line history and Delete the Complete Command line History in linux

To find the file containing the command line history in Linux, you can look for the `~/.bash_history` file in your home directory. This file contains all the command line history for the current user.

To delete the entire command line history, you can simply delete this file using the following command:

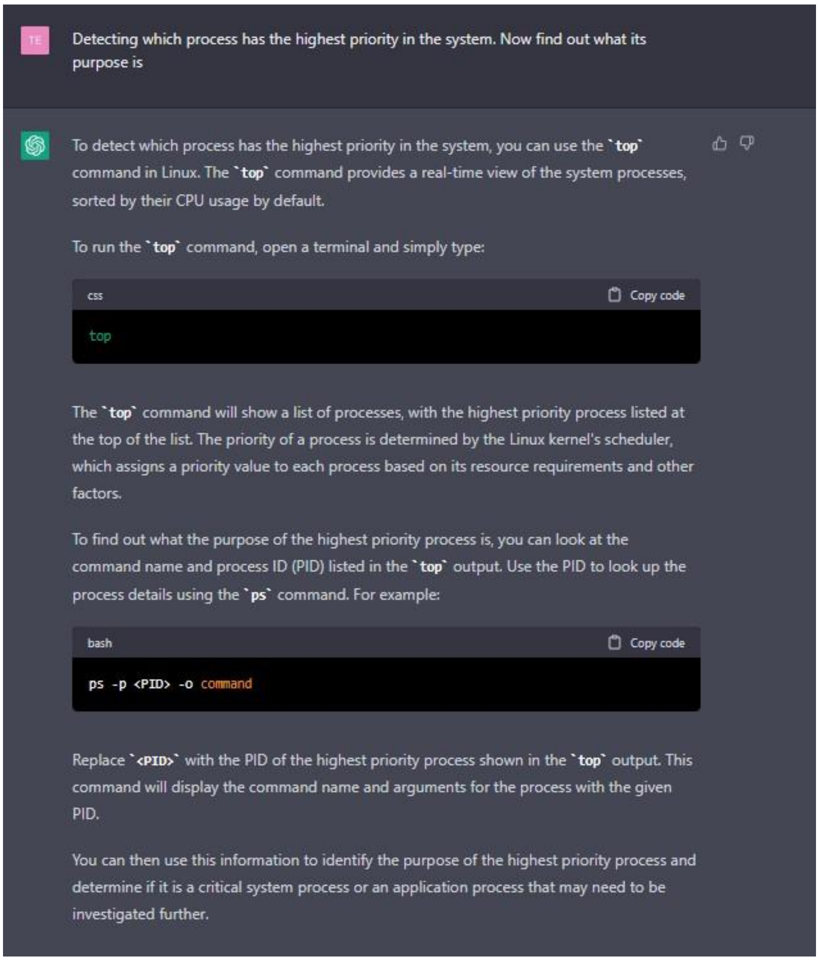
```bash
rm ~/.bash_history
```

Note that deleting this file will permanently delete all your command line history, so make sure you don't need to access it in the future. Once the file is deleted, you can create a new, empty file with the same name using the following command:

```bash
touch ~/.bash_history
```

This will create a new, empty `~/.bash_history` file that will start recording your new command line history.

11. Detecting which process has the highest priority in the system. Now find out what its purpose is.



Detecting which process has the highest priority in the system. Now find out what its purpose is

To detect which process has the highest priority in the system, you can use the `top` command in Linux. The `top` command provides a real-time view of the system processes, sorted by their CPU usage by default.

To run the `top` command, open a terminal and simply type:

```css
top
```

The `top` command will show a list of processes, with the highest priority process listed at the top of the list. The priority of a process is determined by the Linux kernel's scheduler, which assigns a priority value to each process based on its resource requirements and other factors.

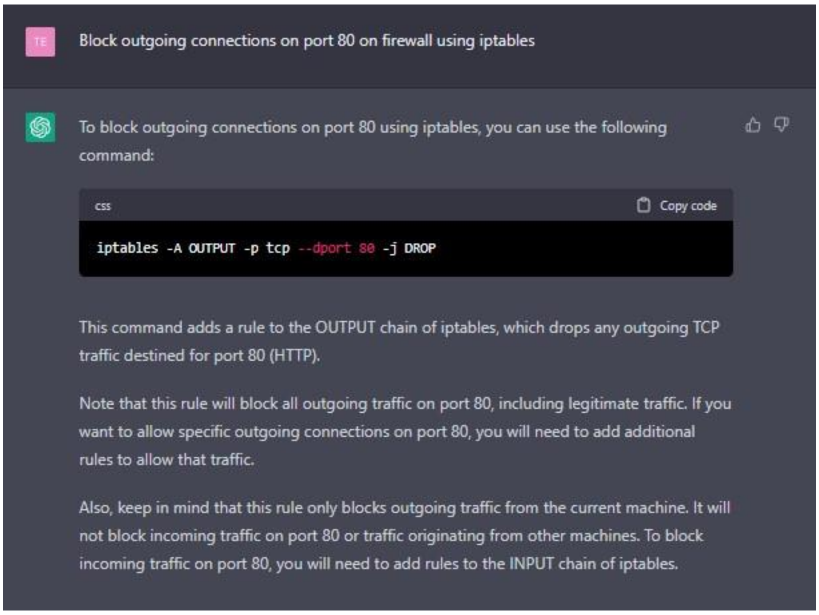
To find out what the purpose of the highest priority process is, you can look at the command name and process ID (PID) listed in the `top` output. Use the PID to look up the process details using the `ps` command. For example:

```bash
ps -p <PID> -o command
```

Replace `<PID>` with the PID of the highest priority process shown in the `top` output. This command will display the command name and arguments for the process with the given PID.

You can then use this information to identify the purpose of the highest priority process and determine if it is a critical system process or an application process that may need to be investigated further.

12. Perform the below tasks on the firewall using iptables:
a. Block outgoing connections on port 80



Block outgoing connections on port 80 on firewall using iptables

To block outgoing connections on port 80 using iptables, you can use the following command:

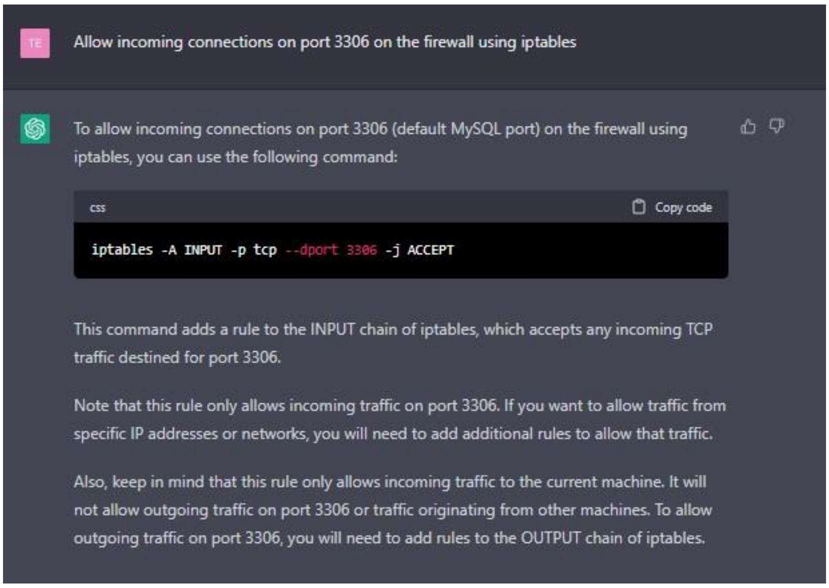
```css
iptables -A OUTPUT -p tcp --dport 80 -j DROP
```

This command adds a rule to the OUTPUT chain of iptables, which drops any outgoing TCP traffic destined for port 80 (HTTP).

Note that this rule will block all outgoing traffic on port 80, including legitimate traffic. If you want to allow specific outgoing connections on port 80, you will need to add additional rules to allow that traffic.

Also, keep in mind that this rule only blocks outgoing traffic from the current machine. It will not block incoming traffic on port 80 or traffic originating from other machines. To block incoming traffic on port 80, you will need to add rules to the INPUT chain of iptables.

b. Allow incoming connections on port 3306



Allow incoming connections on port 3306 on the firewall using iptables

To allow incoming connections on port 3306 (default MySQL port) on the firewall using iptables, you can use the following command:

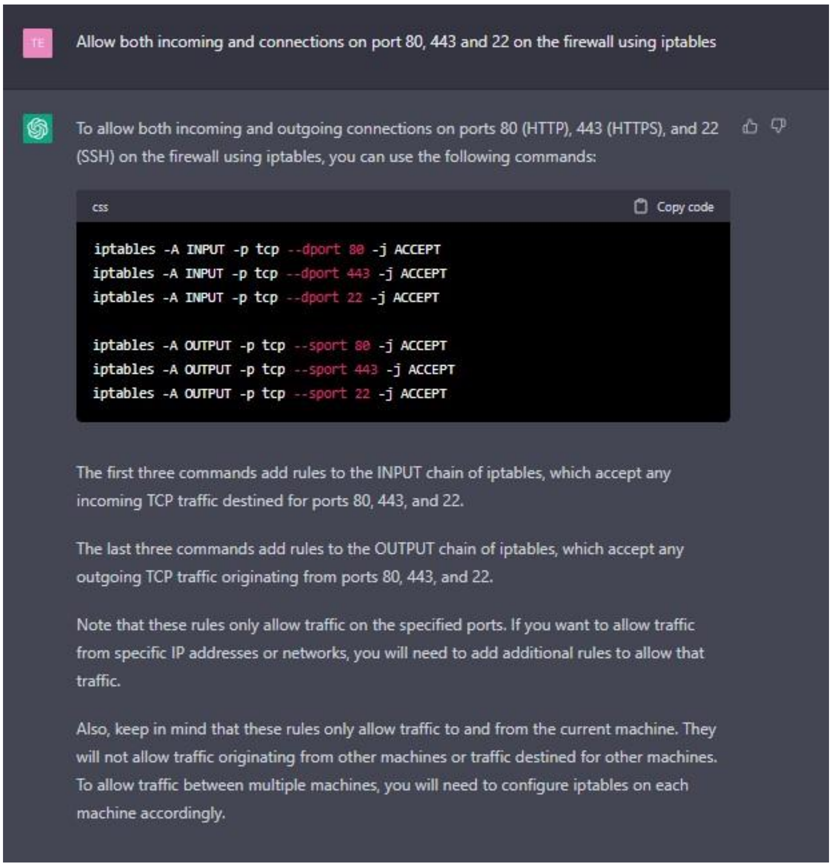
```
iptables -A INPUT -p tcp --dport 3306 -j ACCEPT
```

This command adds a rule to the INPUT chain of iptables, which accepts any incoming TCP traffic destined for port 3306.

Note that this rule only allows incoming traffic on port 3306. If you want to allow traffic from specific IP addresses or networks, you will need to add additional rules to allow that traffic.

Also, keep in mind that this rule only allows incoming traffic to the current machine. It will not allow outgoing traffic on port 3306 or traffic originating from other machines. To allow outgoing traffic on port 3306, you will need to add rules to the OUTPUT chain of iptables.

c. Allow both incoming and connections on port 80, 443 and 22



Allow both incoming and connections on port 80, 443 and 22 on the firewall using iptables

To allow both incoming and outgoing connections on ports 80 (HTTP), 443 (HTTPS), and 22 (SSH) on the firewall using iptables, you can use the following commands:

```
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -A INPUT -p tcp --dport 443 -j ACCEPT
iptables -A INPUT -p tcp --dport 22 -j ACCEPT

iptables -A OUTPUT -p tcp --sport 80 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 443 -j ACCEPT
iptables -A OUTPUT -p tcp --sport 22 -j ACCEPT
```

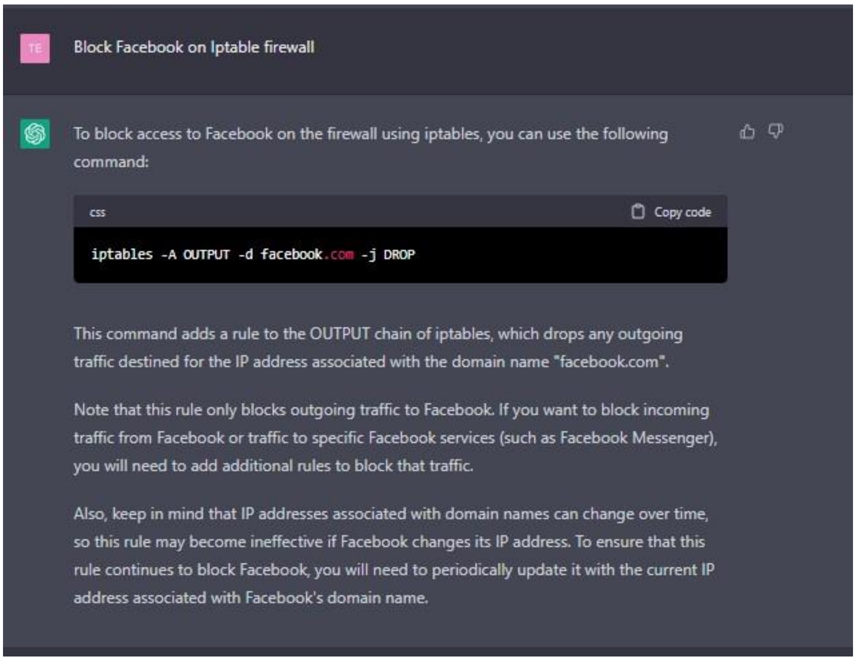
The first three commands add rules to the INPUT chain of iptables, which accept any incoming TCP traffic destined for ports 80, 443, and 22.

The last three commands add rules to the OUTPUT chain of iptables, which accept any outgoing TCP traffic originating from ports 80, 443, and 22.

Note that these rules only allow traffic on the specified ports. If you want to allow traffic from specific IP addresses or networks, you will need to add additional rules to allow that traffic.

Also, keep in mind that these rules only allow traffic to and from the current machine. They will not allow traffic originating from other machines or traffic destined for other machines. To allow traffic between multiple machines, you will need to configure iptables on each machine accordingly.

d. Block Facebook on Iptable firewall



Block Facebook on Iptable firewall

To block access to Facebook on the firewall using iptables, you can use the following command:

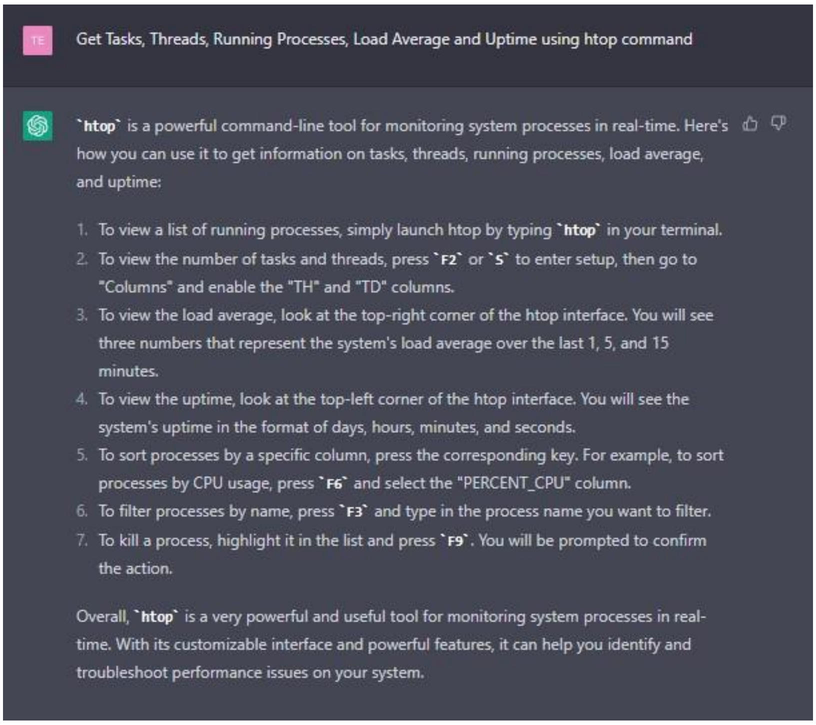
```css
iptables -A OUTPUT -d facebook.com -j DROP
```

This command adds a rule to the OUTPUT chain of iptables, which drops any outgoing traffic destined for the IP address associated with the domain name "facebook.com".

Note that this rule only blocks outgoing traffic to Facebook. If you want to block incoming traffic from Facebook or traffic to specific Facebook services (such as Facebook Messenger), you will need to add additional rules to block that traffic.

Also, keep in mind that IP addresses associated with domain names can change over time, so this rule may become ineffective if Facebook changes its IP address. To ensure that this rule continues to block Facebook, you will need to periodically update it with the current IP address associated with Facebook's domain name.

13. Get Tasks, Threads, Running Processes, Load Average and Uptime using htop command.



Get Tasks, Threads, Running Processes, Load Average and Uptime using htop command

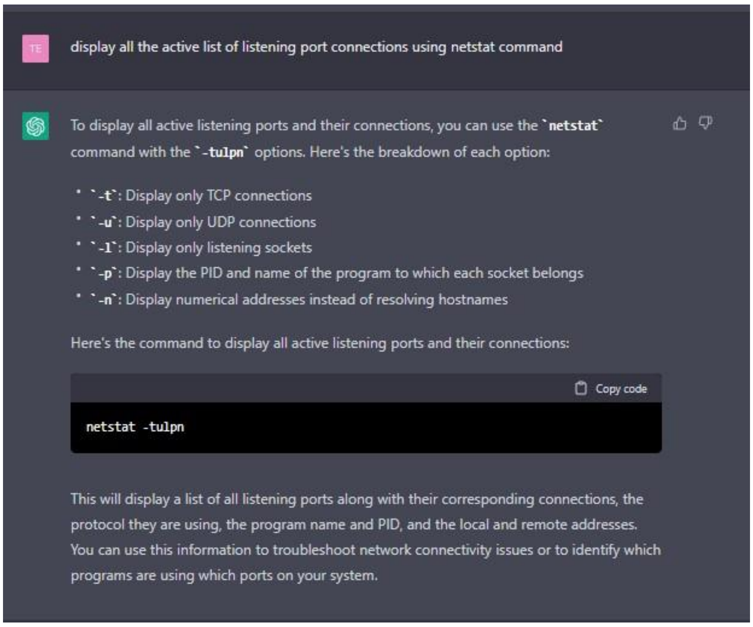
`htop` is a powerful command-line tool for monitoring system processes in real-time. Here's how you can use it to get information on tasks, threads, running processes, load average, and uptime:

1. To view a list of running processes, simply launch htop by typing `htop` in your terminal.
2. To view the number of tasks and threads, press `F2` or `S` to enter setup, then go to "Columns" and enable the "TH" and "TD" columns.
3. To view the load average, look at the top-right corner of the htop interface. You will see three numbers that represent the system's load average over the last 1, 5, and 15 minutes.
4. To view the uptime, look at the top-left corner of the htop interface. You will see the system's uptime in the format of days, hours, minutes, and seconds.
5. To sort processes by a specific column, press the corresponding key. For example, to sort processes by CPU usage, press `F6` and select the "PERCENT_CPU" column.
6. To filter processes by name, press `F3` and type in the process name you want to filter.
7. To kill a process, highlight it in the list and press `F9`. You will be prompted to confirm the action.

Overall, `htop` is a very powerful and useful tool for monitoring system processes in real-time. With its customizable interface and powerful features, it can help you identify and troubleshoot performance issues on your system.

14. Using netstat command, perform the below operations

a. To display all the active list of listening port connections.



display all the active list of listening port connections using netstat command

To display all active listening ports and their connections, you can use the `netstat` command with the `-tulpn` options. Here's the breakdown of each option:

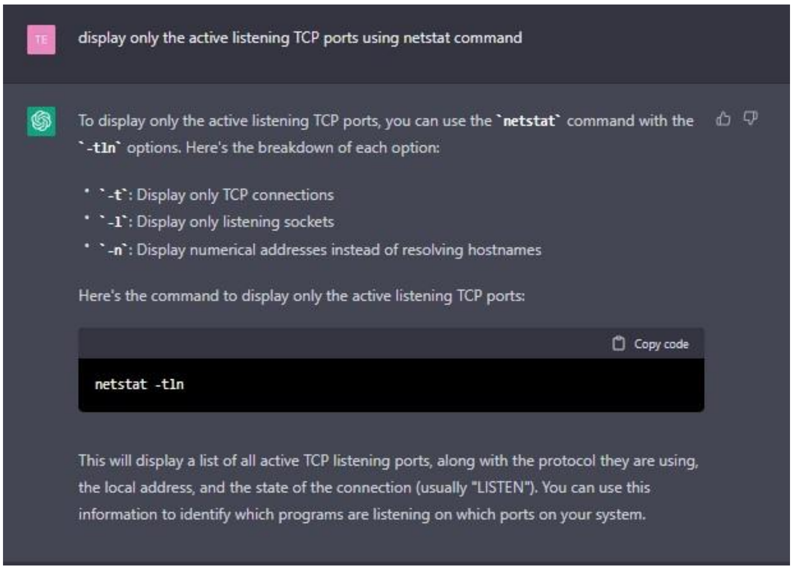
- `-t`: Display only TCP connections
- `-u`: Display only UDP connections
- `-l`: Display only listening sockets
- `-p`: Display the PID and name of the program to which each socket belongs
- `-n`: Display numerical addresses instead of resolving hostnames

Here's the command to display all active listening ports and their connections:

```
netstat -tulpn
```

This will display a list of all listening ports along with their corresponding connections, the protocol they are using, the program name and PID, and the local and remote addresses. You can use this information to troubleshoot network connectivity issues or to identify which programs are using which ports on your system.

b. To display only the active listening TCP ports.



display only the active listening TCP ports using netstat command

To display only the active listening TCP ports, you can use the `netstat` command with the `-tln` options. Here's the breakdown of each option:

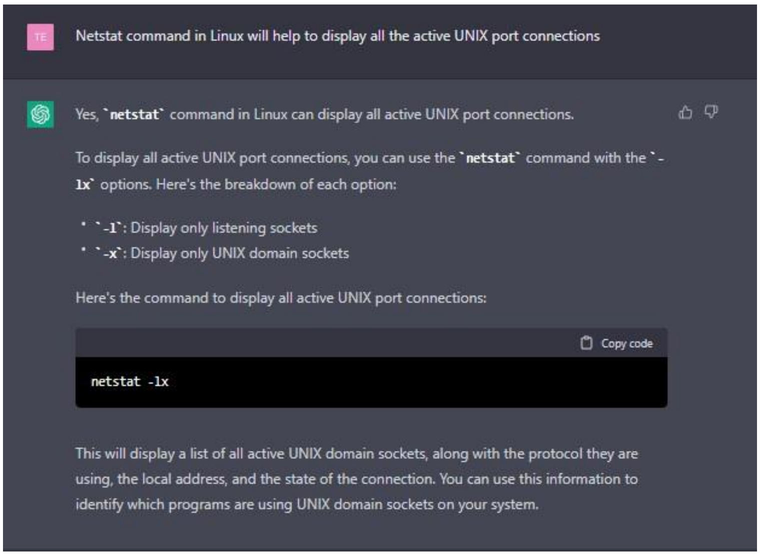
- `-t`: Display only TCP connections
- `-l`: Display only listening sockets
- `-n`: Display numerical addresses instead of resolving hostnames

Here's the command to display only the active listening TCP ports:

```
netstat -tln
```

This will display a list of all active TCP listening ports, along with the protocol they are using, the local address, and the state of the connection (usually "LISTEN"). You can use this information to identify which programs are listening on which ports on your system.

c. Netstat command in Linux will help to display all the active UNIX port connections.



Netstat command in Linux will help to display all the active UNIX port connections

Yes, `netstat` command in Linux can display all active UNIX port connections.

To display all active UNIX port connections, you can use the `netstat` command with the `-lx` options. Here's the breakdown of each option:

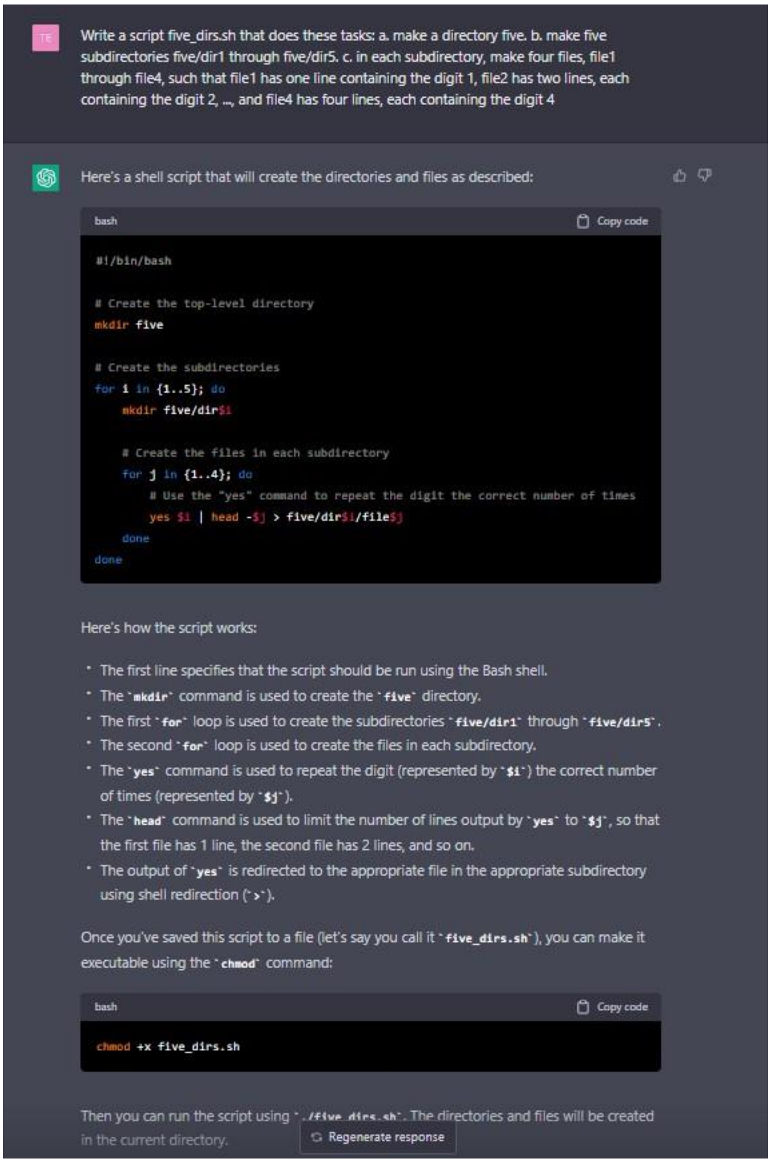
- `-l`: Display only listening sockets
- `-x`: Display only UNIX domain sockets

Here's the command to display all active UNIX port connections:

```
netstat -lx
```

This will display a list of all active UNIX domain sockets, along with the protocol they are using, the local address, and the state of the connection. You can use this information to identify which programs are using UNIX domain sockets on your system.

15. Write a script five_dirs.sh that does these tasks:
    a. make a directory five.
    b. make five subdirectories five/dir1 through five/dir5.
    c. in each subdirectory, make four files, file1 through file4, such that file1 has one line containing the digit 1, file2 has two lines, each containing the digit 2, …, and file4 has four lines, each containing the digit 4



Write a script five_dirs.sh that does these tasks: a. make a directory five. b. make five subdirectories five/dir1 through five/dir5. c. in each subdirectory, make four files, file1 through file4, such that file1 has one line containing the digit 1, file2 has two lines, each containing the digit 2, ..., and file4 has four lines, each containing the digit 4

Here's a shell script that will create the directories and files as described:

```bash
#!/bin/bash

# Create the top-level directory
mkdir five

# Create the subdirectories
for i in {1..5}; do
    mkdir five/dir$i

    # Create the files in each subdirectory
    for j in {1..4}; do
        # Use the "yes" command to repeat the digit the correct number of times
        yes $i | head -$j > five/dir$i/file$j
    done
done
```

Here's how the script works:

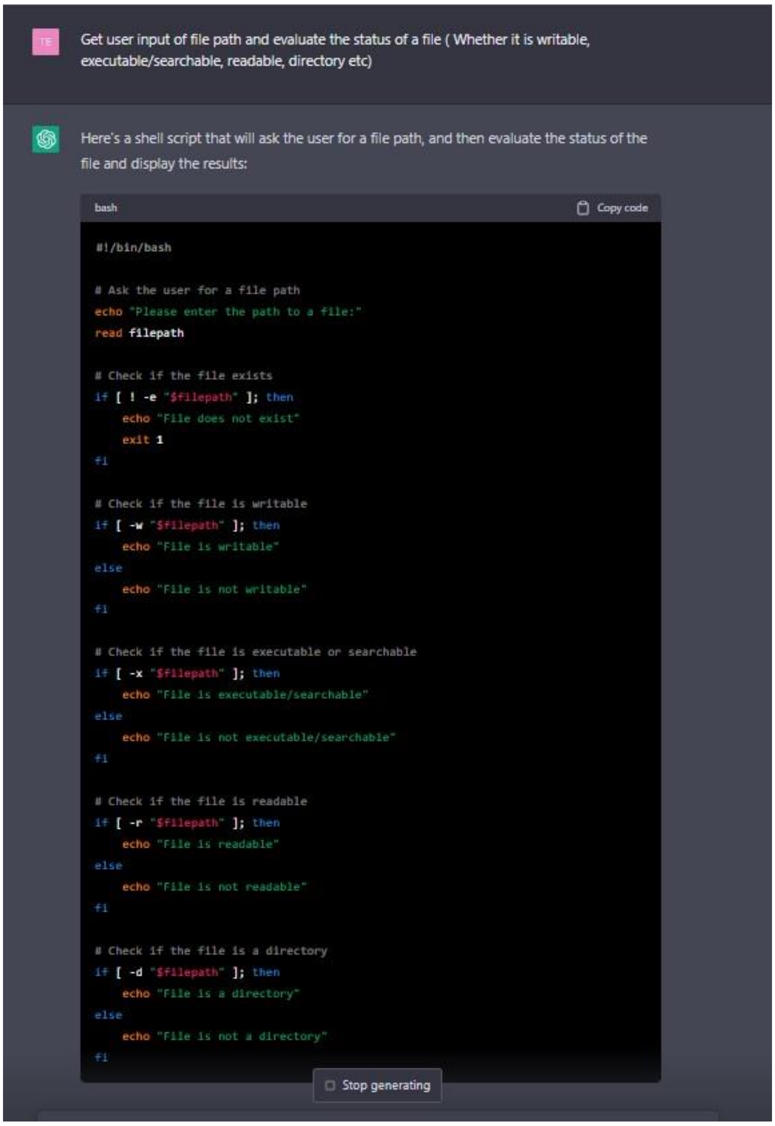
- The first line specifies that the script should be run using the Bash shell.
- The `mkdir` command is used to create the `five` directory.
- The first `for` loop is used to create the subdirectories `five/dir1` through `five/dir5`.
- The second `for` loop is used to create the files in each subdirectory.
- The `yes` command is used to repeat the digit (represented by `$i`) the correct number of times (represented by `$j`).
- The `head` command is used to limit the number of lines output by `yes` to `$j`, so that the first file has 1 line, the second file has 2 lines, and so on.
- The output of `yes` is redirected to the appropriate file in the appropriate subdirectory using shell redirection (`>`).

Once you've saved this script to a file (let's say you call it `five_dirs.sh`), you can make it executable using the `chmod` command:

```bash
chmod +x five_dirs.sh
```

Then you can run the script using `./five_dirs.sh`. The directories and files will be created in the current directory.

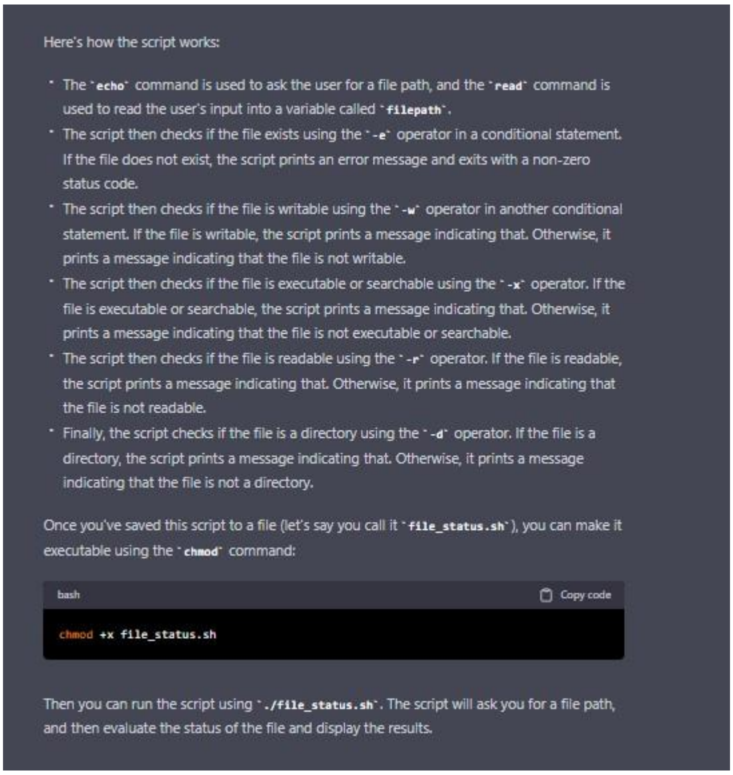
16. Get user input of file path and evaluate the status of a file (Whether it is writable, executable/searchable, readable, directory etc)



Get user input of file path and evaluate the status of a file (Whether it is writable, executable/searchable, readable, directory etc)

Here's a shell script that will ask the user for a file path, and then evaluate the status of the file and display the results:

```bash
#!/bin/bash

# Ask the user for a file path
echo "Please enter the path to a file:"
read filepath

# Check if the file exists
if [ ! -e "$filepath" ]; then
    echo "File does not exist"
    exit 1
fi

# Check if the file is writable
if [ -w "$filepath" ]; then
    echo "File is writable"
else
    echo "File is not writable"
fi

# Check if the file is executable or searchable
if [ -x "$filepath" ]; then
    echo "File is executable/searchable"
else
    echo "File is not executable/searchable"
fi

# Check if the file is readable
if [ -r "$filepath" ]; then
    echo "File is readable"
else
    echo "File is not readable"
fi

# Check if the file is a directory
if [ -d "$filepath" ]; then
    echo "File is a directory"
else
    echo "File is not a directory"
fi
```



Here's how the script works:

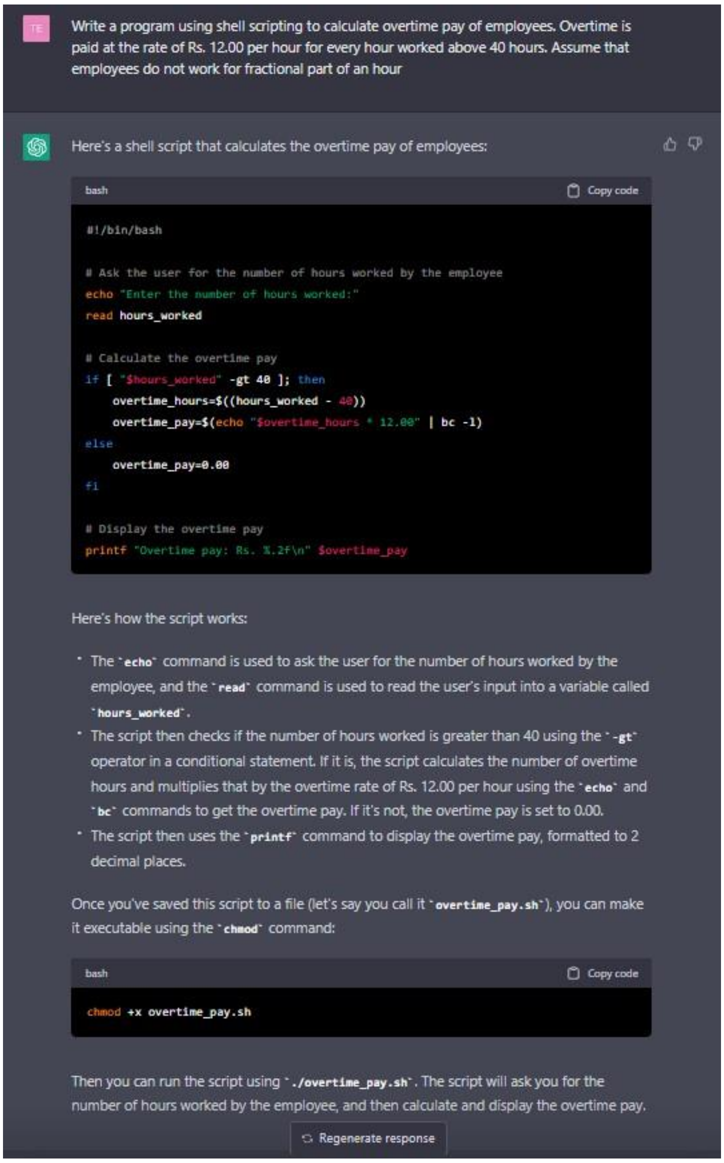
- The `echo` command is used to ask the user for a file path, and the `read` command is used to read the user's input into a variable called `filepath`.
- The script then checks if the file exists using the `-e` operator in a conditional statement. If the file does not exist, the script prints an error message and exits with a non-zero status code.
- The script then checks if the file is writable using the `-w` operator in another conditional statement. If the file is writable, the script prints a message indicating that. Otherwise, it prints a message indicating that the file is not writable.
- The script then checks if the file is executable or searchable using the `-x` operator. If the file is executable or searchable, the script prints a message indicating that. Otherwise, it prints a message indicating that the file is not executable or searchable.
- The script then checks if the file is readable using the `-r` operator. If the file is readable, the script prints a message indicating that. Otherwise, it prints a message indicating that the file is not readable.
- Finally, the script checks if the file is a directory using the `-d` operator. If the file is a directory, the script prints a message indicating that. Otherwise, it prints a message indicating that the file is not a directory.

Once you've saved this script to a file (let's say you call it `file_status.sh`), you can make it executable using the `chmod` command:

```bash
chmod +x file_status.sh
```

Then you can run the script using `./file_status.sh`. The script will ask you for a file path, and then evaluate the status of the file and display the results.

17. Write a program to calculate overtime pay of employees. Overtime is paid at the rate of Rs. 12.00 per hour for every hour worked above 40 hours. Assume that employees do not work for fractional part of an hour.



TE

Write a program using shell scripting to calculate overtime pay of employees. Overtime is paid at the rate of Rs. 12.00 per hour for every hour worked above 40 hours. Assume that employees do not work for fractional part of an hour

Here's a shell script that calculates the overtime pay of employees:

```bash
#!/bin/bash

# Ask the user for the number of hours worked by the employee
echo "Enter the number of hours worked:"
read hours_worked

# Calculate the overtime pay
if [ "$hours_worked" -gt 40 ]; then
    overtime_hours=$((hours_worked - 40))
    overtime_pay=$(echo "$overtime_hours * 12.00" | bc -l)
else
    overtime_pay=0.00
fi

# Display the overtime pay
printf "Overtime pay: Rs. %.2f\n" $overtime_pay
```

Here's how the script works:

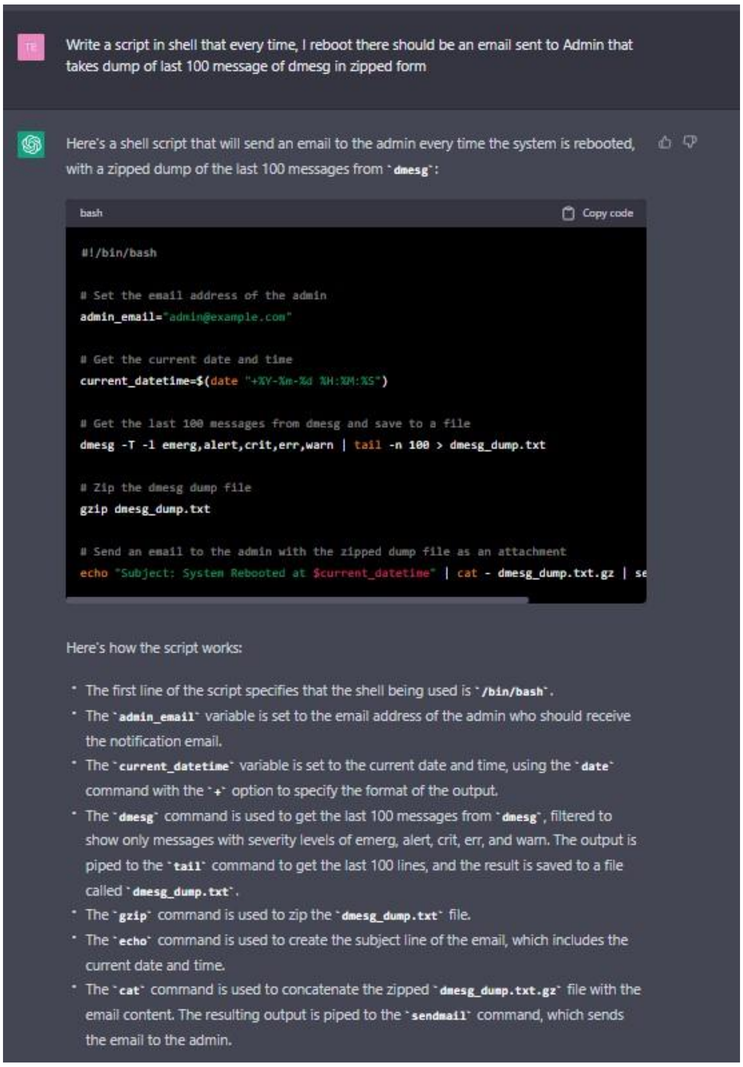
- The `echo` command is used to ask the user for the number of hours worked by the employee, and the `read` command is used to read the user's input into a variable called `hours_worked`.
- The script then checks if the number of hours worked is greater than 40 using the `-gt` operator in a conditional statement. If it is, the script calculates the number of overtime hours and multiplies that by the overtime rate of Rs. 12.00 per hour using the `echo` and `bc` commands to get the overtime pay. If it's not, the overtime pay is set to 0.00.
- The script then uses the `printf` command to display the overtime pay, formatted to 2 decimal places.

Once you've saved this script to a file (let's say you call it `overtime_pay.sh`), you can make it executable using the `chmod` command:

```bash
chmod +x overtime_pay.sh
```

Then you can run the script using `./overtime_pay.sh`. The script will ask you for the number of hours worked by the employee, and then calculate and display the overtime pay.

18. Write a script that every time, I reboot there should be an email sent to Admin that takes dump of last 100 message of dmesg in zipped form.



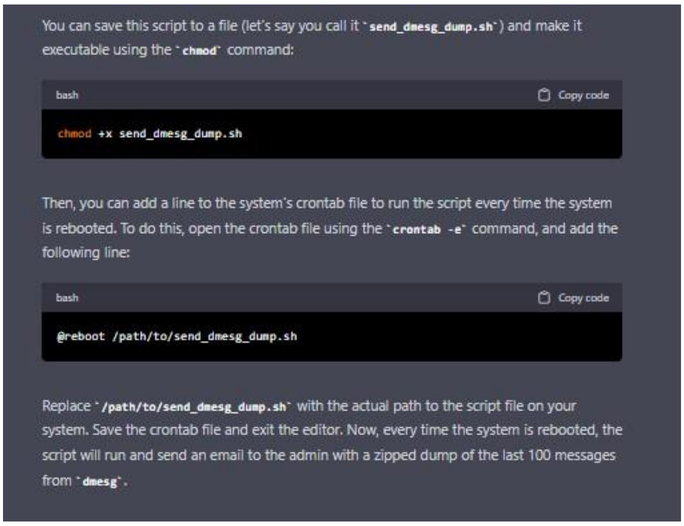
Write a script in shell that every time, I reboot there should be an email sent to Admin that takes dump of last 100 message of dmesg in zipped form

Here's a shell script that will send an email to the admin every time the system is rebooted, with a zipped dump of the last 100 messages from `dmesg`:

```bash
#!/bin/bash

# Set the email address of the admin
admin_email="admin@example.com"

# Get the current date and time
current_datetime=$(date "+%Y-%m-%d %H:%M:%S")

# Get the last 100 messages from dmesg and save to a file
dmesg -T -l emerg,alert,crit,err,warn | tail -n 100 > dmesg_dump.txt

# Zip the dmesg dump file
gzip dmesg_dump.txt

# Send an email to the admin with the zipped dump file as an attachment
echo "Subject: System Rebooted at $current_datetime" | cat - dmesg_dump.txt.gz | se
```

Here's how the script works:

- The first line of the script specifies that the shell being used is `/bin/bash`.
- The `admin_email` variable is set to the email address of the admin who should receive the notification email.
- The `current_datetime` variable is set to the current date and time, using the `date` command with the `+` option to specify the format of the output.
- The `dmesg` command is used to get the last 100 messages from `dmesg`, filtered to show only messages with severity levels of emerg, alert, crit, err, and warn. The output is piped to the `tail` command to get the last 100 lines, and the result is saved to a file called `dmesg_dump.txt`.
- The `gzip` command is used to zip the `dmesg_dump.txt` file.
- The `echo` command is used to create the subject line of the email, which includes the current date and time.
- The `cat` command is used to concatenate the zipped `dmesg_dump.txt.gz` file with the email content. The resulting output is piped to the `sendmail` command, which sends the email to the admin.



You can save this script to a file (let's say you call it `send_dmesg_dump.sh`) and make it executable using the `chmod` command:

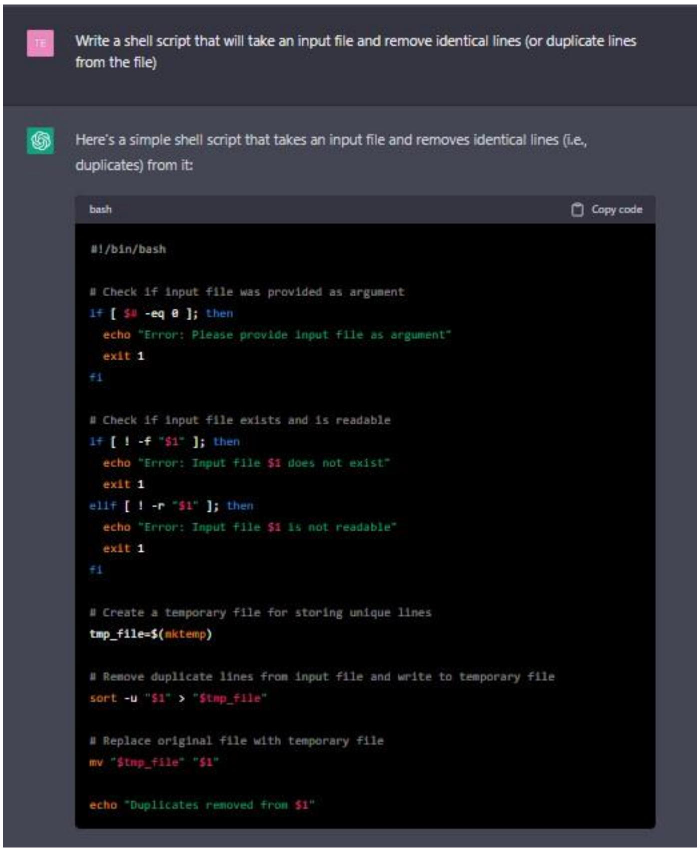
```bash
chmod +x send_dmesg_dump.sh
```

Then, you can add a line to the system's crontab file to run the script every time the system is rebooted. To do this, open the crontab file using the `crontab -e` command, and add the following line:

```bash
@reboot /path/to/send_dmesg_dump.sh
```

Replace `/path/to/send_dmesg_dump.sh` with the actual path to the script file on your system. Save the crontab file and exit the editor. Now, every time the system is rebooted, the script will run and send an email to the admin with a zipped dump of the last 100 messages from `dmesg`.

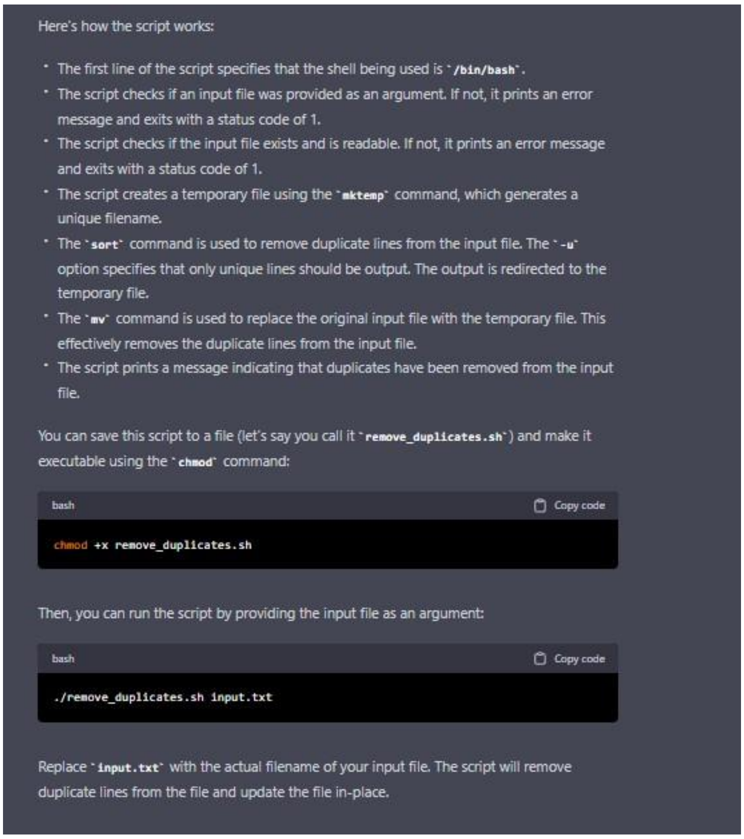
19. Write a shell script that will take an input file and remove identical lines (or duplicate lines from the file).



Write a shell script that will take an input file and remove identical lines (or duplicate lines from the file)

Here's a simple shell script that takes an input file and removes identical lines (i.e., duplicates) from it:

```bash
#!/bin/bash

# Check if input file was provided as argument
if [ $# -eq 0 ]; then
  echo "Error: Please provide input file as argument"
  exit 1
fi

# Check if input file exists and is readable
if [ ! -f "$1" ]; then
  echo "Error: Input file $1 does not exist"
  exit 1
elif [ ! -r "$1" ]; then
  echo "Error: Input file $1 is not readable"
  exit 1
fi

# Create a temporary file for storing unique lines
tmp_file=$(mktemp)

# Remove duplicate lines from input file and write to temporary file
sort -u "$1" > "$tmp_file"

# Replace original file with temporary file
mv "$tmp_file" "$1"

echo "Duplicates removed from $1"
```



Here's how the script works:

- The first line of the script specifies that the shell being used is `/bin/bash`.
- The script checks if an input file was provided as an argument. If not, it prints an error message and exits with a status code of 1.
- The script checks if the input file exists and is readable. If not, it prints an error message and exits with a status code of 1.
- The script creates a temporary file using the `mktemp` command, which generates a unique filename.
- The `sort` command is used to remove duplicate lines from the input file. The `-u` option specifies that only unique lines should be output. The output is redirected to the temporary file.
- The `mv` command is used to replace the original input file with the temporary file. This effectively removes the duplicate lines from the input file.
- The script prints a message indicating that duplicates have been removed from the input file.

You can save this script to a file (let's say you call it `remove_duplicates.sh`) and make it executable using the `chmod` command:

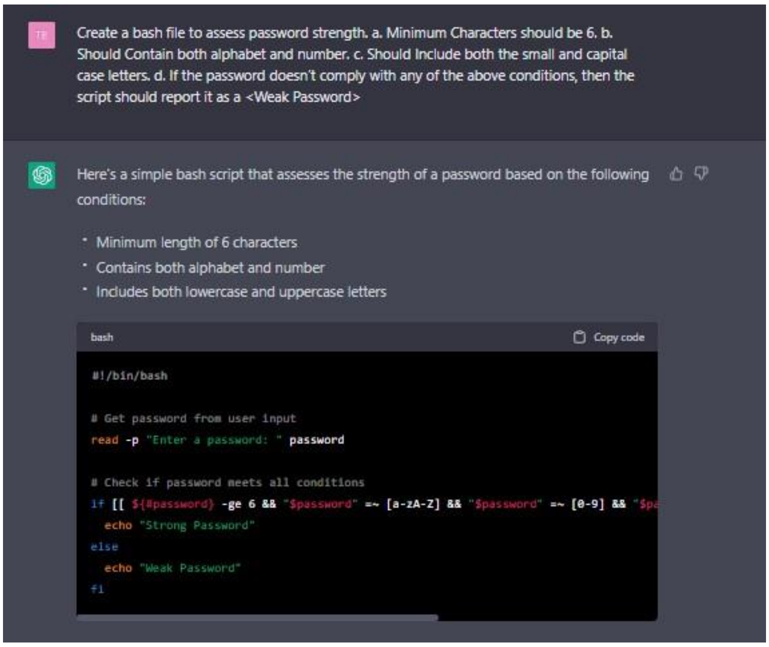
```bash
chmod +x remove_duplicates.sh
```

Then, you can run the script by providing the input file as an argument:

```bash
./remove_duplicates.sh input.txt
```

Replace `input.txt` with the actual filename of your input file. The script will remove duplicate lines from the file and update the file in-place.

20. Create a bash file to assess password strength.
    a. Minimum Characters should be 6.
    b. Should Contain both alphabet and number.
    c. Should Include both the small and capital case letters.
    d. If the password doesn't comply with any of the above conditions, then the script should report it as a <Weak Password>

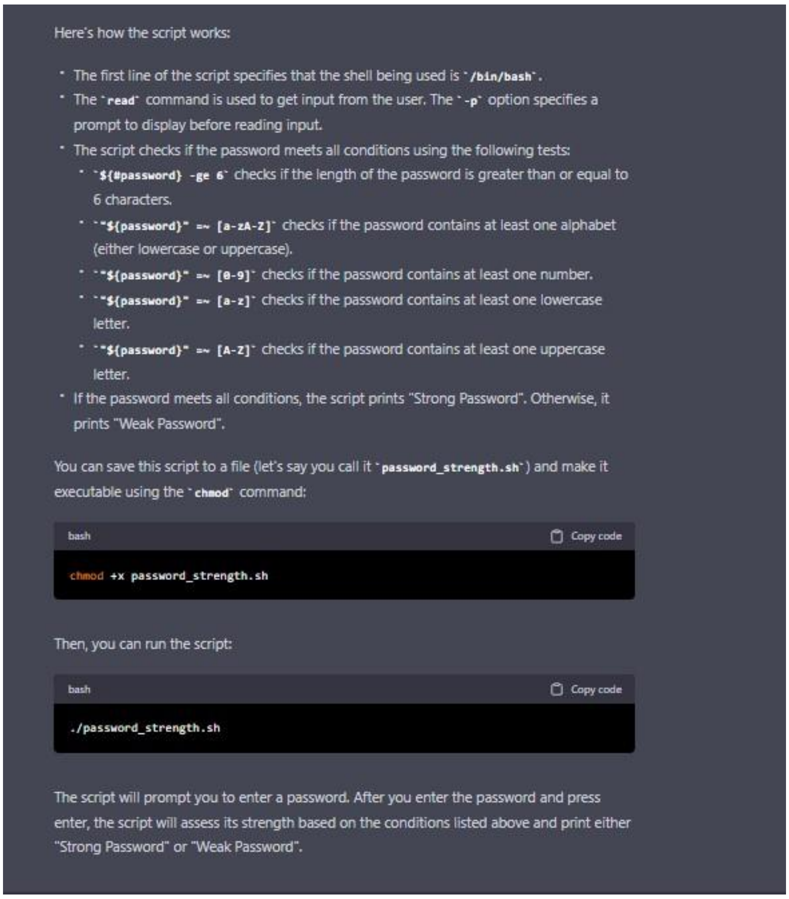


Create a bash file to assess password strength. a. Minimum Characters should be 6. b. Should Contain both alphabet and number. c. Should Include both the small and capital case letters. d. If the password doesn't comply with any of the above conditions, then the script should report it as a <Weak Password>

Here's a simple bash script that assesses the strength of a password based on the following conditions:

- Minimum length of 6 characters
- Contains both alphabet and number
- Includes both lowercase and uppercase letters

```bash
#!/bin/bash

# Get password from user input
read -p "Enter a password: " password

# Check if password meets all conditions
if [[ ${#password} -ge 6 && "$password" =~ [a-zA-Z] && "$password" =~ [0-9] && "$pa
  echo "Strong Password"
else
  echo "Weak Password"
fi
```



Here's how the script works:

- The first line of the script specifies that the shell being used is `/bin/bash`.
- The `read` command is used to get input from the user. The `-p` option specifies a prompt to display before reading input.
- The script checks if the password meets all conditions using the following tests:
  - `${#password} -ge 6` checks if the length of the password is greater than or equal to 6 characters.
  - `"${password}" =~ [a-zA-Z]` checks if the password contains at least one alphabet (either lowercase or uppercase).
  - `"${password}" =~ [0-9]` checks if the password contains at least one number.
  - `"${password}" =~ [a-z]` checks if the password contains at least one lowercase letter.
  - `"${password}" =~ [A-Z]` checks if the password contains at least one uppercase letter.
- If the password meets all conditions, the script prints "Strong Password". Otherwise, it prints "Weak Password".

You can save this script to a file (let's say you call it `password_strength.sh`) and make it executable using the `chmod` command:

```bash
chmod +x password_strength.sh
```

Then, you can run the script:

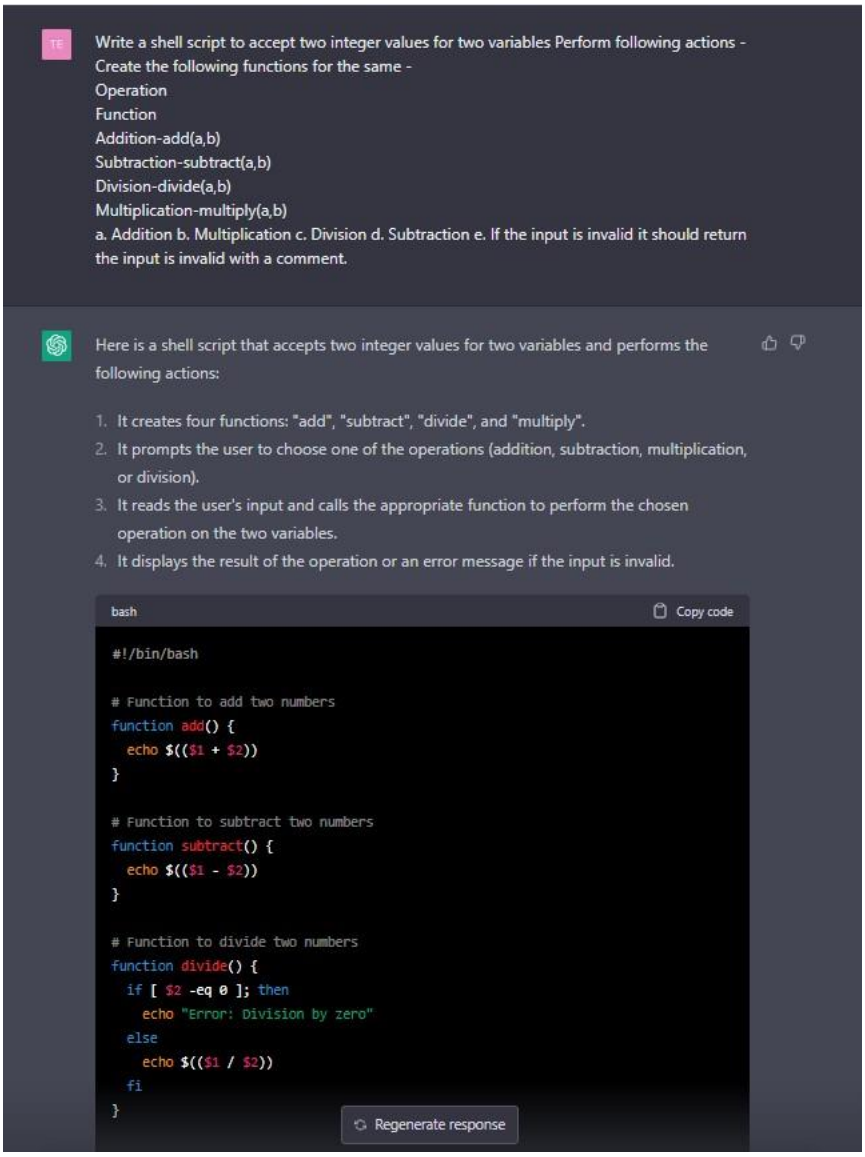
```bash
./password_strength.sh
```

The script will prompt you to enter a password. After you enter the password and press enter, the script will assess its strength based on the conditions listed above and print either "Strong Password" or "Weak Password".

21. Write a shell script to accept two integer values for two variables Perform following actions -
    Create the following functions for the same -

Operation	Function
Addition	add(a,b)
Subtraction	subtract(a,b)
Division	divide(a,b)
Multiplication	multiply(a,b)

a. Addition
b. Multiplication
c. Division
d. Subtraction
e. If the input is invalid it should return the input is invalid with a comment.

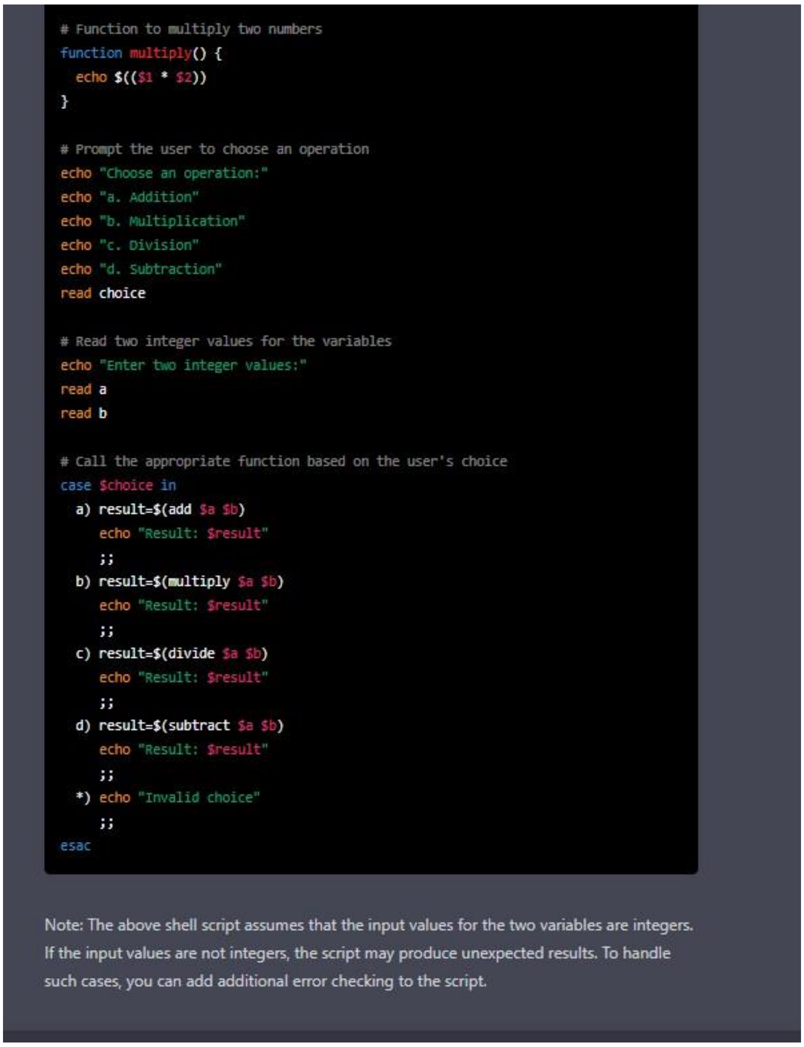


Write a shell script to accept two integer values for two variables Perform following actions -
Create the following functions for the same -
Operation
Function
Addition-add(a,b)
Subtraction-subtract(a,b)
Division-divide(a,b)
Multiplication-multiply(a,b)
a. Addition b. Multiplication c. Division d. Subtraction e. If the input is invalid it should return the input is invalid with a comment.

Here is a shell script that accepts two integer values for two variables and performs the following actions:

1. It creates four functions: "add", "subtract", "divide", and "multiply".
2. It prompts the user to choose one of the operations (addition, subtraction, multiplication, or division).
3. It reads the user's input and calls the appropriate function to perform the chosen operation on the two variables.
4. It displays the result of the operation or an error message if the input is invalid.

```bash
#!/bin/bash

# Function to add two numbers
function add() {
  echo $(($1 + $2))
}

# Function to subtract two numbers
function subtract() {
  echo $(($1 - $2))
}

# Function to divide two numbers
function divide() {
  if [ $2 -eq 0 ]; then
    echo "Error: Division by zero"
  else
    echo $(($1 / $2))
  fi
}
```



```
# Function to multiply two numbers
function multiply() {
  echo $(($1 * $2))
}

# Prompt the user to choose an operation
echo "Choose an operation:"
echo "a. Addition"
echo "b. Multiplication"
echo "c. Division"
echo "d. Subtraction"
read choice

# Read two integer values for the variables
echo "Enter two integer values:"
read a
read b

# Call the appropriate function based on the user's choice
case $choice in
  a) result=$(add $a $b)
     echo "Result: $result"
     ;;
  b) result=$(multiply $a $b)
     echo "Result: $result"
     ;;
  c) result=$(divide $a $b)
     echo "Result: $result"
     ;;
  d) result=$(subtract $a $b)
     echo "Result: $result"
     ;;
  *) echo "Invalid choice"
     ;;
esac
```

Note: The above shell script assumes that the input values for the two variables are integers. If the input values are not integers, the script may produce unexpected results. To handle such cases, you can add additional error checking to the script.

22. Write a shell script that takes a directory as an input and counts the total number of different types of files and directories present in the input directory
    example -

input_dir/

-- dir1/

-- -- file1.txt

-- -- file1.js

-- -- file2.md

-- -- dir2/

-- -- -- -- file2.txt

-- -- -- -- file2.ts

-- file.md

-- file.sh
expected Output -

Output

txt: 2

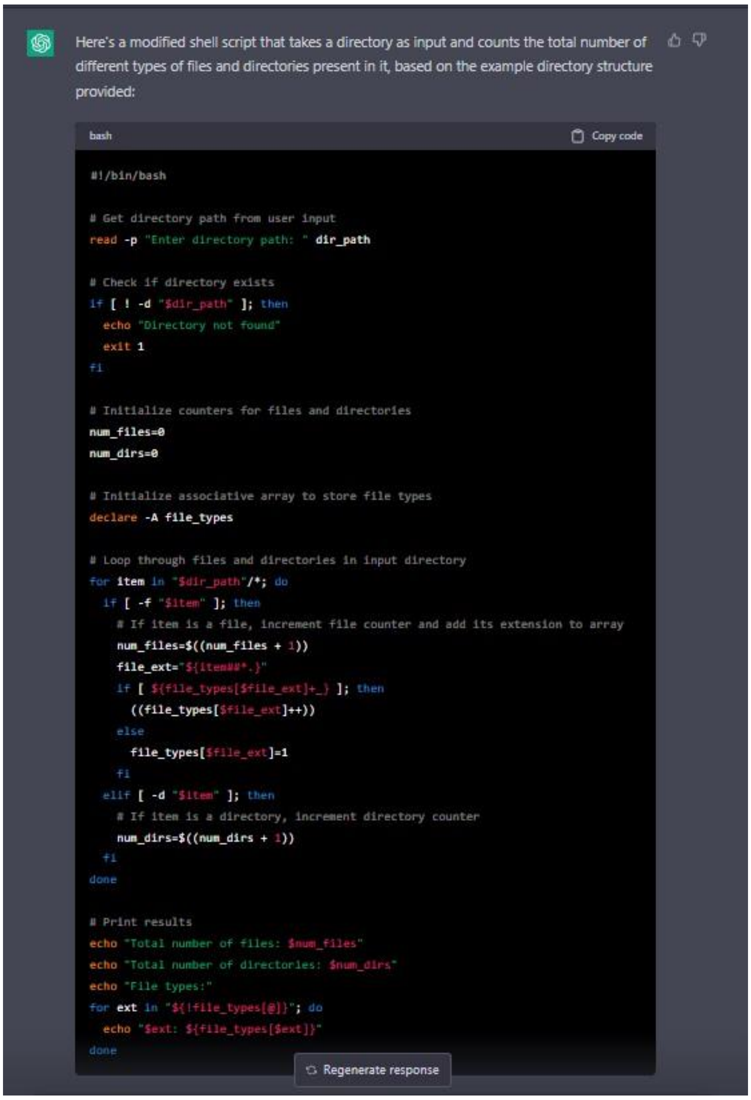
js: 1

md: 2

ts: 1

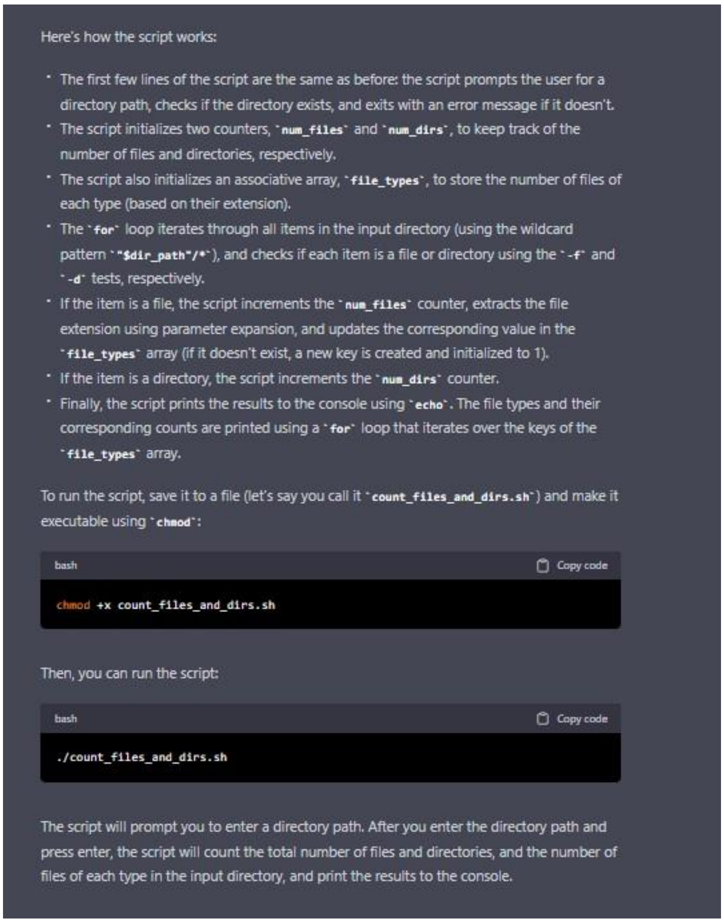
sh: 1

directories: 2



Here's a modified shell script that takes a directory as input and counts the total number of different types of files and directories present in it, based on the example directory structure provided:

```bash
#!/bin/bash

# Get directory path from user input
read -p "Enter directory path: " dir_path

# Check if directory exists
if [ ! -d "$dir_path" ]; then
  echo "Directory not found"
  exit 1
fi

# Initialize counters for files and directories
num_files=0
num_dirs=0

# Initialize associative array to store file types
declare -A file_types

# Loop through files and directories in input directory
for item in "$dir_path"/*; do
  if [ -f "$item" ]; then
    # If item is a file, increment file counter and add its extension to array
    num_files=$((num_files + 1))
    file_ext="${item##*.}"
    if [ ${file_types[$file_ext]+_} ]; then
      ((file_types[$file_ext]++))
    else
      file_types[$file_ext]=1
    fi
  elif [ -d "$item" ]; then
    # If item is a directory, increment directory counter
    num_dirs=$((num_dirs + 1))
  fi
done

# Print results
echo "Total number of files: $num_files"
echo "Total number of directories: $num_dirs"
echo "File types:"
for ext in "${!file_types[@]}"; do
  echo "$ext: ${file_types[$ext]}"
done
```



Here's how the script works:

- The first few lines of the script are the same as before: the script prompts the user for a directory path, checks if the directory exists, and exits with an error message if it doesn't.
- The script initializes two counters, `num_files` and `num_dirs`, to keep track of the number of files and directories, respectively.
- The script also initializes an associative array, `file_types`, to store the number of files of each type (based on their extension).
- The `for` loop iterates through all items in the input directory (using the wildcard pattern `"$dir_path"/*`), and checks if each item is a file or directory using the `-f` and `-d` tests, respectively.
- If the item is a file, the script increments the `num_files` counter, extracts the file extension using parameter expansion, and updates the corresponding value in the `file_types` array (if it doesn't exist, a new key is created and initialized to 1).
- If the item is a directory, the script increments the `num_dirs` counter.
- Finally, the script prints the results to the console using `echo`. The file types and their corresponding counts are printed using a `for` loop that iterates over the keys of the `file_types` array.

To run the script, save it to a file (let's say you call it `count_files_and_dirs.sh`) and make it executable using `chmod`:

```bash
chmod +x count_files_and_dirs.sh
```

Then, you can run the script:

```bash
./count_files_and_dirs.sh
```

The script will prompt you to enter a directory path. After you enter the directory path and press enter, the script will count the total number of files and directories, and the number of files of each type in the input directory, and print the results to the console.

23. In log file which looks like this:
    [status code] IP /endpoint timestamp_utc response_time_s message
    - give avg response times of all /abc calls
    - give all endpoints with more than 5 4xx errors
    - give Ip with most API hits

```
[200] 172.3.4.2 /abc xyz 0.1 OK
[200] 172.3.43.5 /efg xyz 0.1 OK
[200] 172.123.4.6 /qwe xyz 0.1 OK
[200] 172.3.4.24 /abc xyz 0.2 OK
[400] 172.3.44.2 /abc xyz 0.3 OK
[400] 172.3.123.9 /qwc xyz 0.3 OK
[404] 172.33.4.1 /trc xyz 0.3 OK
```

example - <shell script file> <file name> /<endpoint>

./log_analysis.sh <filename> /abc -> 0.2