



UNIVERSITY OF PETROLEUM AND ENERGY STUDIES (UPES)

Linux Practical Report

Submitted by:

Name: Aadarshi Bisht

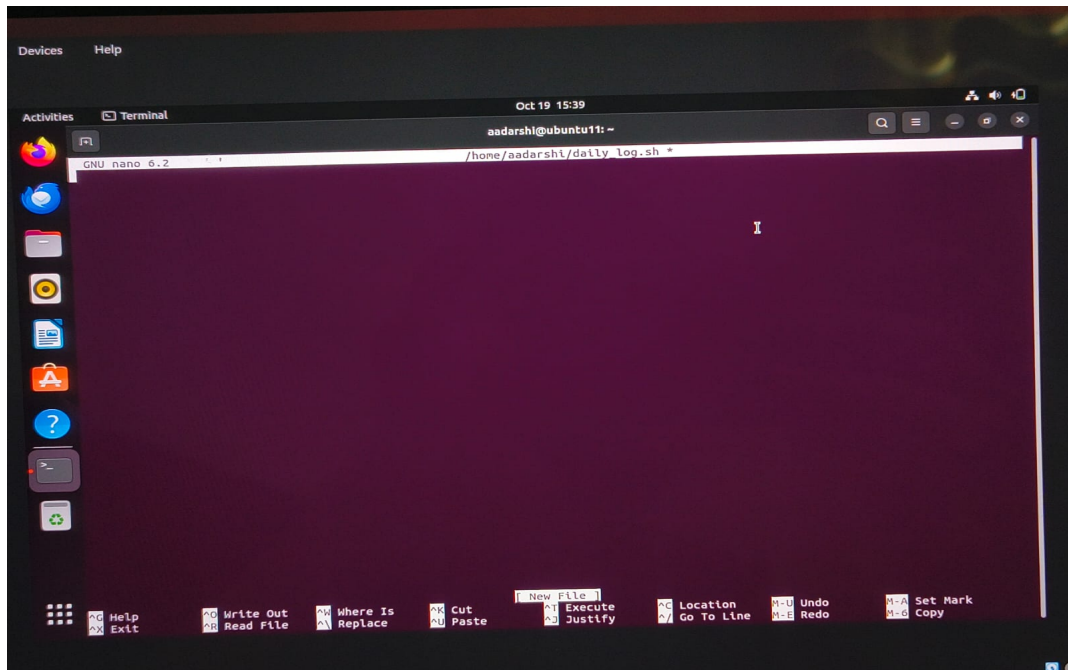
SAP ID: 590029330

Batch: 77-78

Subject: Linux

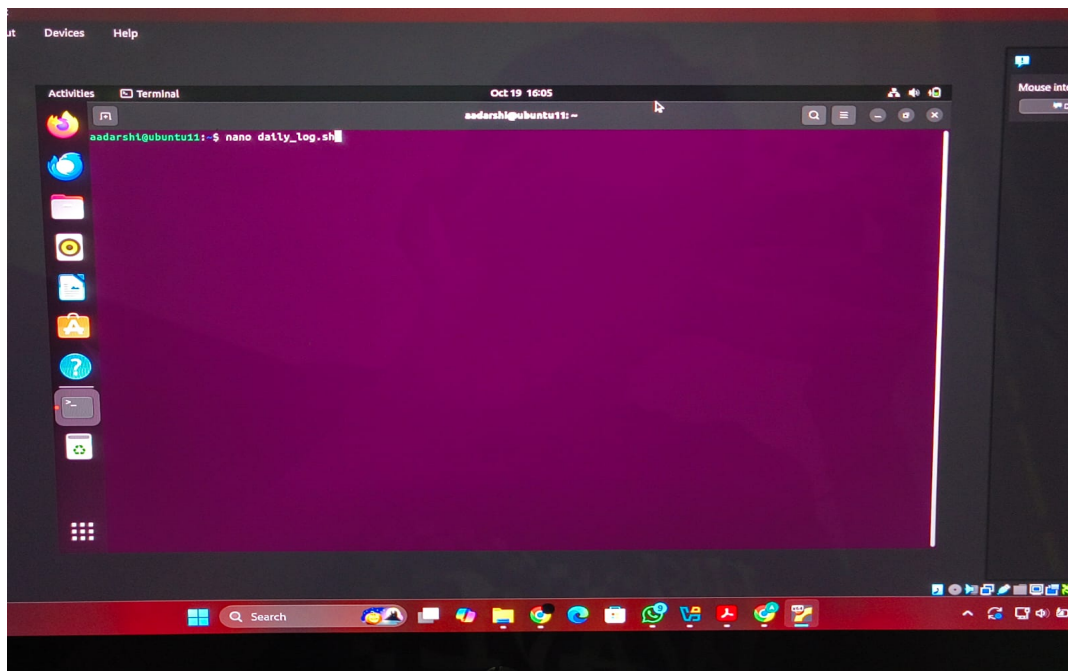
School of Computer Science

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



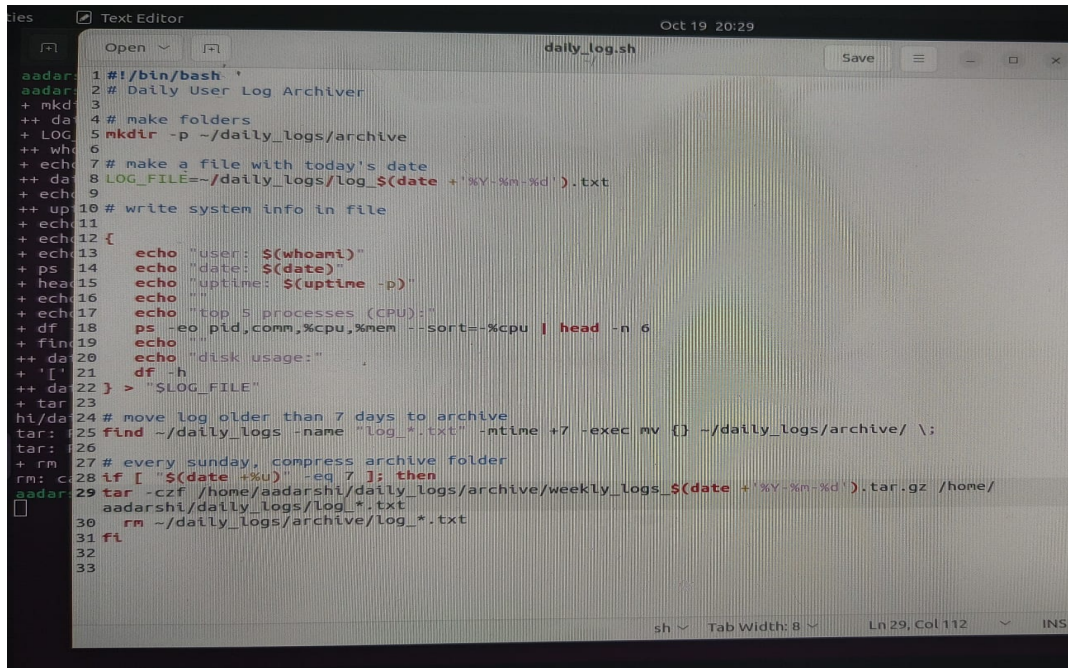
In this step, we create the `daily_log.sh` file using the nano text editor. This script will help automate daily system logging tasks.

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.

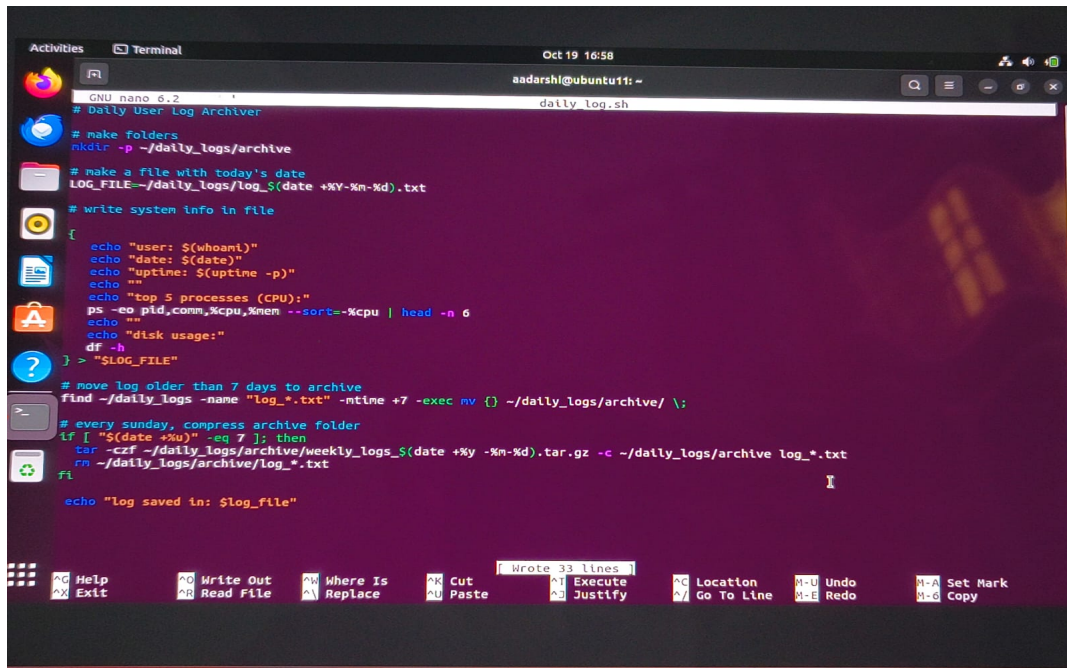
Step 3: Add file creation date and user details using `whoami` and `date`.



```
1#!/bin/bash
2# Daily User Log Archiver
3
4# make folders
5mkdir -p ~/daily_logs/archive
6
7# make a file with today's date
8LOG_FILE=~/daily_logs/log_$(date +%Y-%m-%d).txt
9
10# write system info in file
11
12{
13    echo "user: $(whoami)"
14    echo "date: $(date)"
15    echo "uptime: $(uptime -p)"
16    echo ""
17    echo "top 5 processes (CPU):"
18    ps -eo pid,comm,%cpu,%mem --sort=-%cpu | head -n 6
19    echo ""
20    echo "disk usage:"
21    df -h
22 } > "$LOG_FILE"
23
24# move log older than 7 days to archive
25find ~/daily_logs -name "log_*.txt" -mtime +7 -exec mv {} ~/daily_logs/archive/ \;
26
27# every sunday, compress archive folder
28if [ "$(date +%u)" -eq 7 ]; then
29    tar -czf /home/aadarshi/daily_logs/archive/weekly_logs_$(date +%Y-%m-%d).tar.gz /home/
30    aadarshi/daily_logs/log_*.txt
31    rm ~/daily_logs/archive/log_*.txt
32fi
33
```

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
# 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"

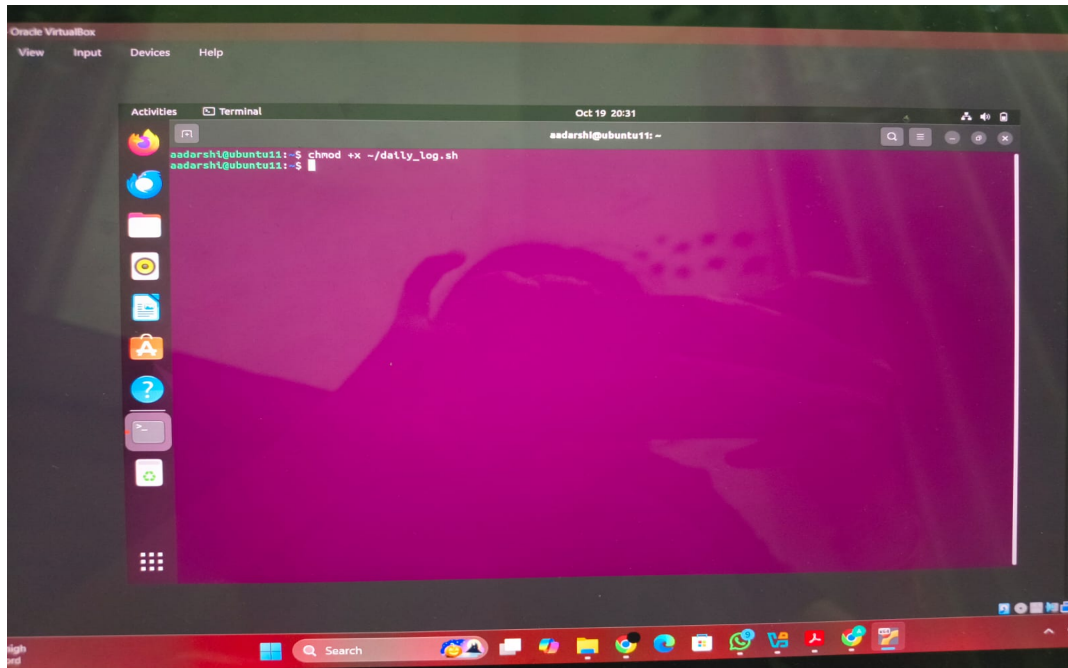
# move log older than 7 days to archive
find ~/daily_logs -name "log_*.txt" -mtime +7 -exec mv {} ~/daily_logs/archive/ \;

# every sunday, compress archive folder
if [ "$(date +%u)" = 7 ]; then
  tar -czf ~/daily_logs/archive/weekly_logs_$(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.

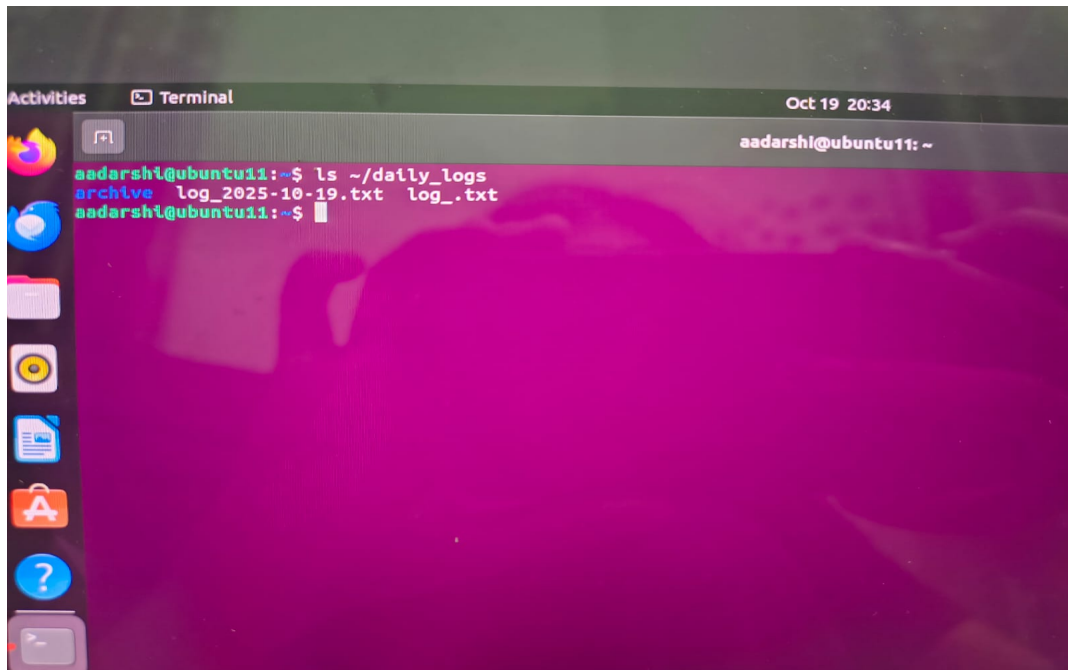
```

aadarshi@ubuntu11:~$ cat ~/daily_log.sh
+ mkdir -p /home/aadarshi/daily_logs/archive
+ date +%Y-%m-%d
+ LOG_FILE=/home/aadarshi/daily_logs/log_2025-10-19.txt
++ whoami
+ echo 'user: aadarshi'
+ date
+ echo 'date: Sunday 19 October 2025 08:32:30 PM IST'
++ uptime -p
+ echo 'uptime: up 4 hours, 55 minutes'
+ echo ''
+ echo 'top 5 processes (CPU):'
+ ps -eo ppid,comm,%cpu,%mem --sort=-%cpu
+ head -n 6
+ echo ''
+ echo 'disk usage:'
+ df -h
+ find /home/aadarshi/daily_logs -name 'log_*.txt' -mtime +7 -exec mv '{}' /home/aadarshi/daily_logs/archive/ ';'
++ date +%u
+ '[' 7 -eq 7 -j ]
++ date +%Y-%m-%d
+ tar -czf /home/aadarshi/daily_logs/archive/weekly_logs_2025-10-19.tar.gz /home/aadarshi/daily_logs/log_2025-10-19.txt /home/aadars
hi/daily_logs/log_*.txt
tar: Removing leading '/' from member names
tar: Removing leading '/' from hard link targets
+ rm /home/aadarshi/daily_logs/archive/log_*.txt
rnc: cannot remove '/home/aadarshi/daily_logs/archive/log_*.txt': No such file or directory
aadarshi@ubuntu11:~$

```

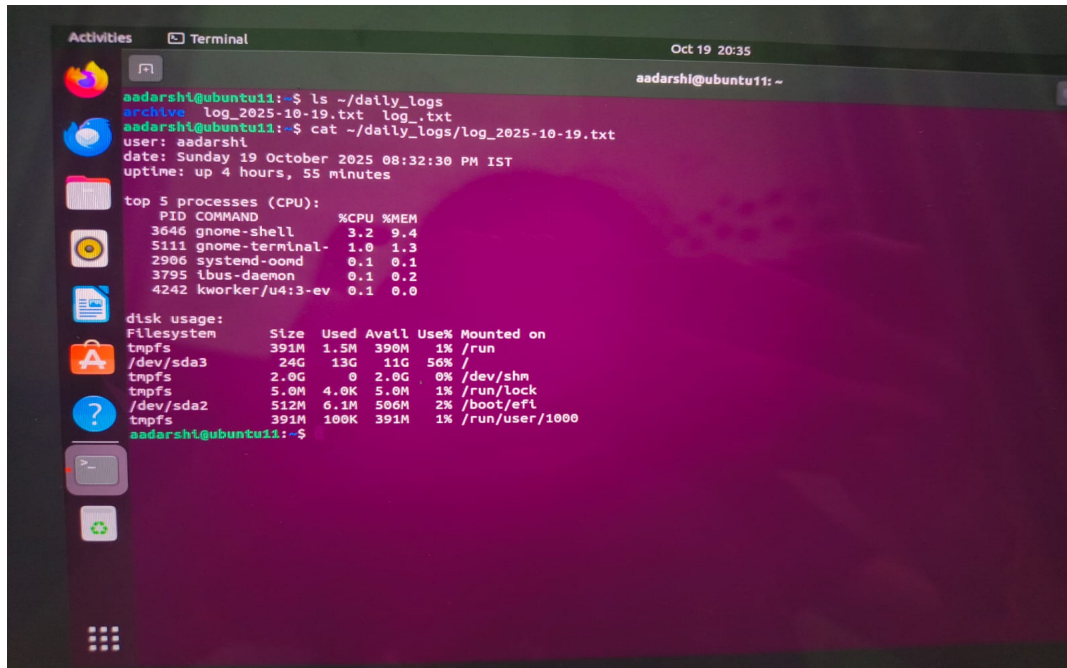
The script is executed, and a log file is generated inside the `/daily_logs/` folder containing system details.

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



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`.



A terminal window on Ubuntu 11. The user 'aadarshi' has executed several commands: `ls ~/daily_logs` (showing `archive`, `log_2025-10-19.txt`, and `log.txt`), `cat ~/daily_logs/log_2025-10-19.txt` (showing system date and uptime), `top` (showing top 5 processes), and `df -h` (showing disk usage). The terminal output is as follows:

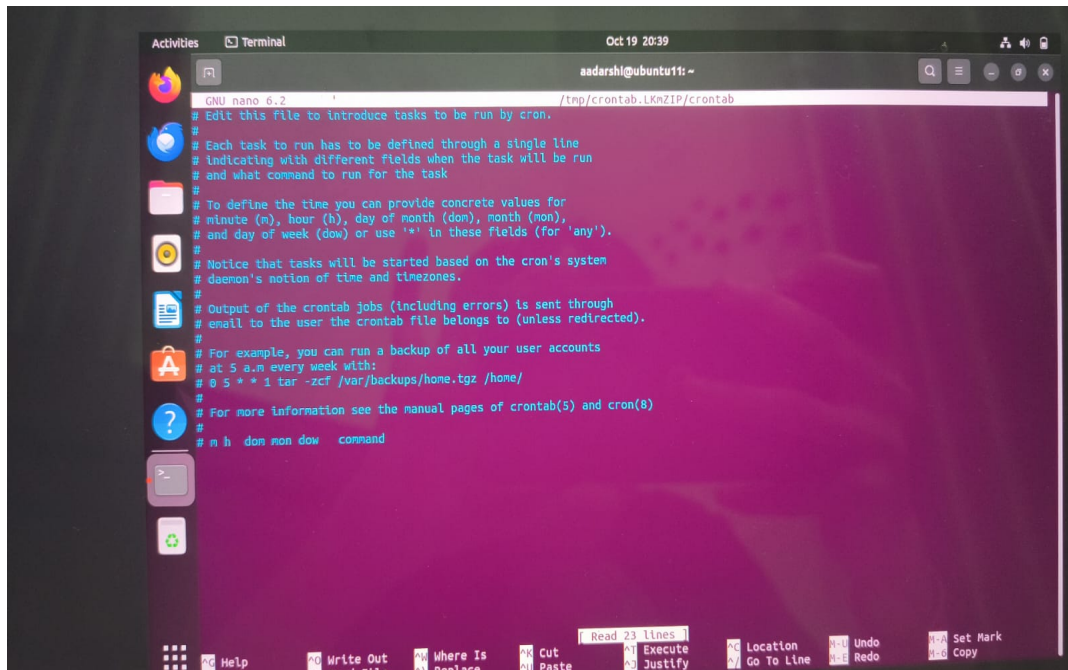
```
aadarshi@ubuntu11:~$ ls ~/daily_logs
archive  log_2025-10-19.txt  log.txt
aadarshi@ubuntu11:~$ cat ~/daily_logs/log_2025-10-19.txt
user: aadarshi
date: Sunday 19 October 2025 08:32:30 PM IST
uptime: up 4 hours, 55 minutes

top 5 processes (CPU):
  PID COMMAND                %CPU %MEM
  ---  ---                ---
 3646 gnome-shell              3.2  9.4
 5111 gnome-terminal-         1.0  1.3
 2906 systemd-oond            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.

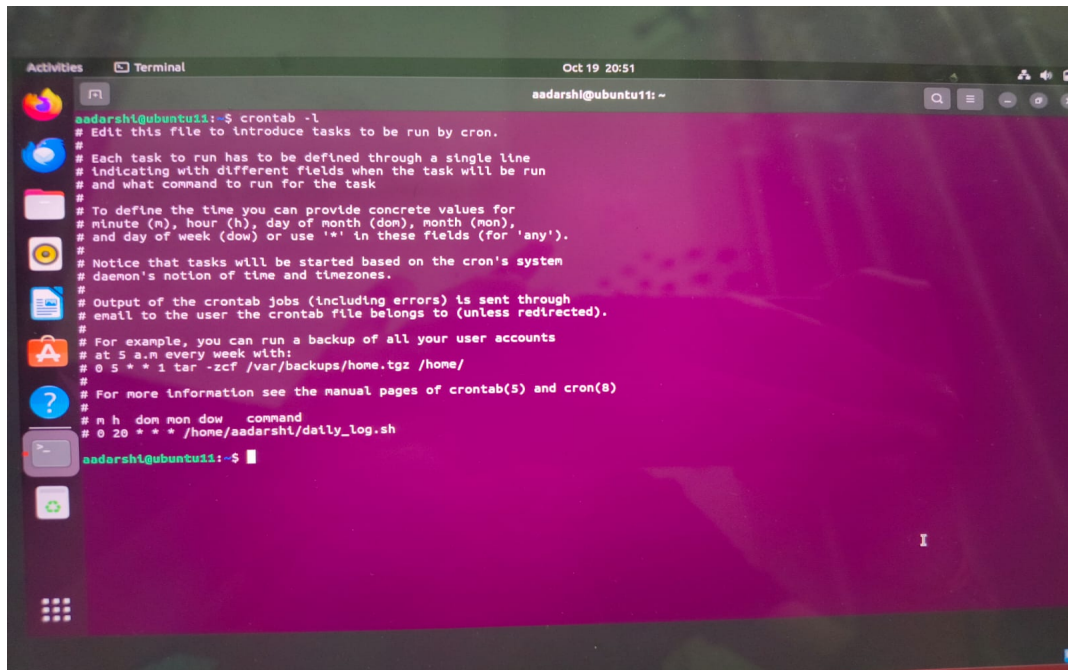


The screenshot shows a terminal window titled "Terminal" with the user "aadarsht@ubuntu11: ~". The terminal is displaying the content of the crontab file, which was opened using the command "cat /tmp/crontab.LKwZIP/crontab". The content includes instructions on how to define cron jobs, such as specifying time fields (minute, hour, day of month, month, day of week) and providing an example of a backup task: "0 5 * * 1 tar -zcf /var/backups/home.tgz /home/". The terminal also shows a status bar at the bottom with various menu options like "Read 23 lines", "Execute", "Justify", "Location", "Go To Line", "Undo", "Redo", "Set Mark", and "Copy".

```
GNU nano 6.2 /tmp/crontab.LKwZIP/crontab
# 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.

A terminal window titled 'Terminal' with a date and time of 'Oct 19 20:51'. The user 'aadarshi@ubuntu11' is at the prompt. They have run 'crontab -l' which displays the current crontab configuration. The configuration includes several comments explaining cron syntax and a single task: '0 20 * * * /home/aadarshi/daily_log.sh'. The prompt returns to '\$' after the command.

```
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)
#
# m 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.