



UNIVERSITY OF PETROLEUM AND ENERGY STUDIES (UPES)

Linux Practical Report

Submitted by:

Name: Aadarshi Bisht

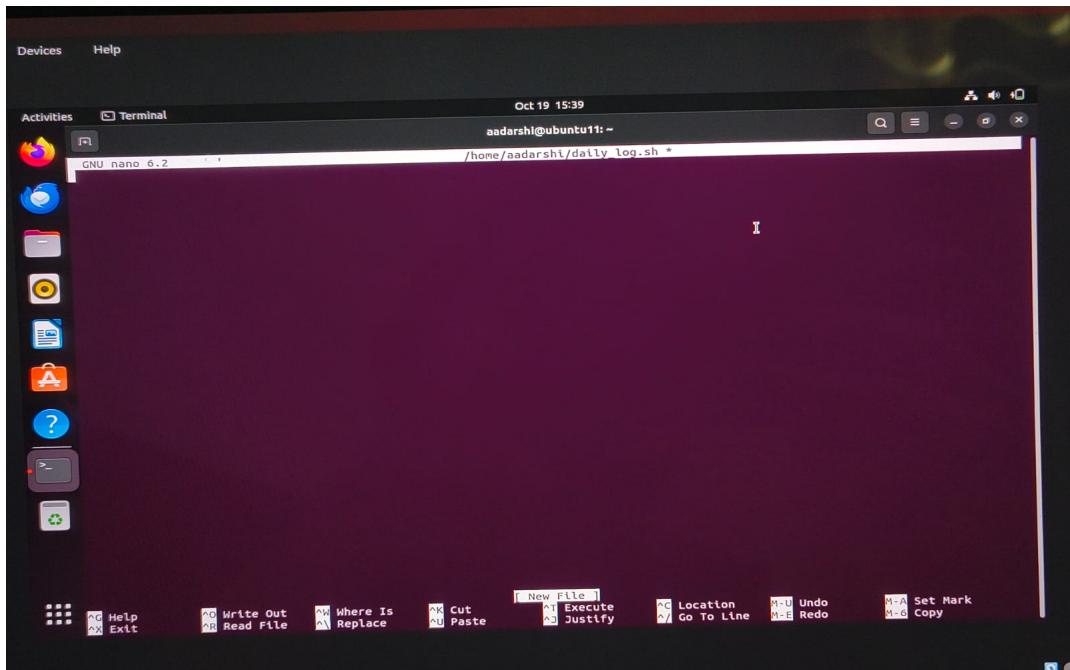
SAP ID: 590029330

Batch: 77-78

Subject: Linux

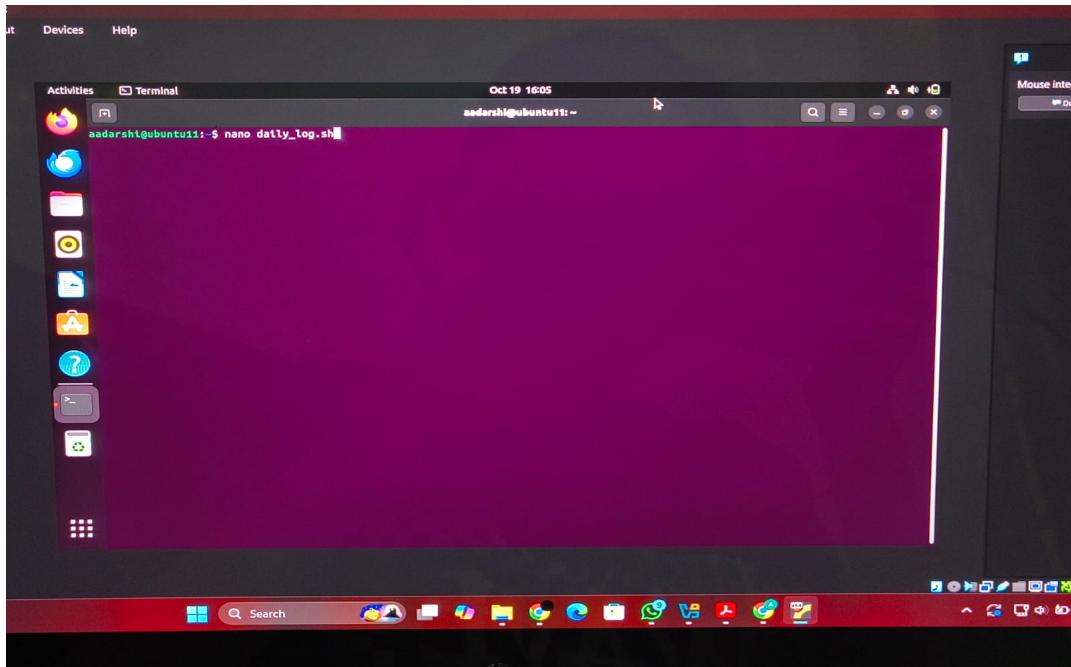
School of Computer Science

Step 1: Create the daily_log.sh script file using nano editor.



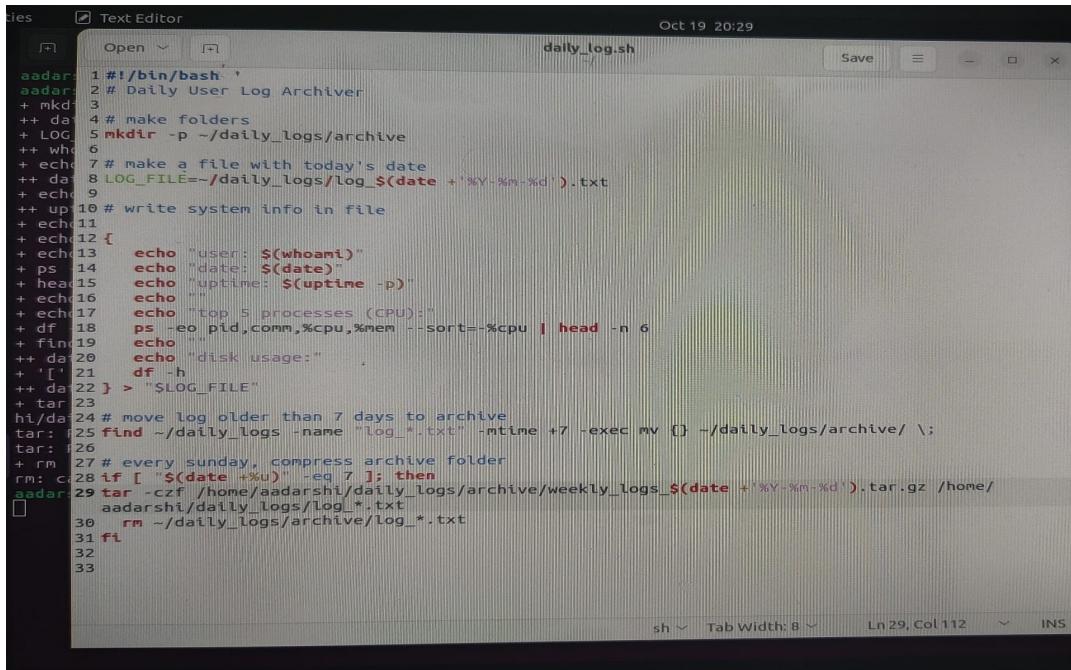
In this step, we create the `daily_log.sh` file using the nano text editor. This script will help automate daily system logging tasks.

Step 2: Write commands to capture system information and save it to a log file.



We add commands to gather information such as uptime, disk usage, and running processes. This helps track system performance.

Step 3: Add file creation date and user details using `whoami` and `date`.



The screenshot shows a terminal window titled "Text Editor" with the file "daily_log.sh" open. The script content is as follows:

```
#!/bin/bash
# Daily User Log Archiver
+ mkdir -p ~/daily_logs/archive
++ date
++ whoami
+ echo
++ date
+ LOG_FILE=~/daily_logs/log_$(date +'%Y-%m-%d').txt
+ echo
++ uptime
+ echo
+ echo "user: $(whoami)"
+ ps
+ head
+ echo
+ echo "top 5 processes (CPU):"
+ df -h
+ df -h
++ date
+ '['
+ df -h
++ date
+ > "$LOG_FILE"
+ tar
+ rm
tar: find ~/daily_logs -name "log_*.txt" -mtime +7 -exec mv {} ~/daily_logs/archive/ \;
tar: rm: every sunday, compress archive folder
rm: if [ "$(date +\%u)" -eq 7 ]; then
tar -czf /home/aadarshi/daily_logs/archive/weekly_logs_$(date +'%Y-%m-%d').tar.gz /home/aadarshi/daily_logs/log_*.txt
rm ~/daily_logs/archive/log_*.txt
fi

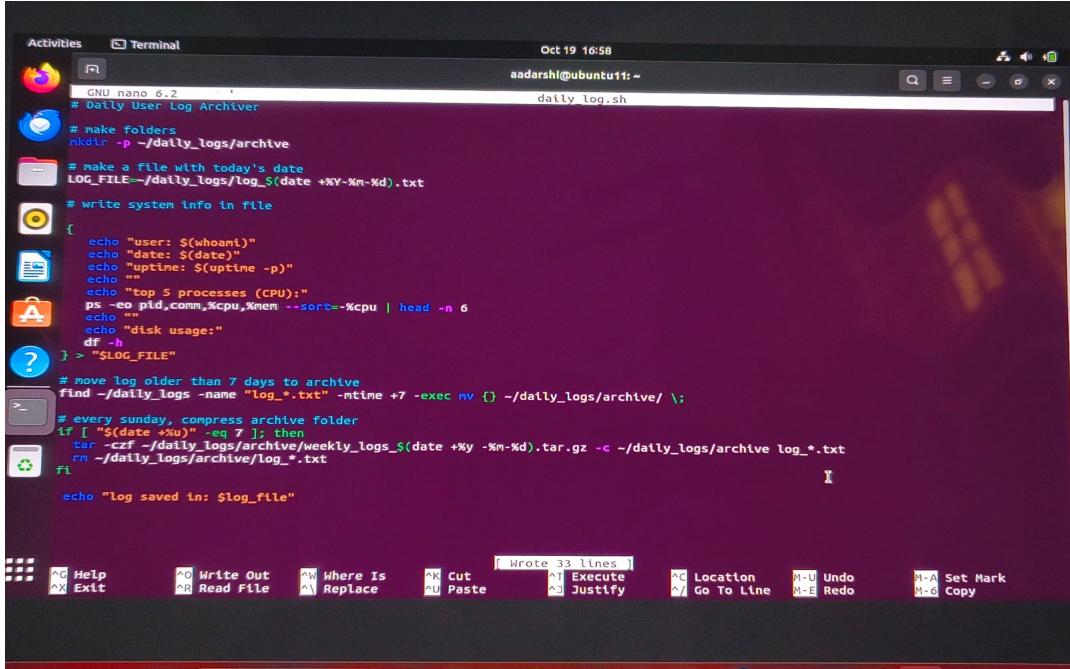
```

The script performs the following tasks:

- Creates a directory structure for logs.
- Creates a log file named after the current date.
- Writes system information to the log file.
- Identifies the current user.
- Shows the top 5 processes using top.
- Shows disk usage.
- Moves log files older than 7 days to an archive directory.
- Compresses the weekly log files into a tar archive on Sundays.

The script adds user details and date automatically to make logs unique and time-stamped.

Step 4: Include top 5 CPU processes and disk usage information.

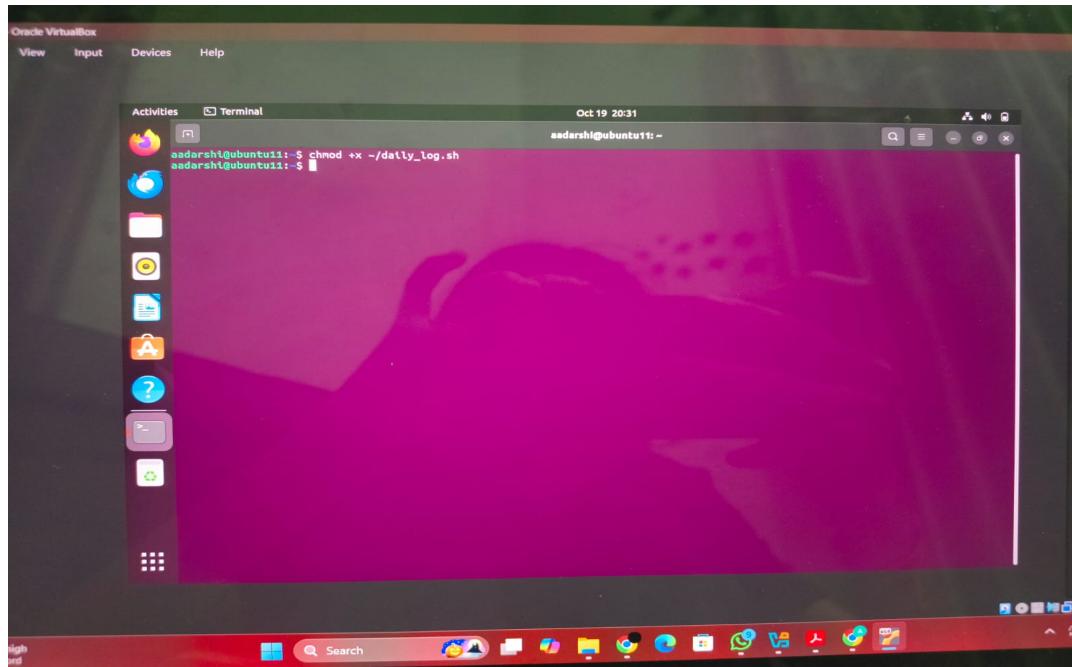


```
GNU nano 6.2          aadarsh@ubuntu11: ~
# Daily User Log Archiver
# make folders
mkdir -p ~/daily_logs/archive
# make a file with today's date
LOG_FILE=~/daily_logs/log_$(date +%Y-%m-%d).txt
# write system info in file
{
echo "user: $(whoami)"
echo "date: $(date)"
echo "uptime: $(uptime -p)"
echo ""
echo "top 5 processes (CPU):"
ps -eo pid,comm,%cpu,%mem --sort=-%cpu | head -n 6
echo ""
echo "disk usage:"
df -h
} > "$LOG_FILE"

# move log older than 7 days to archive
find ~/daily_logs -name "log_*txt" -mtime +7 -exec mv {} ~/daily_logs/archive/ \;
# every sunday, compress archive folder
if [ "$(date +%u)" -eq 7 ]; then
tar -czf ~/daily_logs/archive/weekly_logs_$(date +%Y-%m-%d).tar.gz -c ~/daily_logs/archive log_*.txt
rm ~/daily_logs/archive/log_*.txt
fi
echo "log saved in: $log_file"
```

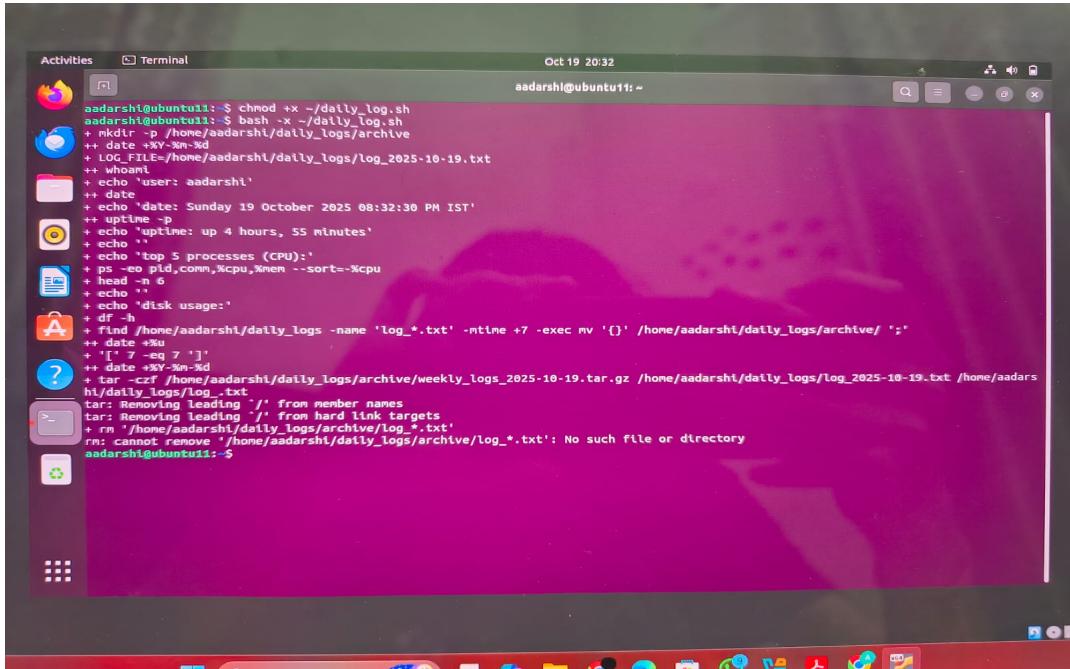
By capturing top CPU processes and disk usage, the log provides valuable system monitoring data.

Step 5: Set file permissions using chmod to make script executable.



We change file permissions using `chmod +x` to make the script executable.

Step 6: Execute the script and check the generated log file output.



The screenshot shows a terminal window on a Ubuntu desktop environment. The terminal title is "Terminal" and the date and time are "Oct 19 20:32". The user is "aadarshi@ubuntu11:~". The terminal displays the execution of a shell script named "daily_log.sh". The script performs several tasks:

- Changes the mode of the script to executable: `chmod +x ~/daily_log.sh`
- Runs the script itself: `bash -x ~/daily_log.sh`
- Makes a directory: `mkdir -p /home/aadarshi/daily_logs/archive`
- Defines a variable: `LOG_FILE=/home/aadarshi/daily_logs/log_2025-10-19.txt`
- Prints the current user: `whoami`
- Prints the current date: `date`
- Prints system uptime: `uptime -p`
- Prints top 5 processes: `top 5 processes (CPU)`
- Prints memory usage: `ps -e pid,comm,%cpu,%mem --sort=-%cpu`
- Prints disk usage: `df -h`
- Finds log files older than 7 days and moves them to an archive: `find /home/aadarshi/daily_logs -name 'log_*.txt' -mtime +7 -exec mv '{}' /home/aadarshi/daily_logs/archive/ ';'`
- Prints the number of days: `date +%u`
- Tarballs the weekly logs: `tar -czf /home/aadarshi/daily_logs/archive/weekly_logs_2025-10-19.tar.gz /home/aadarshi/daily_logs/log_2025-10-19.txt /home/aadarshi/daily_logs/log.log`
- Prints the command used: `tar: Removing leading '/' from member names`
- Prints the command used: `tar: Removing leading '/' from hard link targets`
- Attempts to remove a non-existent file: `rm '/home/aadarshi/daily_logs/archive/log_*.txt'`
- Prints an error message: `rm: cannot remove '/home/aadarshi/daily_logs/archive/log_*.txt': No such file or directory`

The script is executed, and a log file is generated inside the `/daily_logs/` folder containing system details.

Step 7: Archive old logs using the find and tar commands.



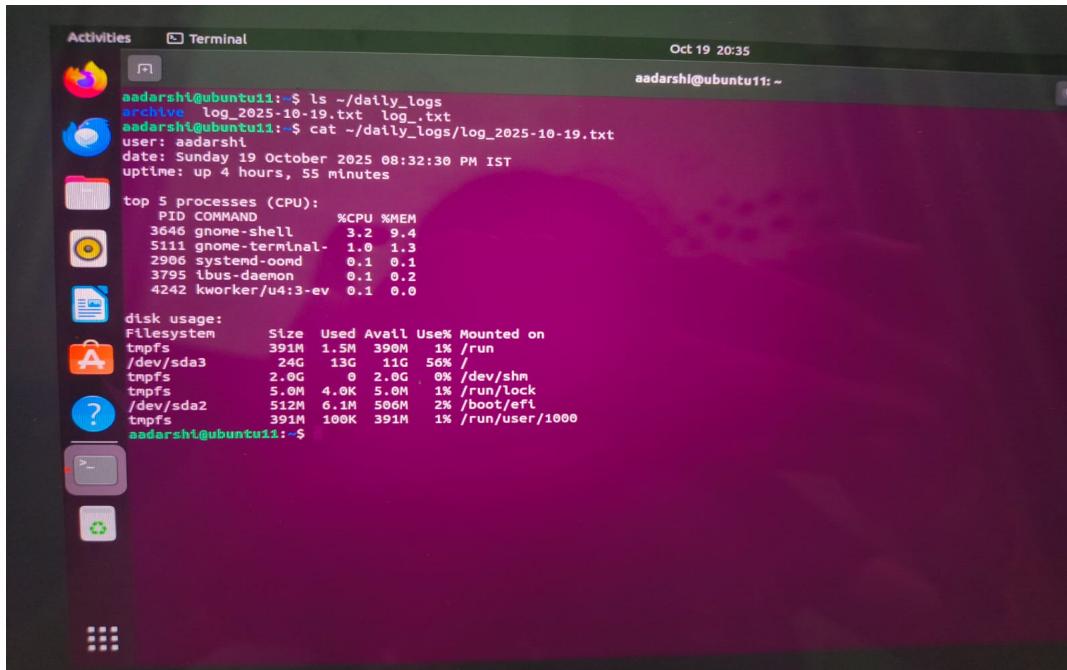
A screenshot of a Linux desktop environment showing a terminal window. The terminal window title is "Terminal". The date and time at the top right are "Oct 19 20:34". The user's name is "adarshi@ubuntu11: ~". The terminal shows the command:

```
adarshi@ubuntu11:~$ ls ~/daily_logs  
archive log_2025-10-19.txt log_.txt  
adarshi@ubuntu11:~$
```

The terminal window has a dark background with light-colored text. To the left of the terminal is a vertical dock containing icons for various applications like a browser, file manager, terminal, and system settings.

Old log files (older than 7 days) are archived into a weekly compressed file using tar.

Step 8: Schedule the script execution using cron job with `crontab -e`.

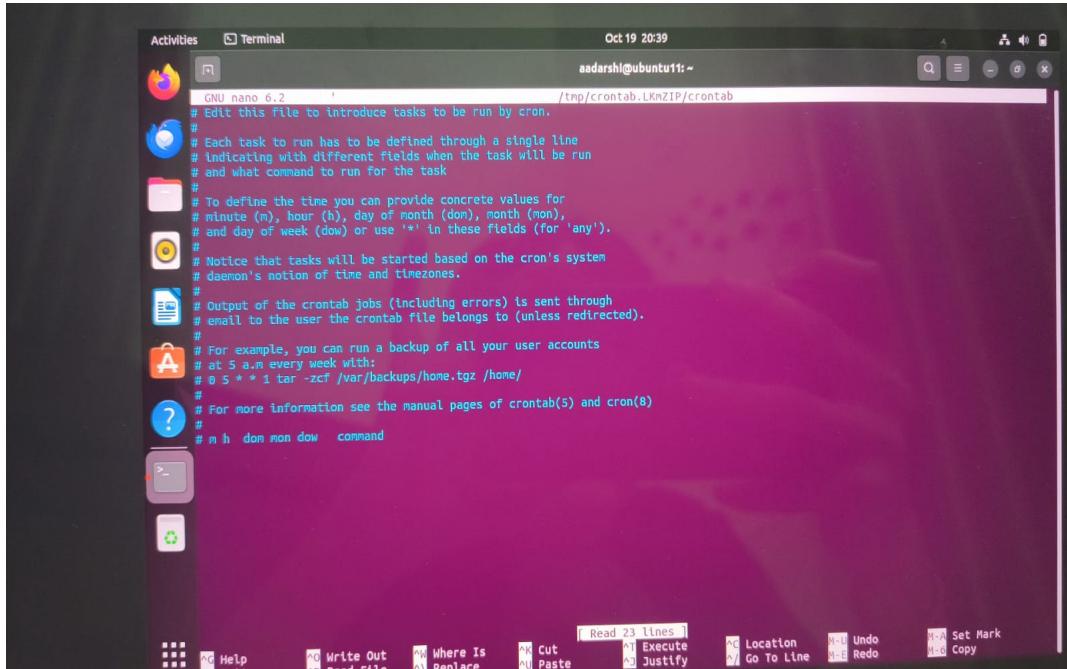


The screenshot shows a Linux desktop environment with a terminal window open. The terminal displays the following command-line session:

```
aadarsh@ubuntu11:~$ ls ~/daily_logs  
archive log_2025-10-19.txt log_.txt  
aadarsh@ubuntu11:~$ cat ~/daily_logs/log_2025-10-19.txt  
user: aadarsh  
date: Sunday 19 October 2025 08:32:30 PM IST  
uptime: up 4 hours, 55 minutes  
top 5 processes (CPU):  
PID COMMAND %CPU %MEM  
3646 gnome-shell 3.2 9.4  
5111 gnome-terminal- 1.0 1.3  
2906 systemd-ondemand 0.1 0.1  
3795 ibus-daemon 0.1 0.2  
4242 kworker/u4:3-ev 0.1 0.0  
disk usage:  
Filesystem Size Used Avail Use% Mounted on  
tmpfs 391M 1.5M 390M 1% /run  
/dev/sda3 24G 13G 11G 56% /  
tmpfs 2.0G 0 2.0G 0% /dev/shm  
tmpfs 5.0M 4.0K 5.0M 1% /run/lock  
/dev/sda2 512M 6.1M 506M 2% /boot/efi  
tmpfs 391M 100K 391M 1% /run/user/1000
```

The cron scheduler is used to automate script execution daily at 8:00 PM using `crontab -e`.

Step 9: Verify the cron job with `crontab -l` command.

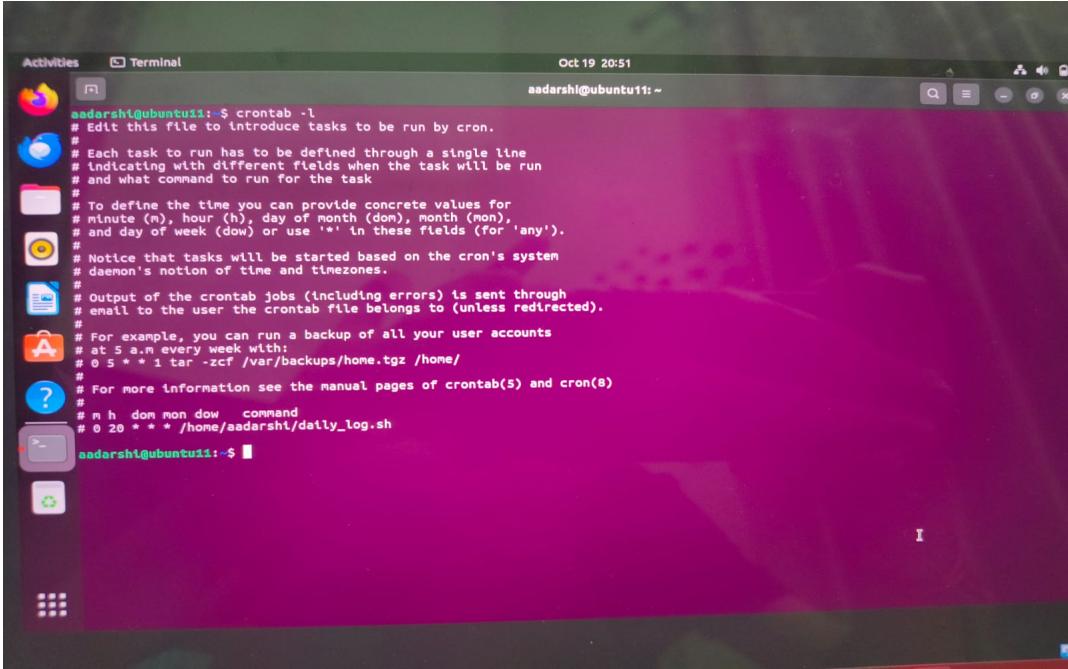


The screenshot shows a terminal window titled "Terminal" with the command "crontab -l" running. The window title bar includes "Activities", "Terminal", the date "Oct 19 20:39", and the user "sadarshl@ubuntu11: ~". The terminal content displays the crontab file, which contains comments explaining the syntax of cron jobs. It includes examples like a weekly backup command: "# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/". The bottom of the terminal shows a menu bar with options like "Help", "Write Out", "Where Is", "Cut", "Paste", "Execute", "Location", "Go To Line", "Undo", "Redo", "Set Mark", and "Copy".

```
# Edit this file to introduce tasks to be run by cron.  
#  
# Each task to run has to be defined through a single line  
# indicating with different fields when the task will be run  
# and what command to run for the task  
#  
# To define the time you can provide concrete values for  
# minute (m), hour (h), day of month (dom), month (mon),  
# and day of week (dow) or use '*' in these fields (for 'any').  
#  
# Notice that tasks will be started based on the cron's system  
# daemon's notion of time and timezones.  
#  
# Output of the crontab jobs (including errors) is sent through  
# email to the user the crontab file belongs to (unless redirected).  
#  
# For example, you can run a backup of all your user accounts  
# at 5 a.m every week with:  
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/  
#  
# For more information see the manual pages of crontab(5) and cron(8)  
#  
# m h dom mon dow   command
```

Verification of cron jobs ensures that the task is correctly scheduled and active.

Step 10: Check the final log and archive results.



The screenshot shows a Linux desktop environment with a terminal window open. The terminal window title is "Terminal" and the status bar shows the date and time as "Oct 19 20:51". The user is running the command "crontab -l" which lists the contents of the cron tab file. The output shows a cron entry:

```
aadarshi@ubuntu11:~$ crontab -l
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# n h dom mon dow   command
# 0 20 * * * /home/aadarshi/daily_log.sh
aadarshi@ubuntu11:~$
```

The final results are checked, confirming proper log generation and archiving.