

“Super Secret”

Linux Project

Task 1

- Create a folder called `super_secret_stuff` and inside that folder place a file called `top_secret.txt`.
- `top_secret.txt` may contain whatever content you wish.
- Once the file is created, use the `updatedb` command and the `locate` command to find the path to `top_secret.txt` .
- Using redirection, save the path that the `locate` command gives you to a new file called `secret_place.txt` in your home directory.

Task 2

Part A)

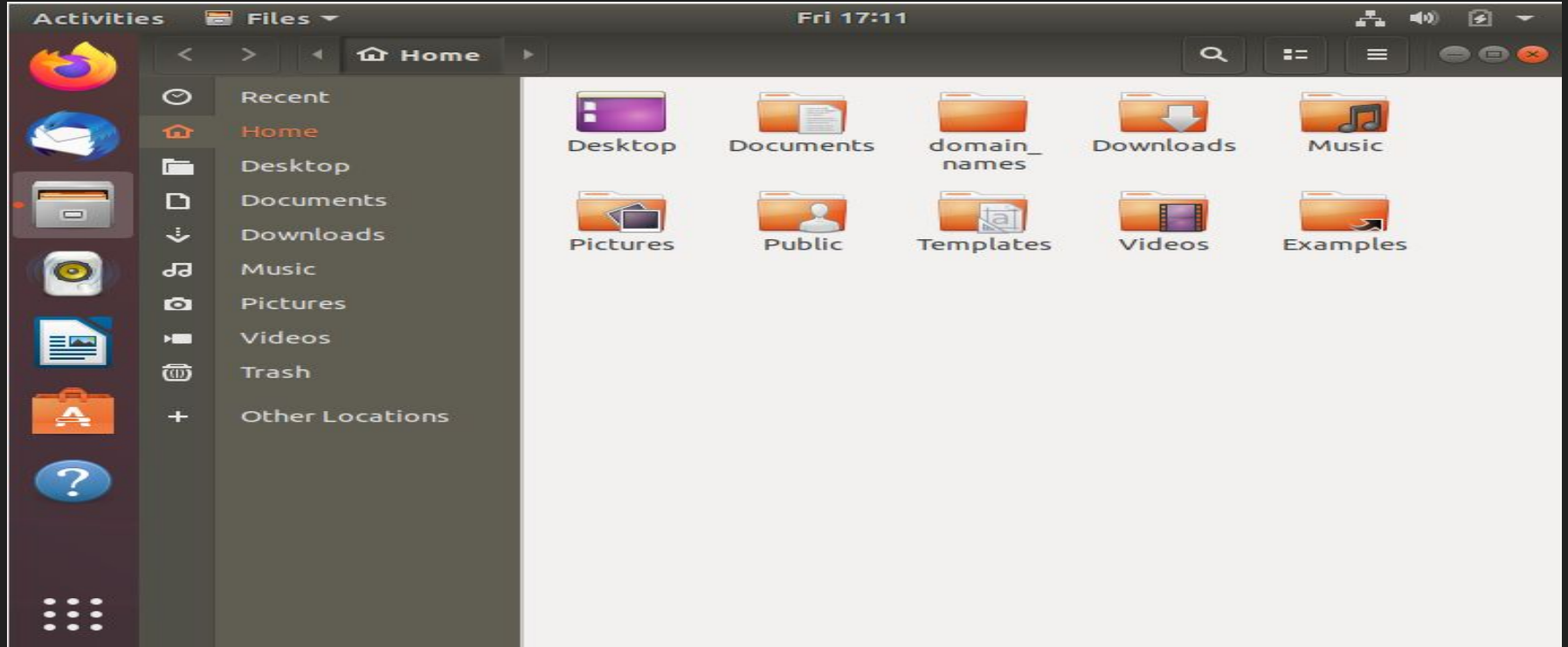
- Create an advanced pipeline that will create a sorted list of the various file sizes on your system.
- Firstly, use the find command to search entire file tree; starting from the / directory, for all files that are over 1 MebiByte in size.
- Use the maxdepth option to limit the find command's search to only go 4 levels deep.
- The search should only show files, not directories.
- Use the -exec option of the find command to run the ls -lh command on each of those results.

Task 2

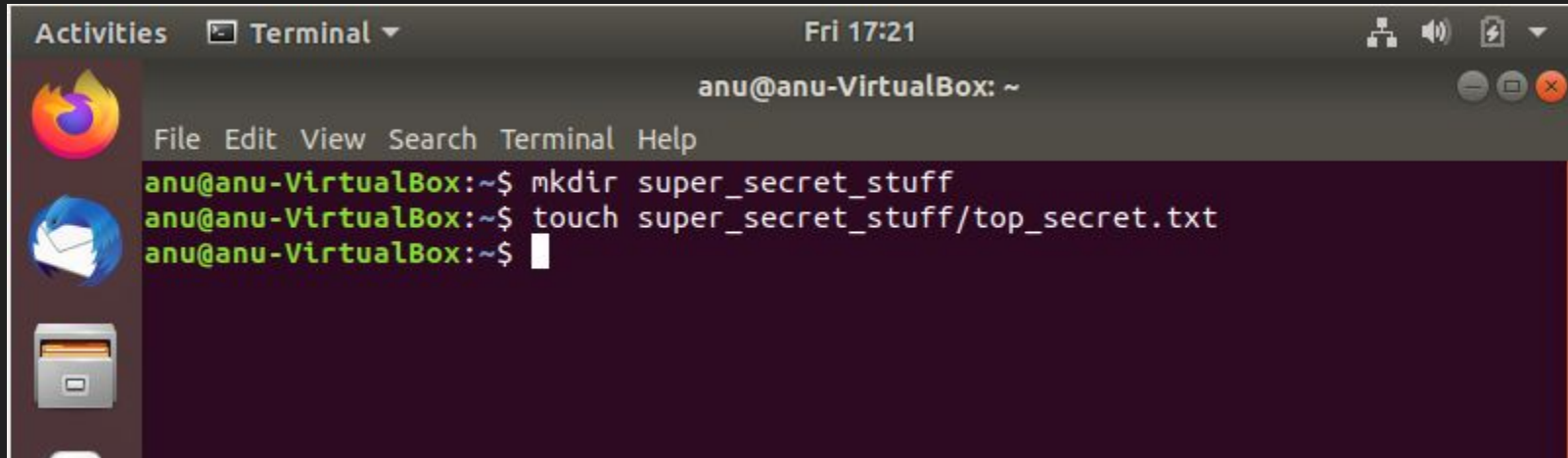
Part B)

- Sort the output from Part A using the sort command.
- You should sort the data so that the largest file sizes are at the top of the list and the smallest file sizes are at the bottom.
- Using redirection, output this data to a file called filesizes.txt in your home directory.
- Hint 1: You will need to use the `-k` option for the sort command and define a KEYDEF.
- Hint 2: The file sizes are the 5th column of data. Hint 3: You need to let the sort command to be able to deal with “human-readable” data.

Home page before starting the task



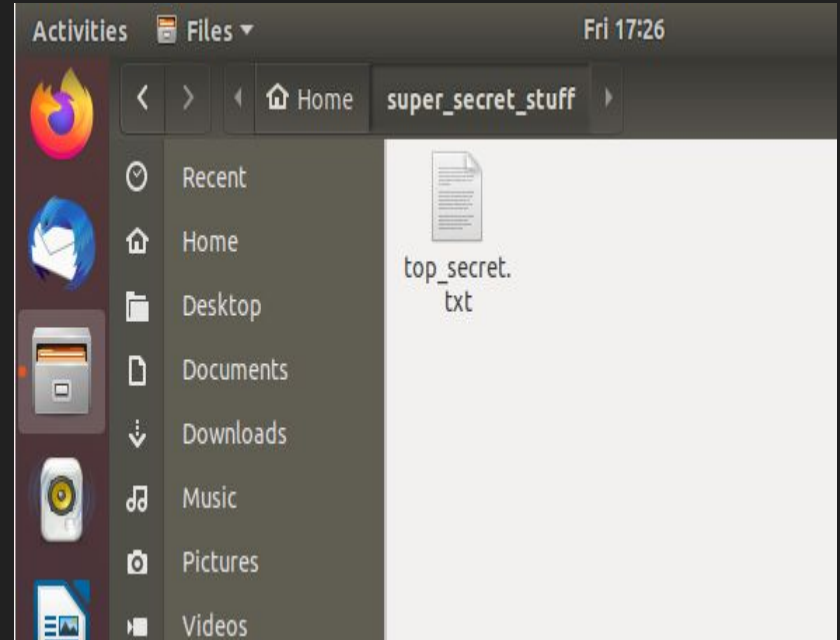
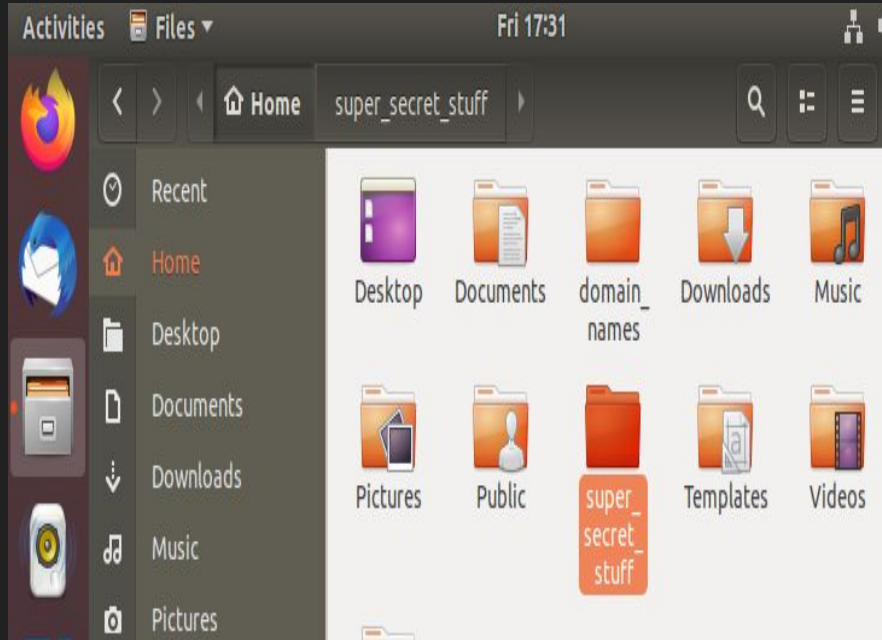
- Creating the folder “super_secret_stuff”.
- Creating the file “top_secret.txt” in the folder “super_secret_stuff”.



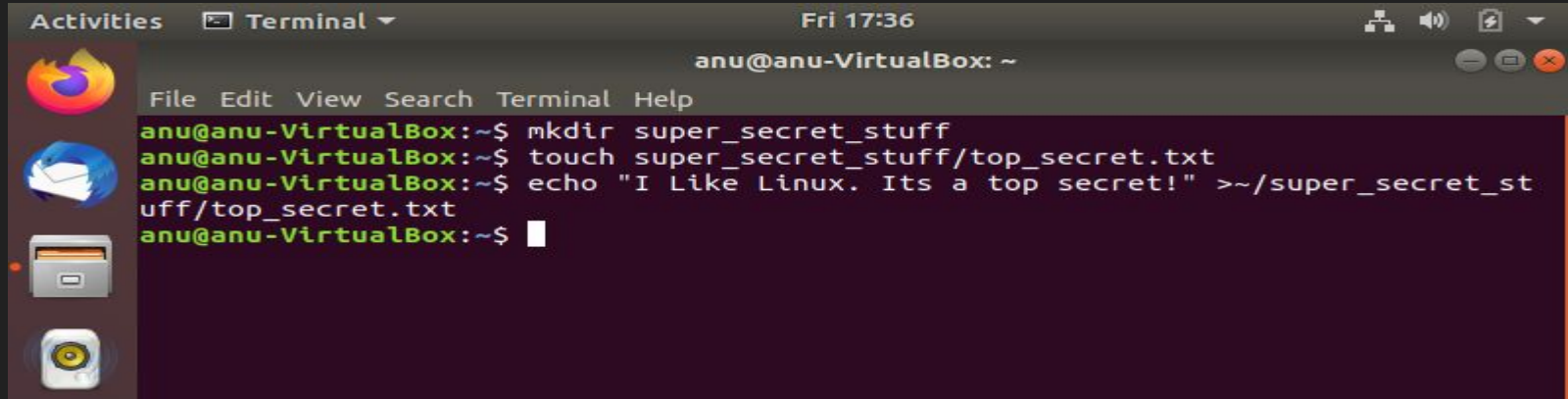
The screenshot shows a terminal window titled "Terminal" with a menu bar containing "File", "Edit", "View", "Search", "Terminal", and "Help". The window title bar also includes "Activities", "Fri 17:21", and system icons. The terminal prompt is "anu@anu-VirtualBox: ~". The user has entered two commands: "mkdir super_secret_stuff" and "touch super_secret_stuff/top_secret.txt". The terminal output shows the successful execution of these commands.

```
anu@anu-VirtualBox:~$ mkdir super_secret_stuff
anu@anu-VirtualBox:~$ touch super_secret_stuff/top_secret.txt
anu@anu-VirtualBox:~$
```

Now we see both the folder and file created in my home directory.

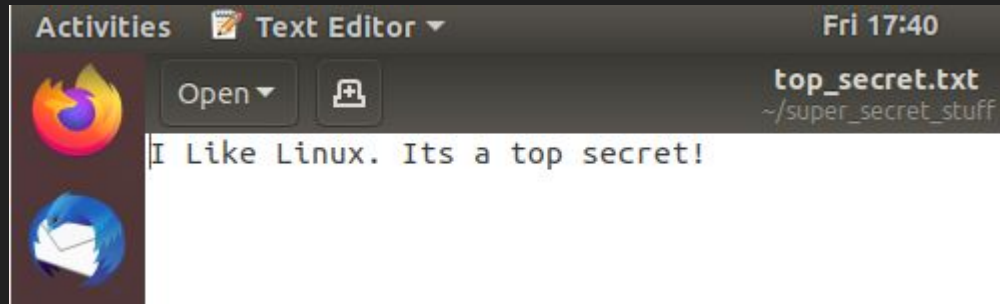


Using echo cmd redirecting some text message to the file “top_secret.txt”



A terminal window titled "Terminal" with a timestamp of "Fri 17:36". The user is "anu" on a system named "anu-VirtualBox". The terminal shows the following commands and output:

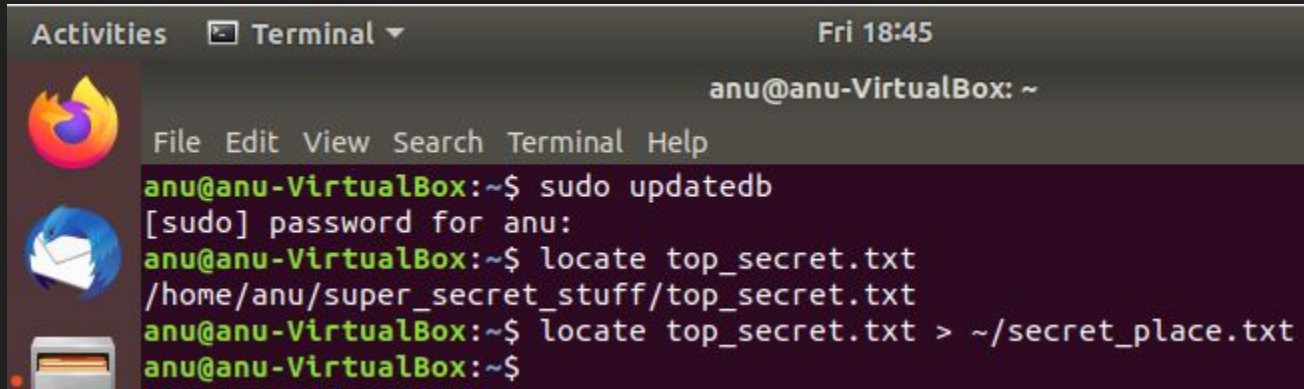
```
File Edit View Search Terminal Help
anu@anu-VirtualBox:~$ mkdir super_secret_stuff
anu@anu-VirtualBox:~$ touch super_secret_stuff/top_secret.txt
anu@anu-VirtualBox:~$ echo "I Like Linux. Its a top secret!" >~/super_secret_stuff/top_secret.txt
anu@anu-VirtualBox:~$
```



A text editor window titled "Text Editor" with a timestamp of "Fri 17:40". The file being edited is "top_secret.txt" located at "~/super_secret_stuff". The text inside the file is:

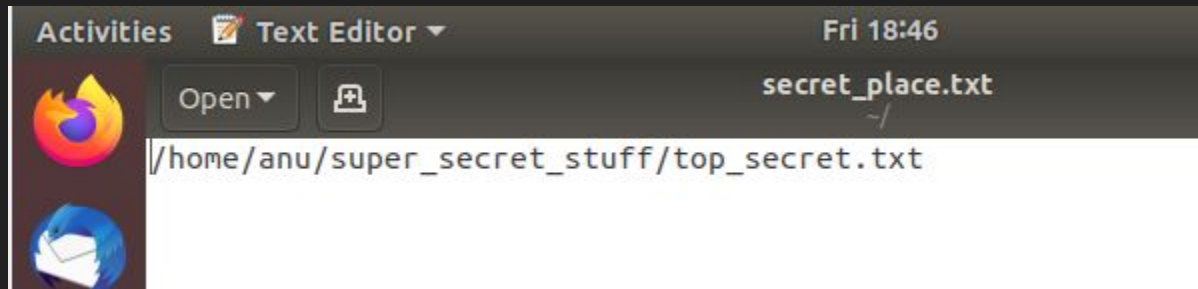
```
Open [icon]
top_secret.txt
~/super_secret_stuff
I Like Linux. Its a top secret!
```


Using Locate cmd to search for the file “top_secret.txt”



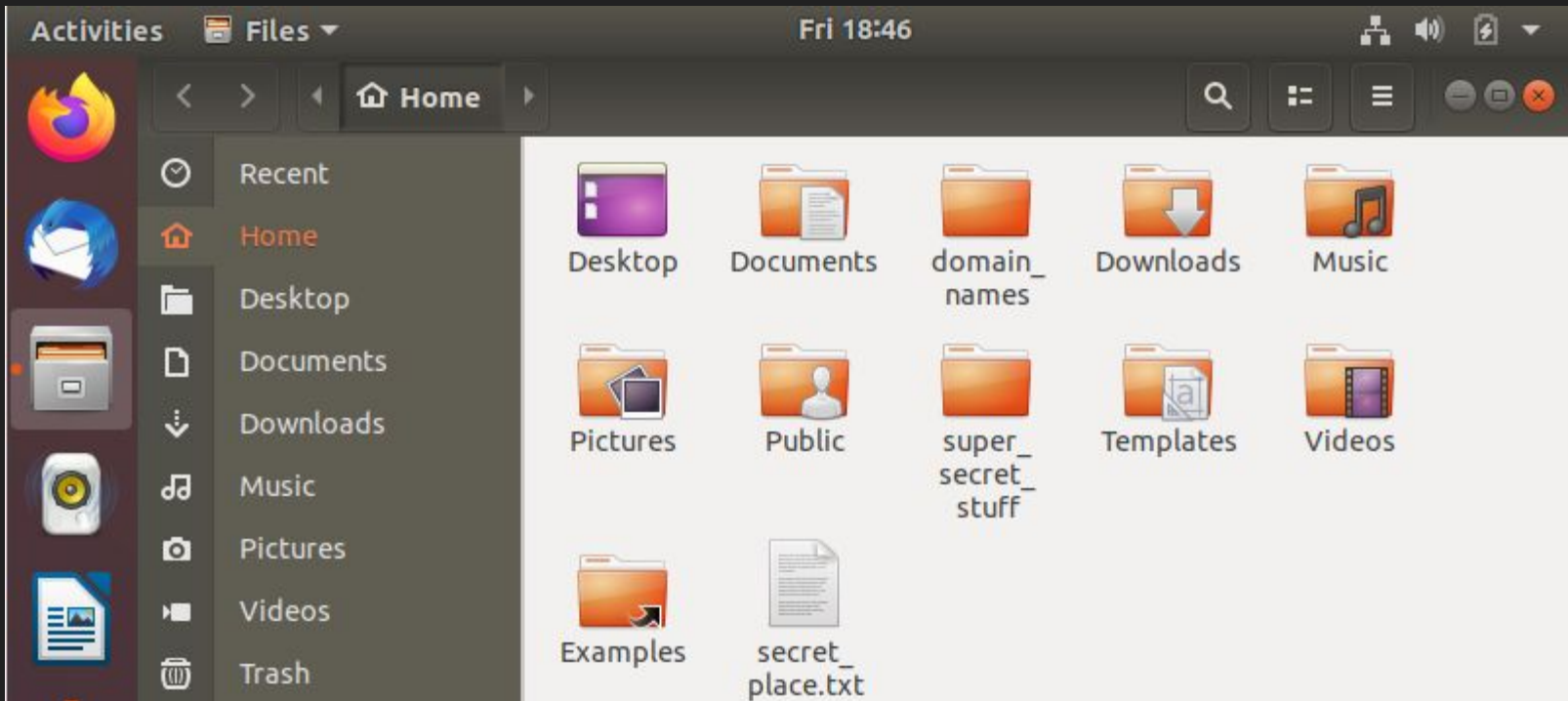
A terminal window titled "Terminal" with a timestamp of "Fri 18:45". The user "anu" is logged into "anu-VirtualBox". The terminal shows the following commands and output:

```
File Edit View Search Terminal Help
anu@anu-VirtualBox:~$ sudo updatedb
[sudo] password for anu:
anu@anu-VirtualBox:~$ locate top_secret.txt
/home/anu/super_secret_stuff/top_secret.txt
anu@anu-VirtualBox:~$ locate top_secret.txt > ~/secret_place.txt
anu@anu-VirtualBox:~$
```

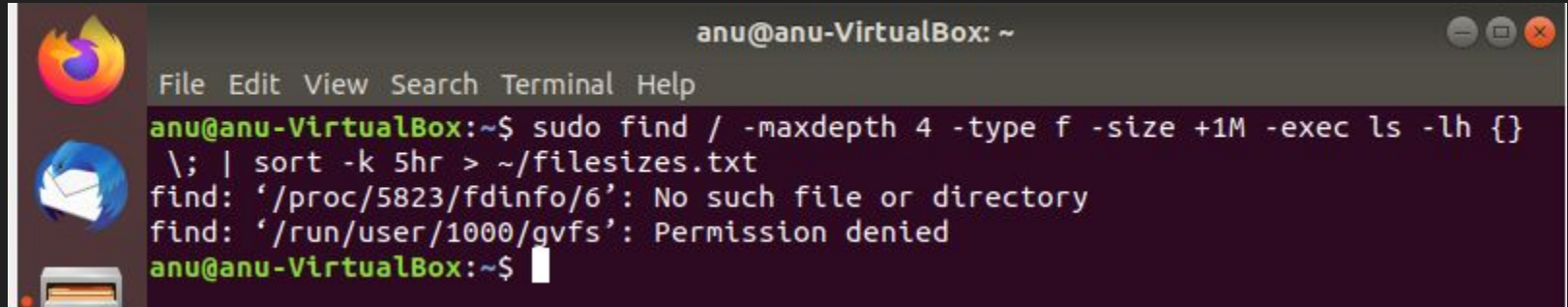


A text editor window titled "Text Editor" with a timestamp of "Fri 18:46". The file "secret_place.txt" is open, showing the path "/home/anu/super_secret_stuff/top_secret.txt".

```
secret_place.txt
~/
/home/anu/super_secret_stuff/top_secret.txt
```

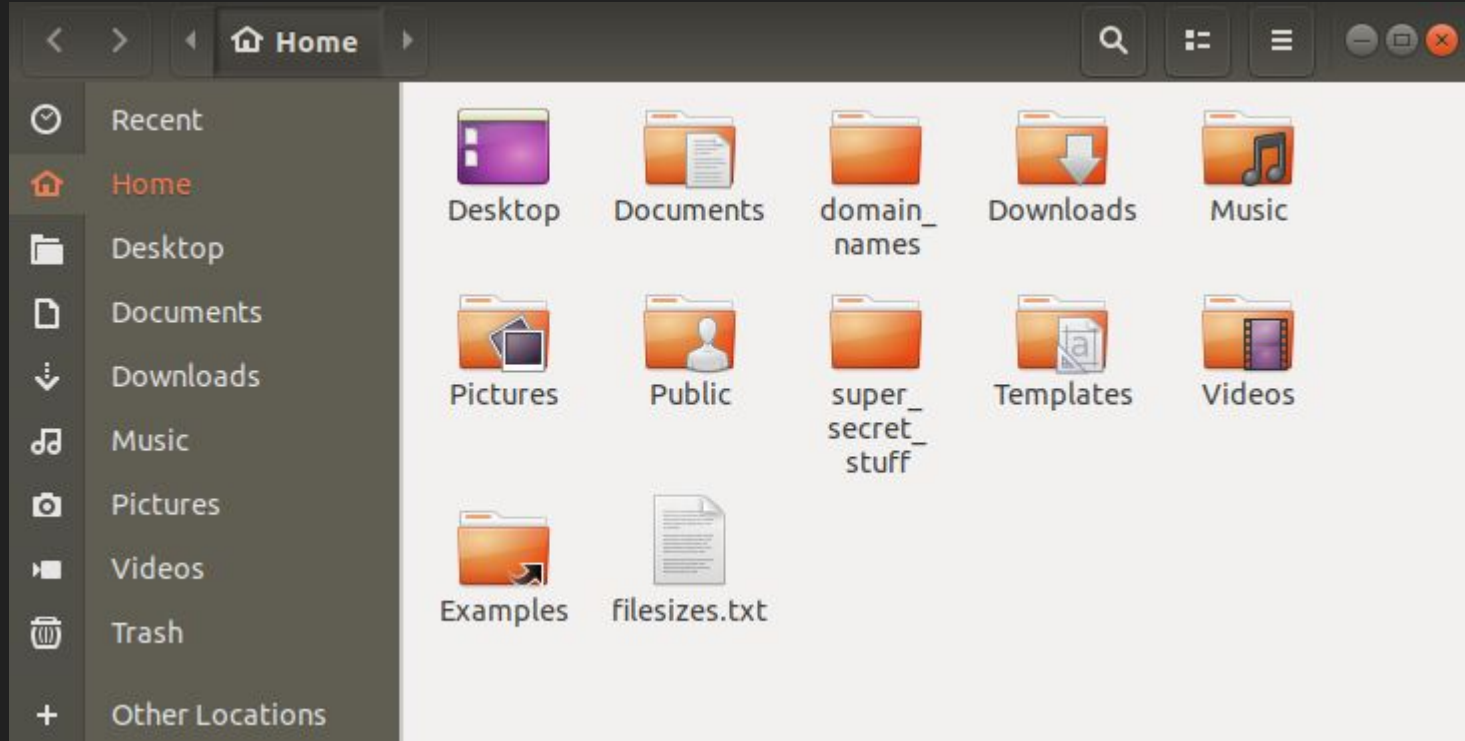


Piping the find command and sort command.

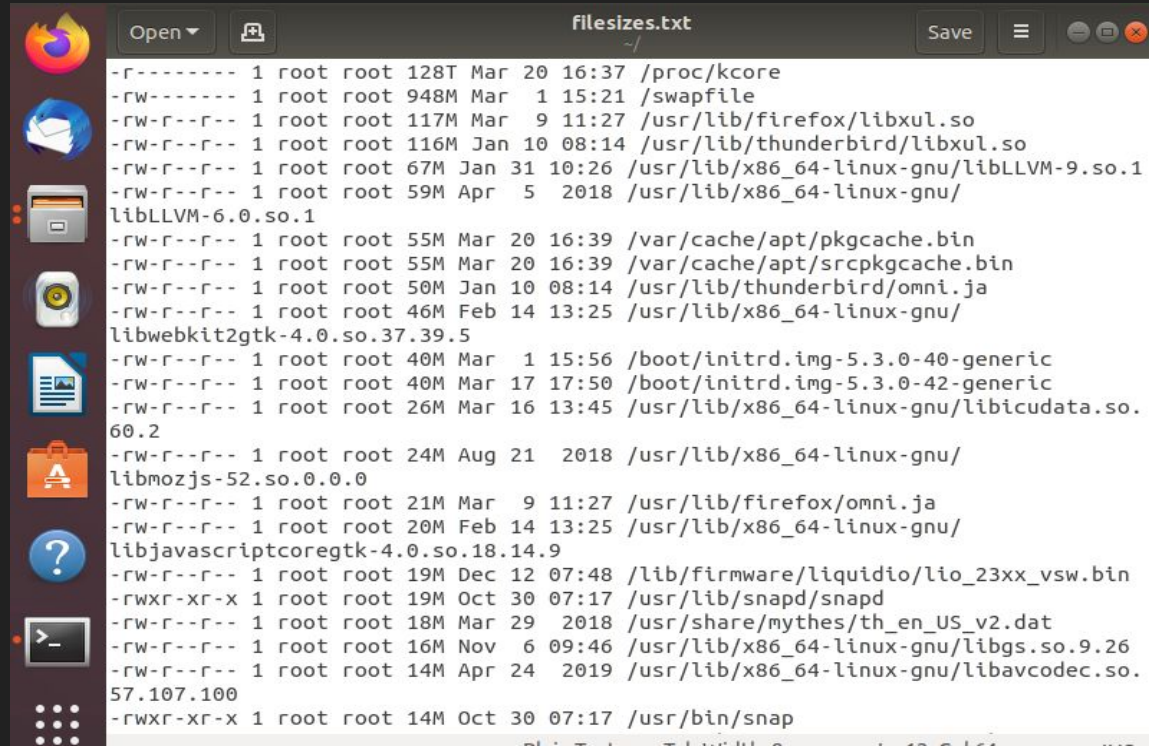
A terminal window titled 'anu@anu-VirtualBox: ~' with a menu bar (File, Edit, View, Search, Terminal, Help) and window control buttons. The terminal shows a command being executed: 'sudo find / -maxdepth 4 -type f -size +1M -exec ls -lh {} \; | sort -k 5hr > ~/filesizes.txt'. The output shows two error messages: 'find: '/proc/5823/fdinfo/6': No such file or directory' and 'find: '/run/user/1000/gvfs': Permission denied'. The prompt 'anu@anu-VirtualBox:~\$' is shown at the bottom with a cursor.

```
anu@anu-VirtualBox: ~  
File Edit View Search Terminal Help  
anu@anu-VirtualBox:~$ sudo find / -maxdepth 4 -type f -size +1M -exec ls -lh {}  
 \; | sort -k 5hr > ~/filesizes.txt  
find: '/proc/5823/fdinfo/6': No such file or directory  
find: '/run/user/1000/gvfs': Permission denied  
anu@anu-VirtualBox:~$
```

We can see the file “filesizes.txt” created in the home.



Contents of filesizes.txt . We can see that it has only list of files and they in the reverse sorted order according to the 5th column (human readable form)



```
Open filesizes.txt Save
-r----- 1 root root 128T Mar 20 16:37 /proc/kcore
-rw----- 1 root root 948M Mar 1 15:21 /swapfile
-rw-r--r-- 1 root root 117M Mar 9 11:27 /usr/lib/firefox/libxul.so
-rw-r--r-- 1 root root 116M Jan 10 08:14 /usr/lib/thunderbird/libxul.so
-rw-r--r-- 1 root root 67M Jan 31 10:26 /usr/lib/x86_64-linux-gnu/libLLVM-9.so.1
-rw-r--r-- 1 root root 59M Apr 5 2018 /usr/lib/x86_64-linux-gnu/
libLLVM-6.0.so.1
-rw-r--r-- 1 root root 55M Mar 20 16:39 /var/cache/apt/pkgcache.bin
-rw-r--r-- 1 root root 55M Mar 20 16:39 /var/cache/apt/srcpkgcache.bin
-rw-r--r-- 1 root root 50M Jan 10 08:14 /usr/lib/thunderbird/omni.ja
-rw-r--r-- 1 root root 46M Feb 14 13:25 /usr/lib/x86_64-linux-gnu/
libwebkit2gtk-4.0.so.37.39.5
-rw-r--r-- 1 root root 40M Mar 1 15:56 /boot/initrd.img-5.3.0-40-generic
-rw-r--r-- 1 root root 40M Mar 17 17:50 /boot/initrd.img-5.3.0-42-generic
-rw-r--r-- 1 root root 26M Mar 16 13:45 /usr/lib/x86_64-linux-gnu/libicudata.so.
60.2
-rw-r--r-- 1 root root 24M Aug 21 2018 /usr/lib/x86_64-linux-gnu/
libmozjs-52.so.0.0.0
-rw-r--r-- 1 root root 21M Mar 9 11:27 /usr/lib/firefox/omni.ja
-rw-r--r-- 1 root root 20M Feb 14 13:25 /usr/lib/x86_64-linux-gnu/
libjavascriptcoregtk-4.0.so.18.14.9
-rw-r--r-- 1 root root 19M Dec 12 07:48 /lib/firmware/liquidio/lio_23xx_vsw.bin
-rwxr-xr-x 1 root root 19M Oct 30 07:17 /usr/lib/snapd/snapd
-rw-r--r-- 1 root root 18M Mar 29 2018 /usr/share/mythes/th_en_US_v2.dat
-rw-r--r-- 1 root root 16M Nov 6 09:46 /usr/lib/x86_64-linux-gnu/libgs.so.9.26
-rw-r--r-- 1 root root 14M Apr 24 2019 /usr/lib/x86_64-linux-gnu/libavcodec.so.
57.107.100
-rwxr-xr-x 1 root root 14M Oct 30 07:17 /usr/bin/snap
```

“Hungry for Data” Project

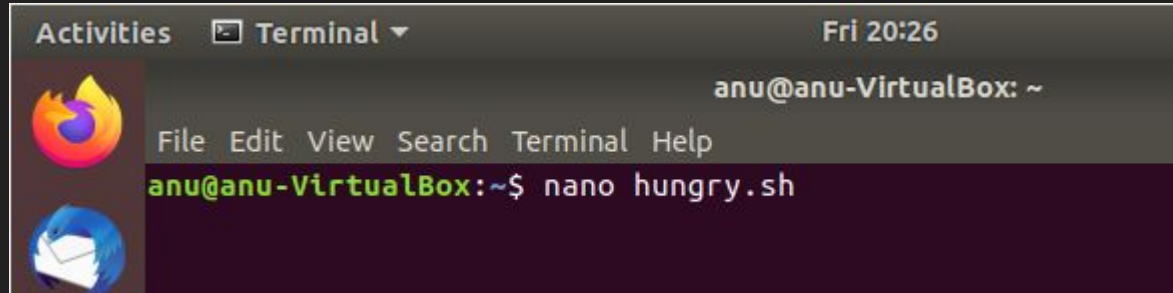
Task 1

- Create a bash script called hungry.sh in your home directory.
- hungry.sh should do two things:
- Firstly, it should output the text “I am hungry. Feed me data.” to a file in your home directory called demands.txt .
- Secondly, hungry.sh should also output the date and time that the demand was made to a file in your home directory called demands.log
- Do ensure that each output is appended to the previous one.

Task 2

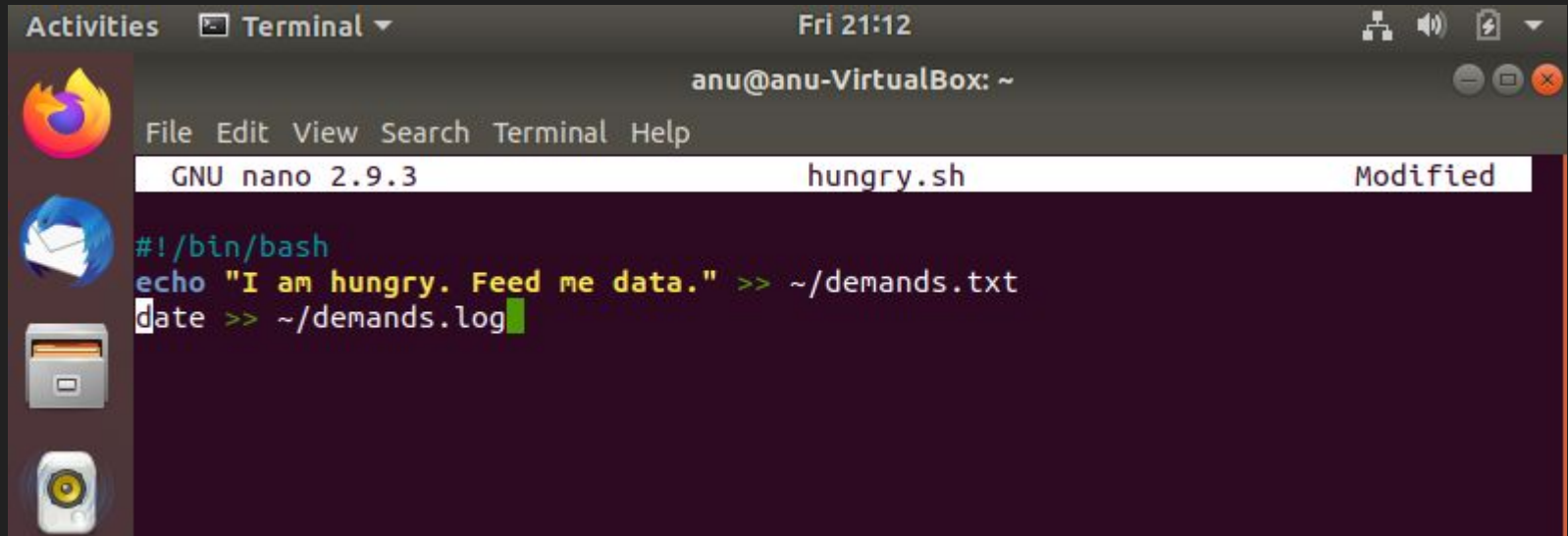
- Once you have created hungry.sh , edit your crontab and add a new row so that hungry.sh runs every minute.

Creating the bash script.



A terminal window titled 'Terminal' with a dropdown arrow. The top bar shows 'Fri 20:26'. The prompt is 'anu@anu-VirtualBox: ~'. The menu bar includes 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The command 'nano hungry.sh' has been entered at the prompt.

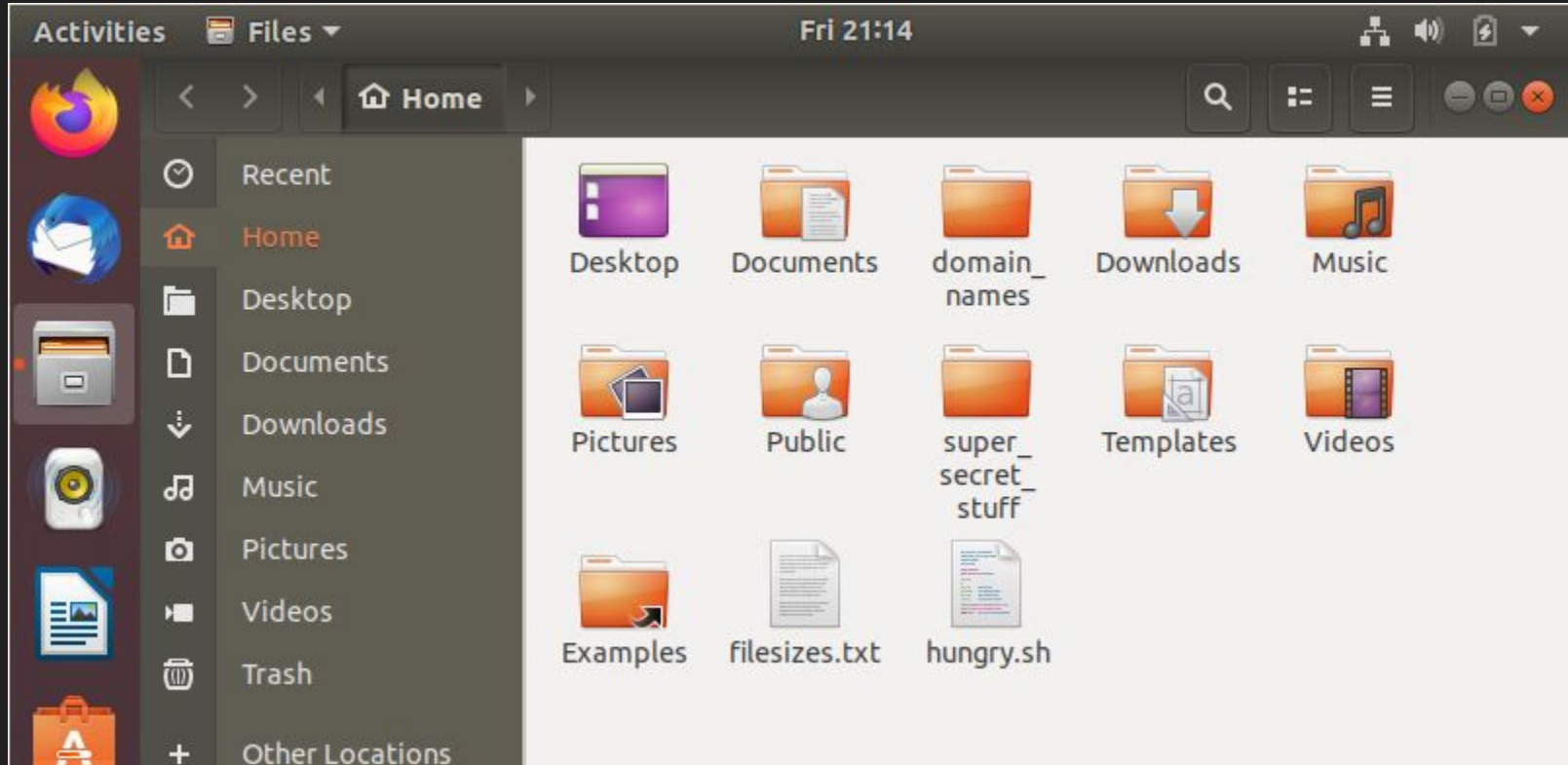
```
anu@anu-VirtualBox: ~$ nano hungry.sh
```



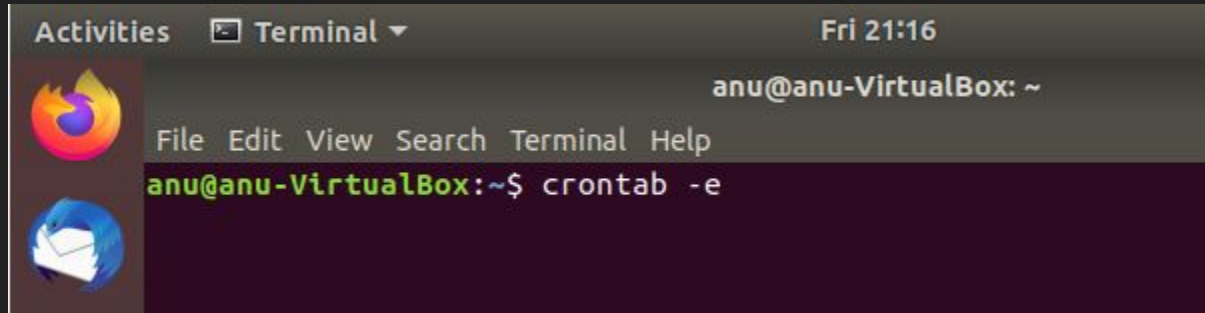
A terminal window titled 'Terminal' with a dropdown arrow. The top bar shows 'Fri 21:12'. The prompt is 'anu@anu-VirtualBox: ~'. The menu bar includes 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. A status bar at the top of the editor area shows 'GNU nano 2.9.3', 'hungry.sh', and 'Modified'. The script content is as follows:

```
#!/bin/bash
echo "I am hungry. Feed me data." >> ~/demands.txt
date >> ~/demands.log
```


We can see the bash file created.



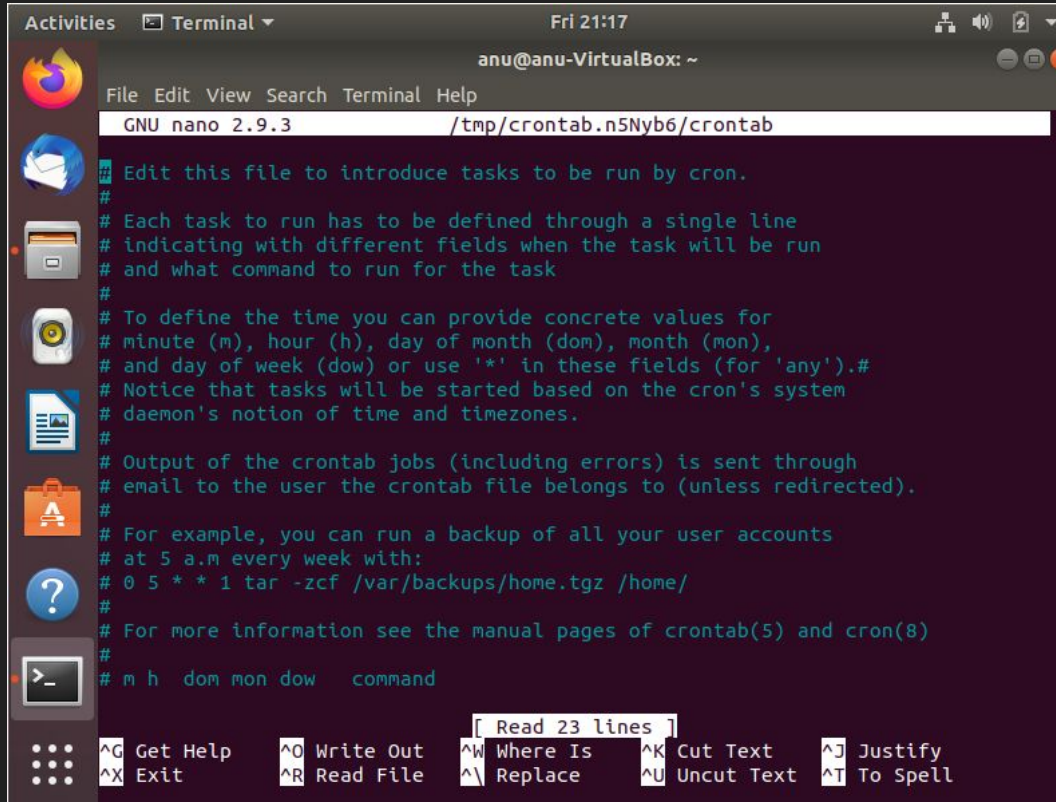
Command to edit the crontab.



A screenshot of a Linux terminal window. The window title bar shows 'Activities', 'Terminal', and 'Fri 21:16'. The terminal content shows the prompt 'anu@anu-VirtualBox: ~' followed by a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. Below the menu bar, the command 'anu@anu-VirtualBox:~\$ crontab -e' has been entered. On the left side of the terminal, there are two icons: the Firefox logo and a blue icon representing a document or folder.

```
Activities Terminal Fri 21:16
anu@anu-VirtualBox: ~
File Edit View Search Terminal Help
anu@anu-VirtualBox:~$ crontab -e
```

Editing the crontab to schedule the bash script.



The screenshot shows a terminal window titled "anu@anu-VirtualBox: ~" with a menu bar (File, Edit, View, Search, Terminal, Help) and a status bar (Fri 21:17). The terminal is running the GNU nano 2.9.3 text editor, editing the file `/tmp/crontab.n5Nyb6/crontab`. The editor's content includes instructions on how to define cron tasks, such as specifying time fields (minute, hour, day of month, month, day of week) and providing an example of a backup task. The bottom of the screen displays a status bar with various keyboard shortcuts for navigating and editing the file.

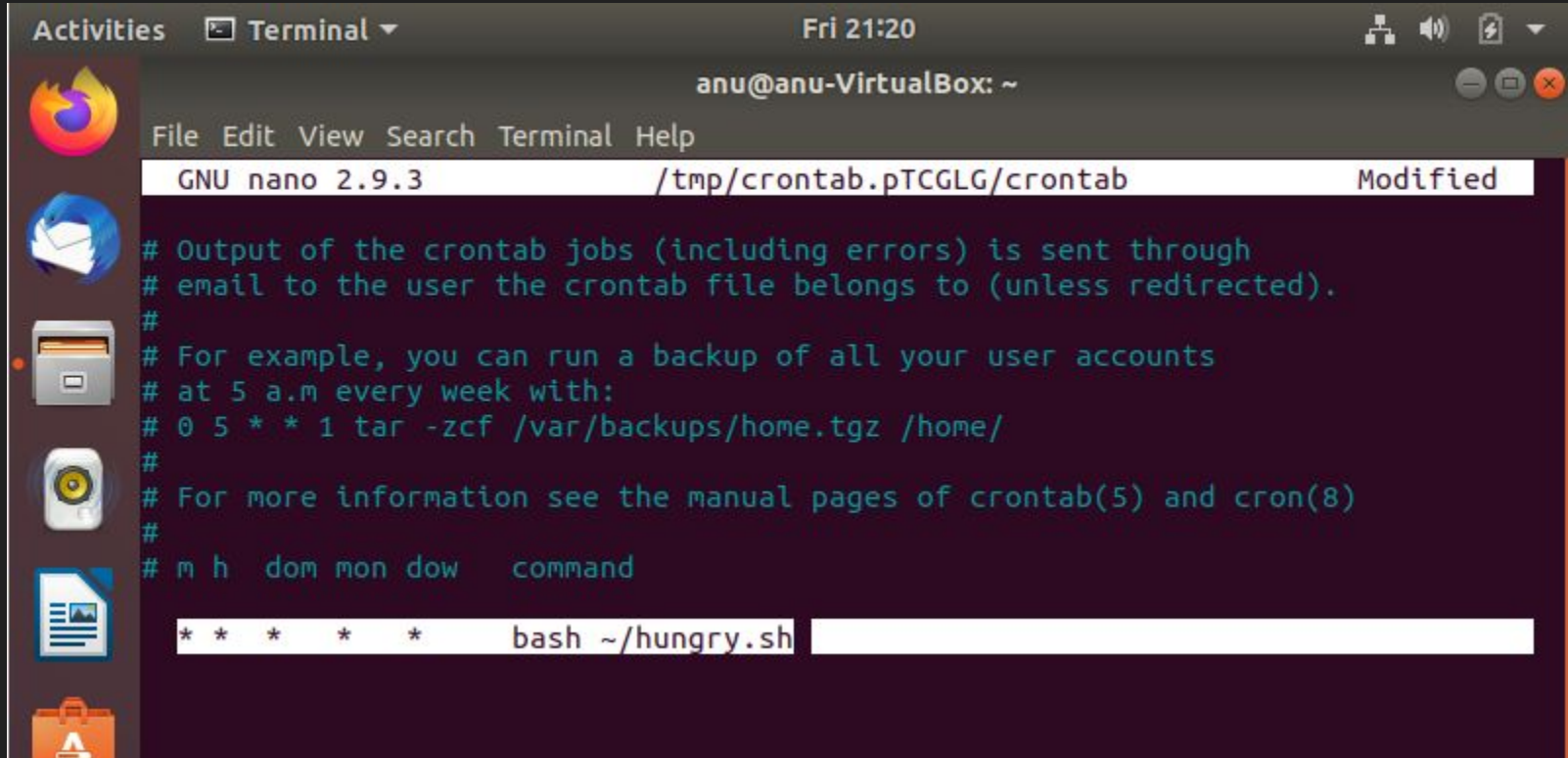
```
File Edit View Search Terminal Help
GNU nano 2.9.3 /tmp/crontab.n5Nyb6/crontab

# 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
```

[Read 23 lines]

Get Help	Write Out	Where Is	Cut Text	Justify
Exit	Read File	Replace	Uncut Text	To Spell

Scheduling the bash script “hungry.sh”.



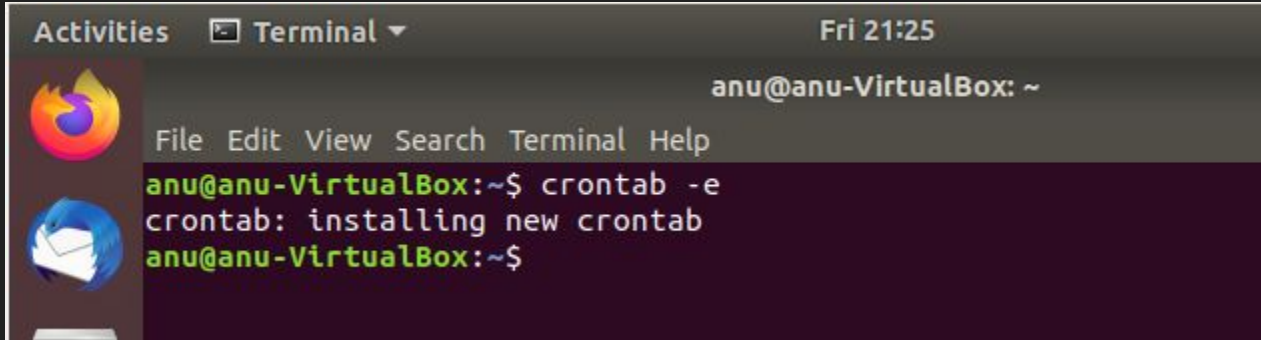
The screenshot shows a Linux desktop environment with a terminal window open. The terminal window has a title bar that reads "Fri 21:20" and "anu@anu-VirtualBox: ~". Below the title bar is a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". The main content of the terminal is a nano text editor editing a file named "/tmp/crontab.pTCGLG/crontab". The editor shows the following text:

```
GNU nano 2.9.3 /tmp/crontab.pTCGLG/crontab Modified

# 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
* * * * * bash ~/hungry.sh
```

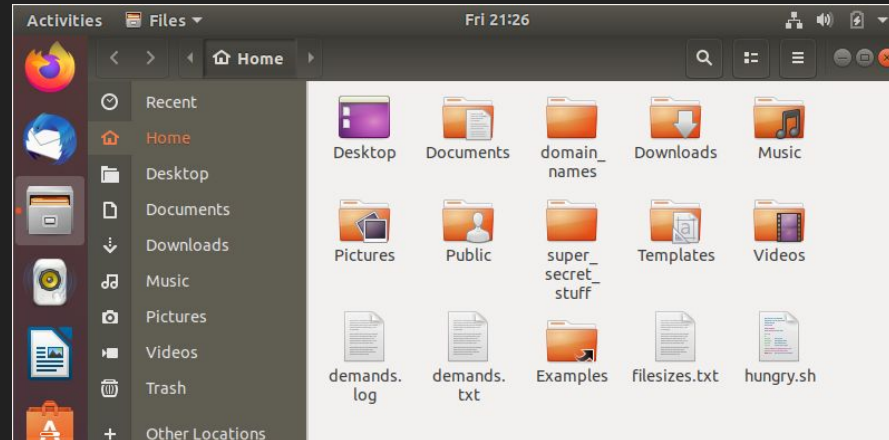
The terminal window also shows a sidebar on the left with icons for various applications, including a web browser, a mail client, a file manager, and a terminal.

We can that our bash script is working

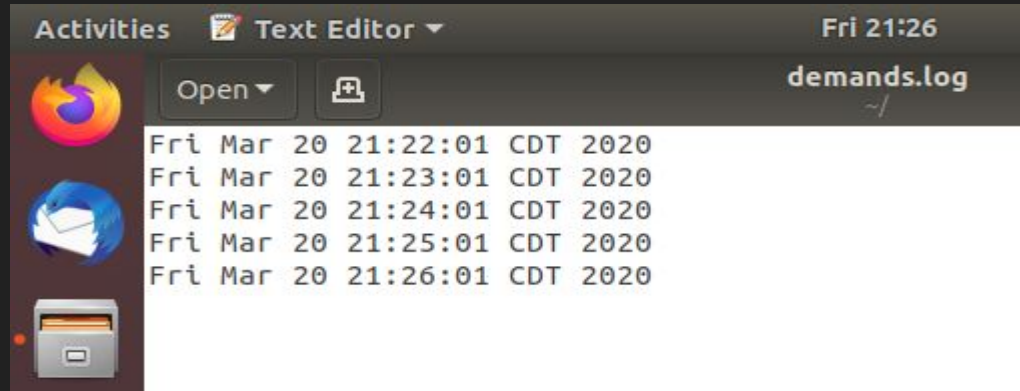


A terminal window titled 'Terminal' with a timestamp of 'Fri 21:25'. The user 'anu' is logged into 'anu-VirtualBox' at the home directory '~'. The terminal shows the command 'crontab -e' being executed, which results in the message 'crontab: installing new crontab'. The prompt returns to 'anu@anu-VirtualBox:~\$'.

```
anu@anu-VirtualBox:~$ crontab -e
crontab: installing new crontab
anu@anu-VirtualBox:~$
```



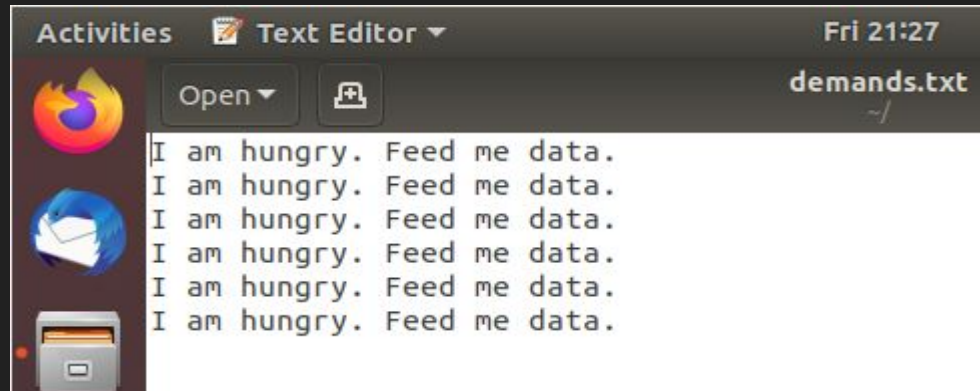
These files are created as a result of our bash script and we can see that our bash script is executing every minute, as we scheduled.



Activities Text Editor ▾ Fri 21:26

demands.log
~/

```
Fri Mar 20 21:22:01 CDT 2020
Fri Mar 20 21:23:01 CDT 2020
Fri Mar 20 21:24:01 CDT 2020
Fri Mar 20 21:25:01 CDT 2020
Fri Mar 20 21:26:01 CDT 2020
```



Activities Text Editor ▾ Fri 21:27

demands.txt
~/

```
I am hungry. Feed me data.
I am hungry. Feed me data.
I am hungry. Feed me data.
I am hungry. Feed me data.
I am hungry. Feed me data.
I am hungry. Feed me data.
```

Thank You !!