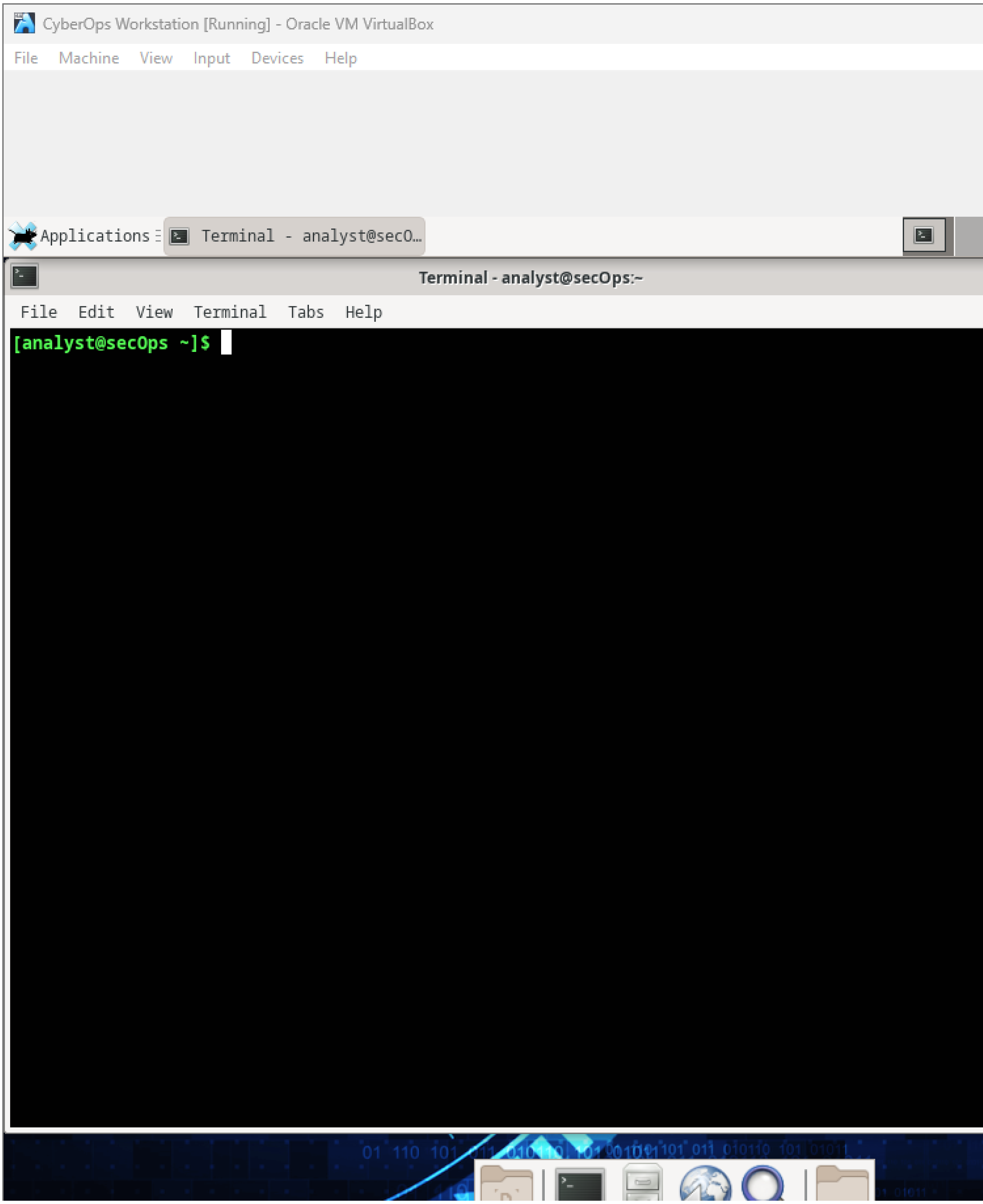
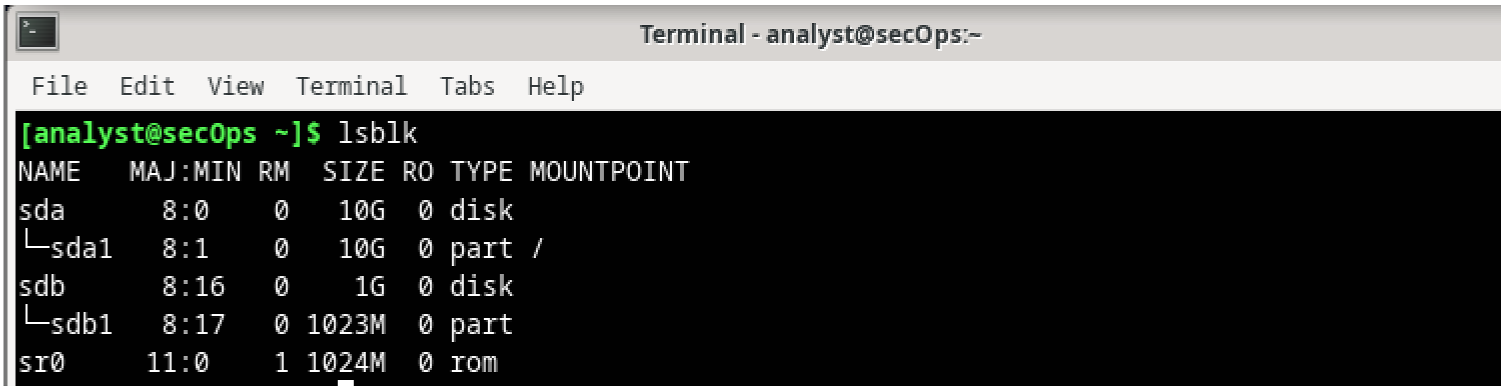
Name: Joseph Dela Cruz Jr.
Course: 4th year BSIT
Part 1
Step 1 Open Command line


CyberOps Workstation [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications
Terminal - analyst@secO…
Terminal - analyst@secOps:~
File Edit View Terminal Tabs Help

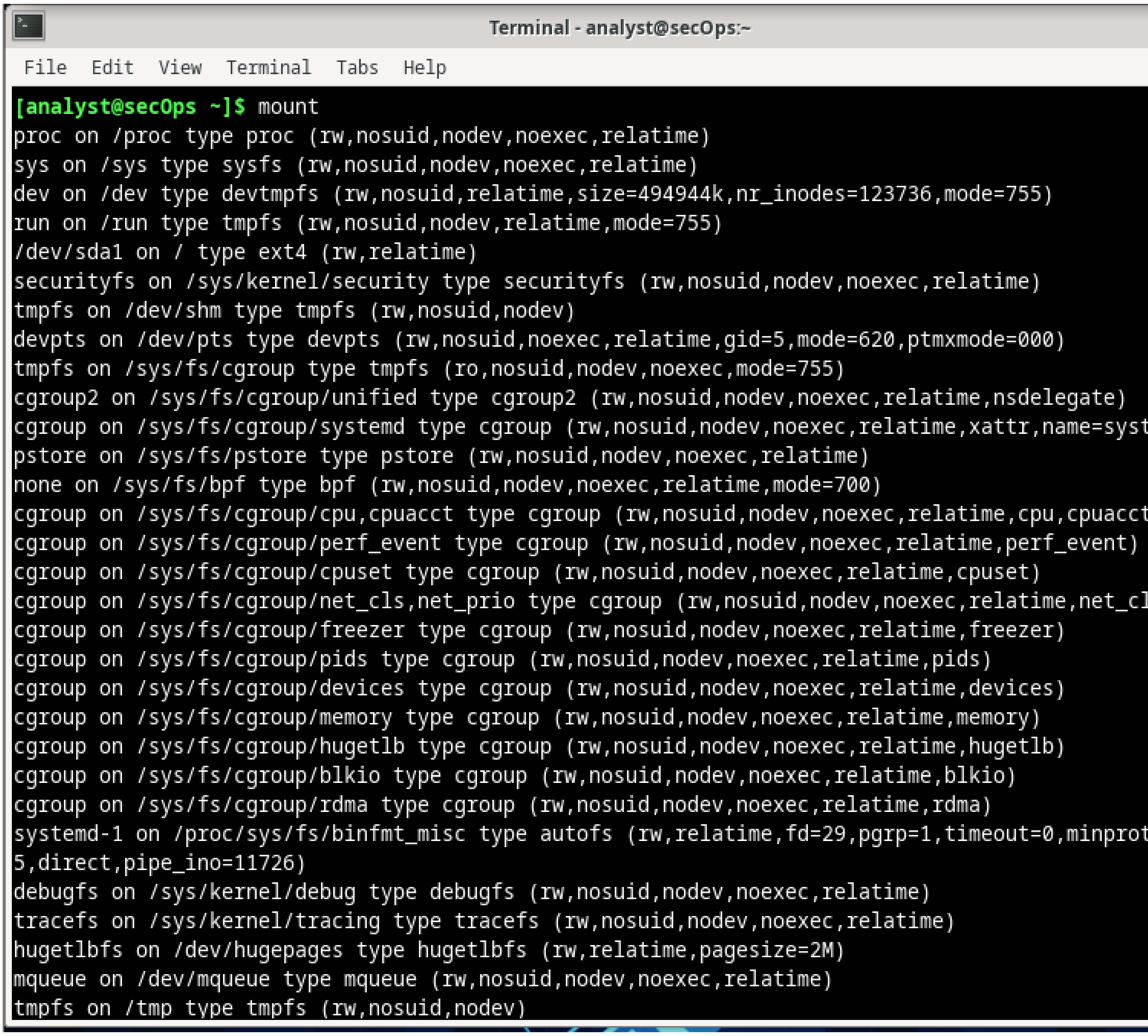
```
[analyst@secOps ~]$
```

Step 2 – a


Terminal - analyst@secOps:~
File Edit View Terminal Tabs Help

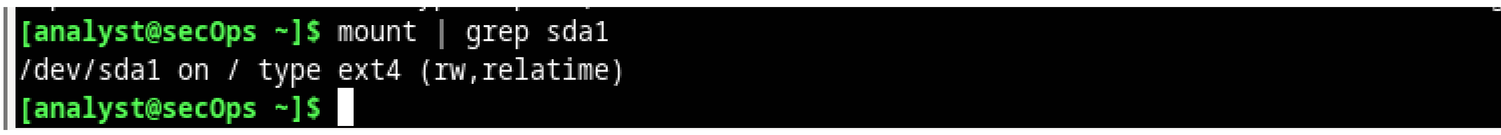
```
[analyst@secOps ~]$ lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda      8:0    0   10G  0 disk
└─sda1   8:1    0   10G  0 part /
sdb      8:16   0    1G  0 disk
└─sdb1   8:17   0 1023M  0 part
sr0     11:0    1 1024M  0 rom
```

Step 2 – b


Terminal - analyst@secOps:~
File Edit View Terminal Tabs Help

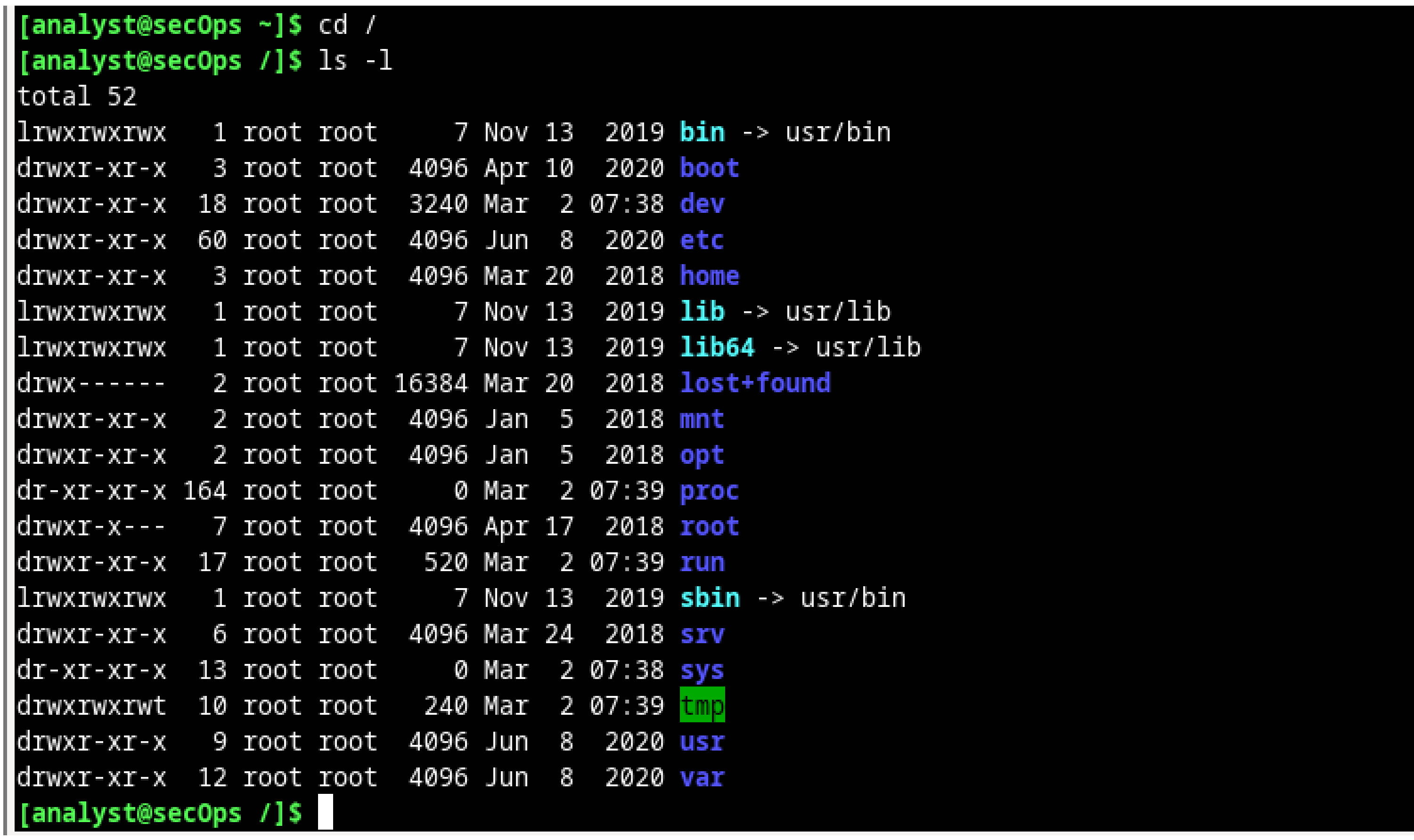
```
[analyst@secOps ~]$ mount
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
sys on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
dev on /dev type devtmpfs (rw,nosuid,relatime,size=494944k,nr_inodes=123736,mode=755)
run on /run type tmpfs (rw,nosuid,nodev,relatime,mode=755)
/dev/sda1 on / type ext4 (rw,relatime)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,mode=755)
cgroup2 on /sys/fs/cgroup/unified type cgroup2 (rw,nosuid,nodev,noexec,relatime,nsdelegate)
cgroup on /sys/fs/cgroup/systemd type cgroup (rw,nosuid,nodev,noexec,relatime,xattr,name=syst
pstore on /sys/fs/pstore type pstore (rw,nosuid,nodev,noexec,relatime)
none on /sys/fs/bpf type bpf (rw,nosuid,nodev,noexec,relatime,mode=700)
cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup (rw,nosuid,nodev,noexec,relatime,cpu,cpuacct
cgroup on /sys/fs/cgroup/perf_event type cgroup (rw,nosuid,nodev,noexec,relatime,perf_event)
cgroup on /sys/fs/cgroup/cpuset type cgroup (rw,nosuid,nodev,noexec,relatime,cpuset)
cgroup on /sys/fs/cgroup/net_cls,net_prio type cgroup (rw,nosuid,nodev,noexec,relatime,net_cl
cgroup on /sys/fs/cgroup/freezer type cgroup (rw,nosuid,nodev,noexec,relatime,freezer)
cgroup on /sys/fs/cgroup/pids type cgroup (rw,nosuid,nodev,noexec,relatime,pids)
cgroup on /sys/fs/cgroup/devices type cgroup (rw,nosuid,nodev,noexec,relatime,devices)
cgroup on /sys/fs/cgroup/memory type cgroup (rw,nosuid,nodev,noexec,relatime,memory)
cgroup on /sys/fs/cgroup/hugetlb type cgroup (rw,nosuid,nodev,noexec,relatime,hugetlb)
cgroup on /sys/fs/cgroup/blkio type cgroup (rw,nosuid,nodev,noexec,relatime,blkio)
cgroup on /sys/fs/cgroup/rdma type cgroup (rw,nosuid,nodev,noexec,relatime,rdma)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=29,pgrp=1,timeout=0,minprot
5,direct,pipe_ino=11726)
debugfs on /sys/kernel/debug type debugfs (rw,nosuid,nodev,noexec,relatime)
tracefs on /sys/kernel/tracing type tracefs (rw,nosuid,nodev,noexec,relatime)
hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime,pagesize=2M)
mqueue on /dev/mqueue type mqueue (rw,nosuid,nodev,noexec,relatime)
tmpfs on /tmp type tmpfs (rw,nosuid,nodev)
```

Step 2 – c



```
[analyst@secOps ~]$ mount | grep sda1
/dev/sda1 on / type ext4 (rw,relatime)
[analyst@secOps ~]$
```

Step 2 – d

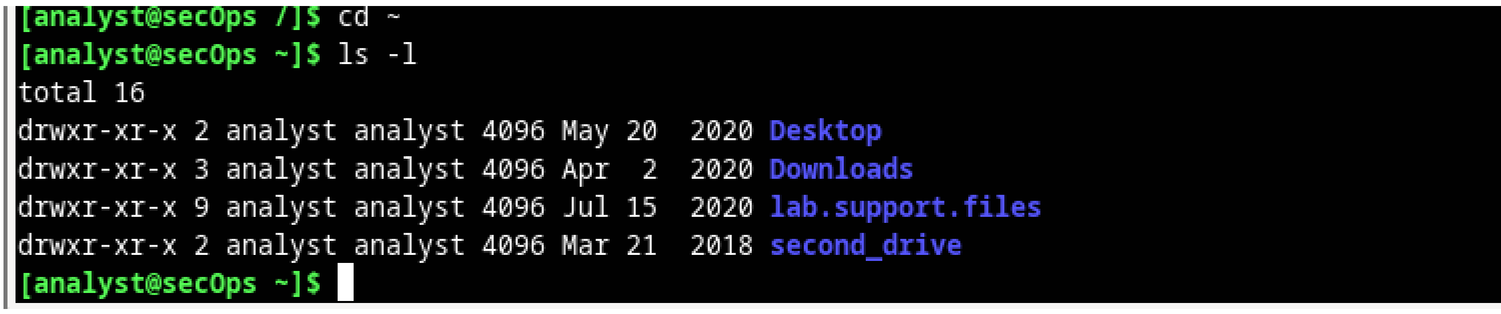


```
[analyst@secOps ~]$ cd /
[analyst@secOps /]$ ls -l
total 52
lrwxrwxrwx   1 root root     7 Nov 13  2019 bin -> usr/bin
drwxr-xr-x   3 root root  4096 Apr 10  2020 boot
drwxr-xr-x  18 root root  3240 Mar  2 07:38 dev
drwxr-xr-x  60 root root  4096 Jun  8  2020 etc
drwxr-xr-x   3 root root  4096 Mar 20  2018 home
lrwxrwxrwx   1 root root     7 Nov 13  2019 lib -> usr/lib
lrwxrwxrwx   1 root root     7 Nov 13  2019 lib64 -> usr/lib
drwx------   2 root root 16384 Mar 20  2018 lost+found
drwxr-xr-x   2 root root  4096 Jan  5  2018 mnt
drwxr-xr-x   2 root root  4096 Jan  5  2018 opt
dr-xr-xr-x 164 root root     0 Mar  2 07:39 proc
drwxr-x---   7 root root  4096 Apr 17  2018 root
drwxr-xr-x  17 root root   520 Mar  2 07:39 run
lrwxrwxrwx   1 root root     7 Nov 13  2019 sbin -> usr/bin
drwxr-xr-x   6 root root  4096 Mar 24  2018 srv
dr-xr-xr-x  13 root root     0 Mar  2 07:38 sys
drwxrwxrwt  10 root root   240 Mar  2 07:39 tmp
drwxr-xr-x   9 root root  4096 Jun  8  2020 usr
drwxr-xr-x  12 root root  4096 Jun  8  2020 var
[analyst@secOps /]$
```

What is the meaning of the output? Where are the listed files physically stored?
The first command changes the directory to the root directory. The root directory is the highest level of the filesystems. Because /dev/sda1 is mounted on the root directory ("/"), by listing the files in the root directory, the user is actually listing files physically stored in the root of the /dev/sda1 filesystem.

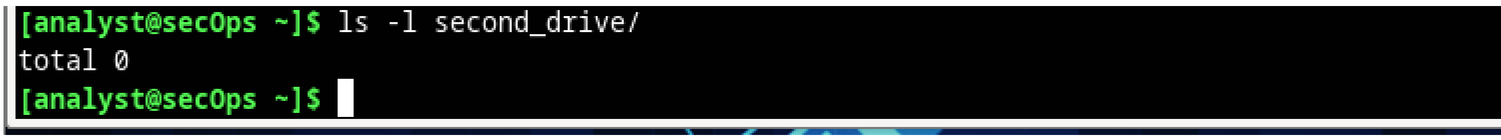
Why is /dev/sdb1 not shown in the output above?
Because /dev/sdb1 is not currently mounted.

Step 3 – a



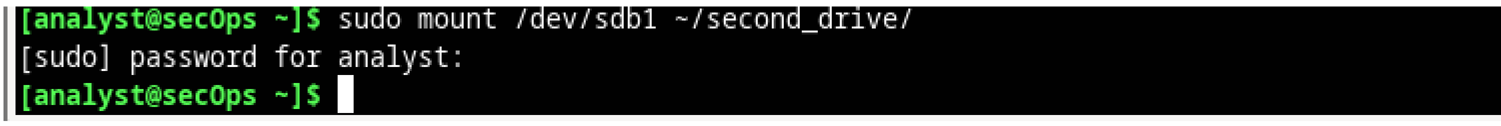
```
[analyst@secOps /]$ cd ~
[analyst@secOps ~]$ ls -l
total 16
drwxr-xr-x 2 analyst analyst 4096 May 20  2020 Desktop
drwxr-xr-x 3 analyst analyst 4096 Apr  2  2020 Downloads
drwxr-xr-x 9 analyst analyst 4096 Jul 15  2020 lab.support.files
drwxr-xr-x 2 analyst analyst 4096 Mar 21  2018 second_drive
[analyst@secOps ~]$
```

Step 3 – b



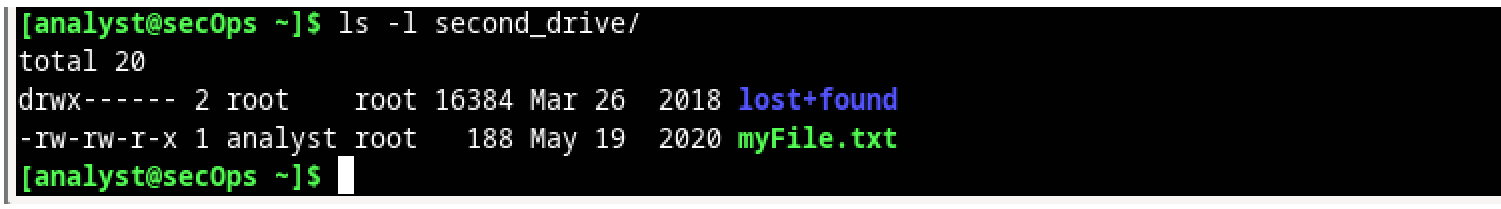
```
[analyst@secOps ~]$ ls -l second_drive/
total 0
[analyst@secOps ~]$
```

Step 3 – c



```
[analyst@secOps ~]$ sudo mount /dev/sdb1 ~/second_drive/
[sudo] password for analyst:
[analyst@secOps ~]$
```

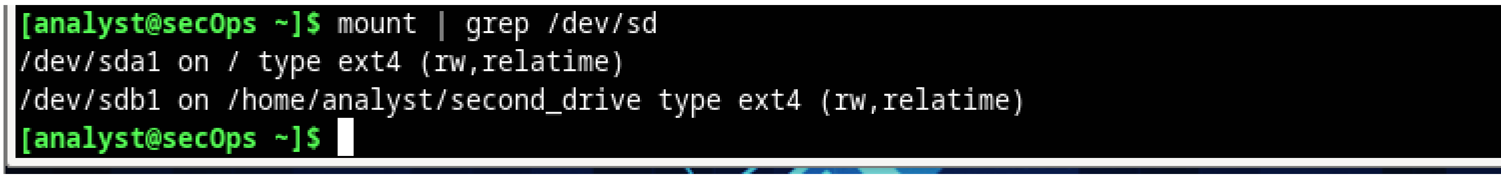
Step 3 – d



```
[analyst@secOps ~]$ ls -l second_drive/
total 20
drwx------ 2 root    root 16384 Mar 26  2018 lost+found
-rw-rw-r-x 1 analyst root   188 May 19  2020 myFile.txt
[analyst@secOps ~]$
```

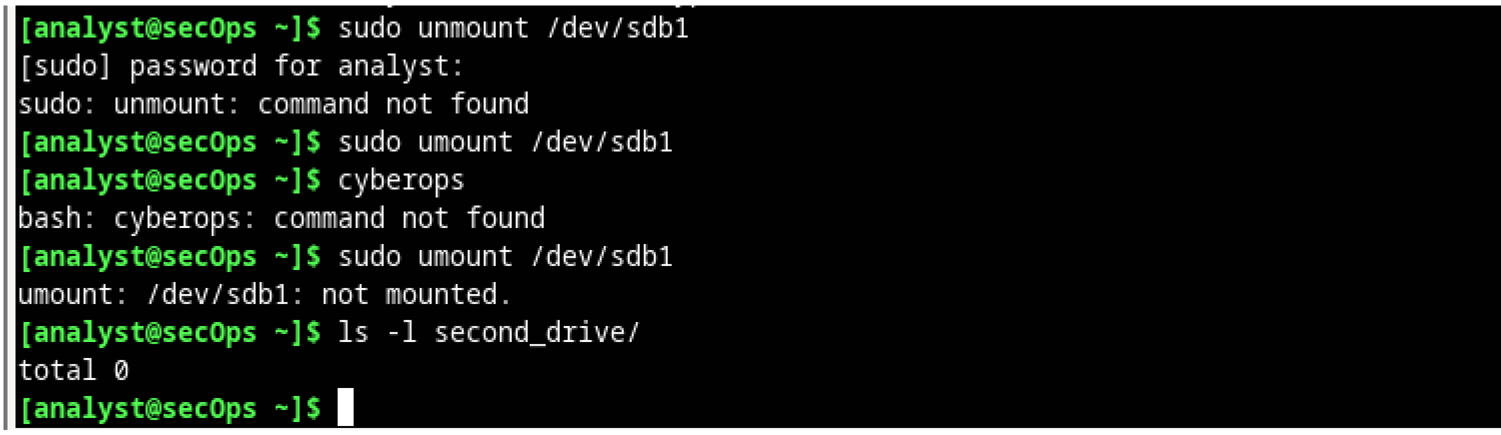
Why is the directory no longer empty? Where are the listed files physically stored?
After the mount, /home/analyst/second_drive becomes the entry point to the filesystem physically stored in /dev/sdb1.

Step 3 – e



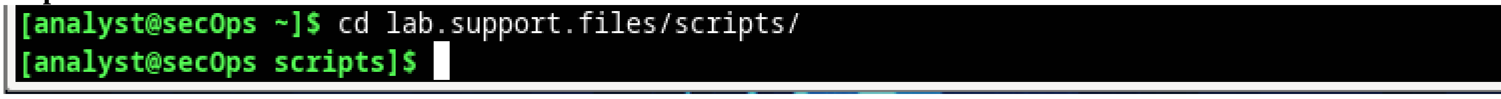
```
[analyst@secOps ~]$ mount | grep /dev/sd
/dev/sda1 on / type ext4 (rw,relatime)
/dev/sdb1 on /home/analyst/second_drive type ext4 (rw,relatime)
[analyst@secOps ~]$
```

Step 3 – f



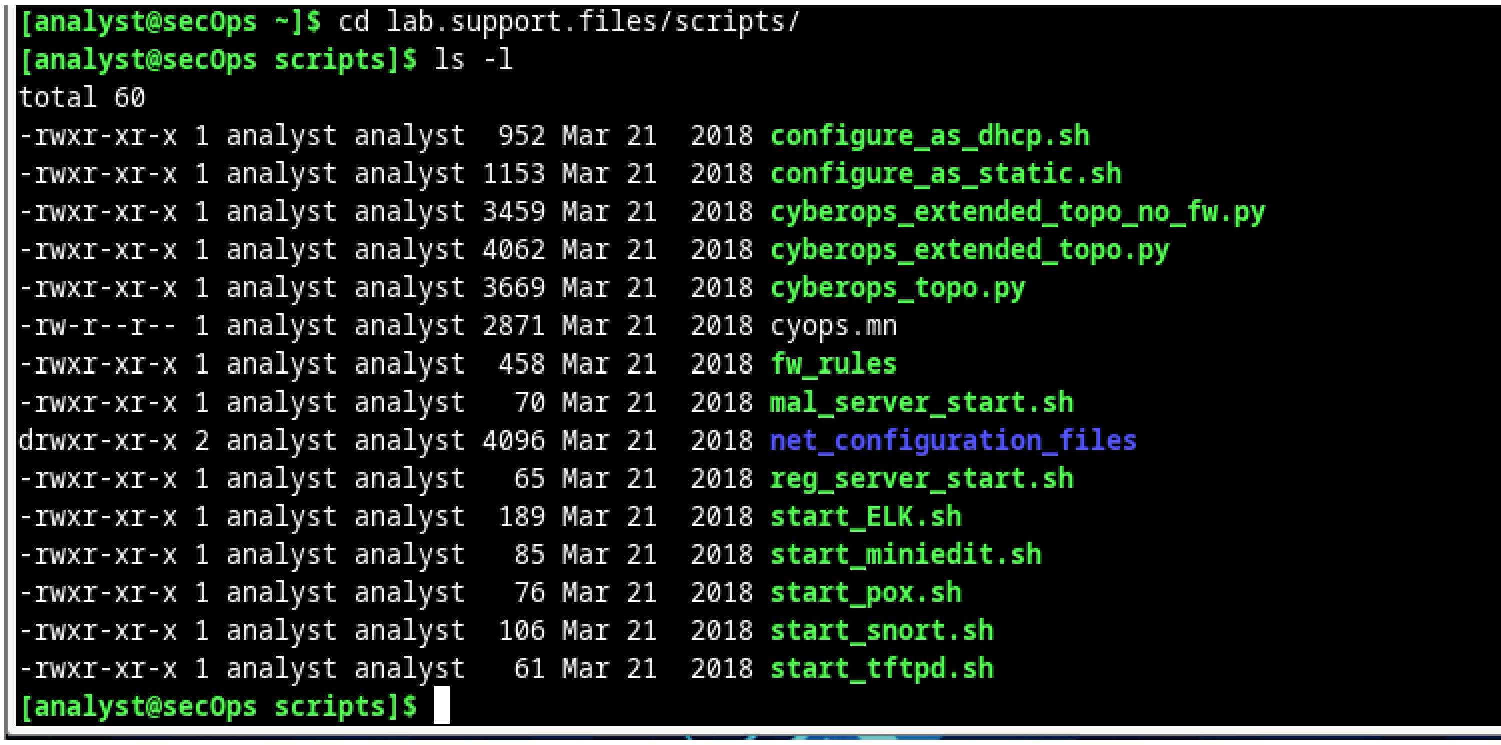
```
[analyst@secOps ~]$ sudo unmount /dev/sdb1
[sudo] password for analyst:
sudo: unmount: command not found
[analyst@secOps ~]$ sudo umount /dev/sdb1
[analyst@secOps ~]$ cyberops
bash: cyberops: command not found
[analyst@secOps ~]$ sudo umount /dev/sdb1
umount: /dev/sdb1: not mounted.
[analyst@secOps ~]$ ls -l second_drive/
total 0
[analyst@secOps ~]$
```

Part 2
Step 1 – a



```
[analyst@secOps ~]$ cd lab.support.files/scripts/
[analyst@secOps scripts]$
```

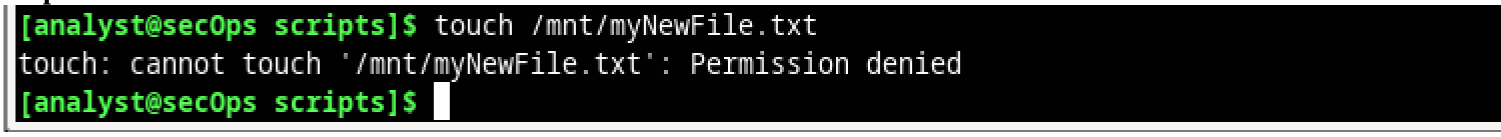
Step 1 – b



```
[analyst@secOps ~]$ cd lab.support.files/scripts/
[analyst@secOps scripts]$ ls -l
total 60
-rwxr-xr-x 1 analyst analyst  952 Mar 21  2018 configure_as_dhcp.sh
-rwxr-xr-x 1 analyst analyst 1153 Mar 21  2018 configure_as_static.sh
-rwxr-xr-x 1 analyst analyst 3459 Mar 21  2018 cyberops_extended_topo_no_fw.py
-rwxr-xr-x 1 analyst analyst 4062 Mar 21  2018 cyberops_extended_topo.py
-rwxr-xr-x 1 analyst analyst 3669 Mar 21  2018 cyberops_topo.py
-rw-r--r-- 1 analyst analyst 2871 Mar 21  2018 cyops.mn
-rwxr-xr-x 1 analyst analyst  458 Mar 21  2018 fw_rules
-rwxr-xr-x 1 analyst analyst   70 Mar 21  2018 mal_server_start.sh
drwxr-xr-x 2 analyst analyst 4096 Mar 21  2018 net_configuration_files
-rwxr-xr-x 1 analyst analyst   65 Mar 21  2018 reg_server_start.sh
-rwxr-xr-x 1 analyst analyst  189 Mar 21  2018 start_ELK.sh
-rwxr-xr-x 1 analyst analyst   85 Mar 21  2018 start_miniedit.sh
-rwxr-xr-x 1 analyst analyst   76 Mar 21  2018 start_pox.sh
-rwxr-xr-x 1 analyst analyst  106 Mar 21  2018 start_snort.sh
-rwxr-xr-x 1 analyst analyst   61 Mar 21  2018 start_tftpd.sh
[analyst@secOps scripts]$
```

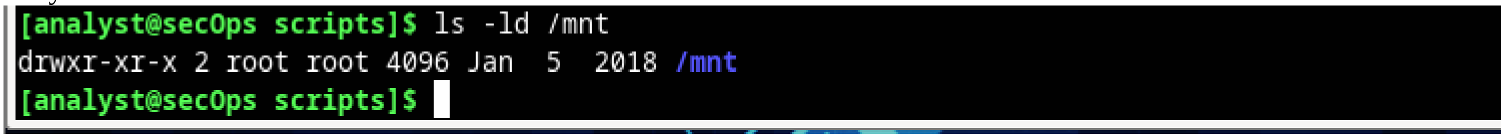
Consider the cyops.mn file as an example. Who is the owner of the file? How about the group?
Owner: analyst; Group: analyst
The permissions for cyops.mn are –rw-r--r--. What does that mean?
The analyst user who owns the file can read and write to it but cannot execute it (-rw). Other than the owner, members of the analyst group are only permitted to read files (-r-); no executing or writing is permitted. No other users are permitted to modify or run that file.

Step 1 – c



```
[analyst@secOps scripts]$ touch /mnt/myNewFile.txt
touch: cannot touch '/mnt/myNewFile.txt': Permission denied
[analyst@secOps scripts]$
```

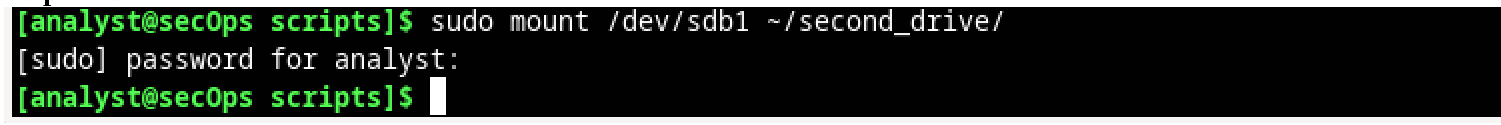
Why was the file not created? List the permissions, ownership and content of the /mnt directory and explain what happened. With the addition of -d option, it lists the permission of the parent directory. Record the answer in the lines below.
The root user owns the /mnt directory and has the rights drwxr-xr-x. In this manner, the /mnt folder can only be written to by the root user.



```
[analyst@secOps scripts]$ ls -ld /mnt
drwxr-xr-x 2 root root 4096 Jan  5  2018 /mnt
[analyst@secOps scripts]$
```

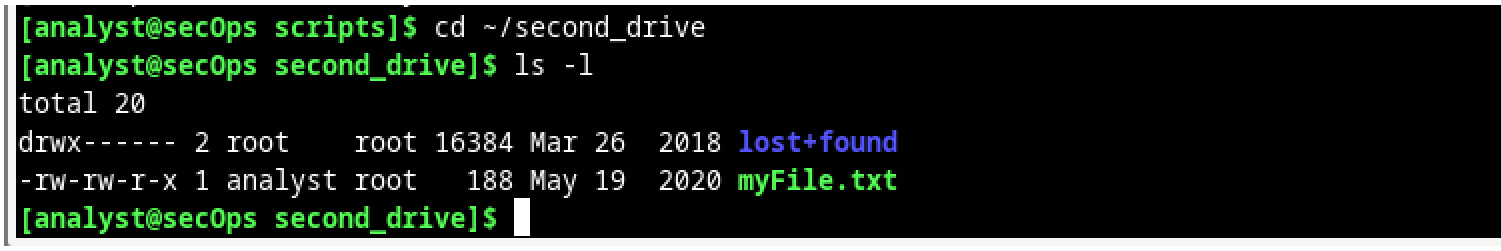
What can be done for the touch command shown above to be successful?
The /mnt directory's permissions can be changed, or the command can be run as root by appending sudo before it.

Step 1 – d



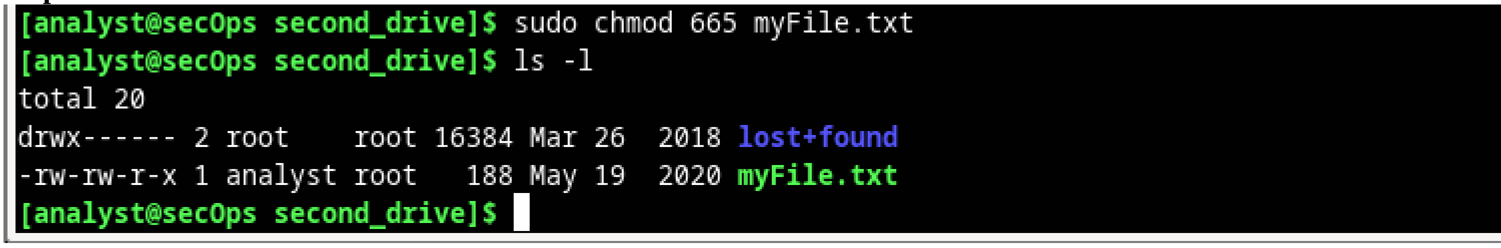
```
[analyst@secOps scripts]$ sudo mount /dev/sdb1 ~/second_drive/
[sudo] password for analyst:
[analyst@secOps scripts]$
```

Step 1 – e



```
[analyst@secOps scripts]$ cd ~/second_drive
[analyst@secOps second_drive]$ ls -l
total 20
drwx------ 2 root    root 16384 Mar 26  2018 lost+found
-rw-rw-r-x 1 analyst root   188 May 19  2020 myFile.txt
[analyst@secOps second_drive]$ 
```

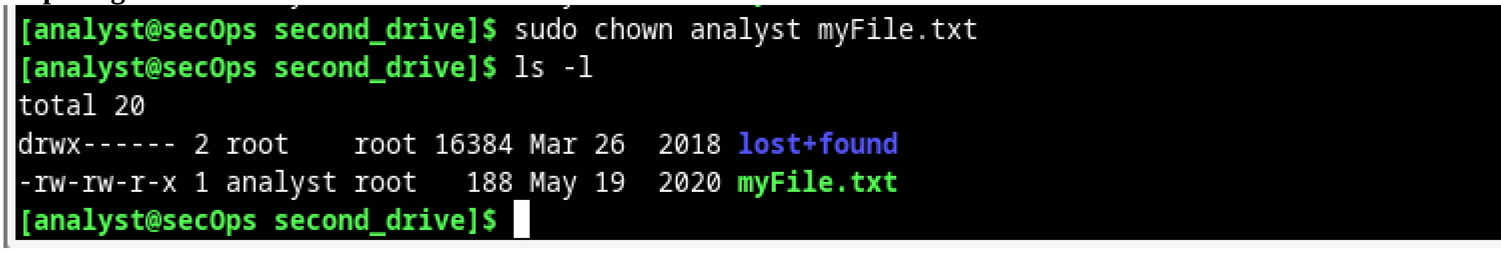
What are the permissions of the myFile.txt file?
The permission is -rw-r–r–.
Step 1 – f



```
[analyst@secOps second_drive]$ sudo chmod 665 myFile.txt
[analyst@secOps second_drive]$ ls -l
total 20
drwx------ 2 root    root 16384 Mar 26  2018 lost+found
-rw-rw-r-x 1 analyst root   188 May 19  2020 myFile.txt
[analyst@secOps second_drive]$ 
```

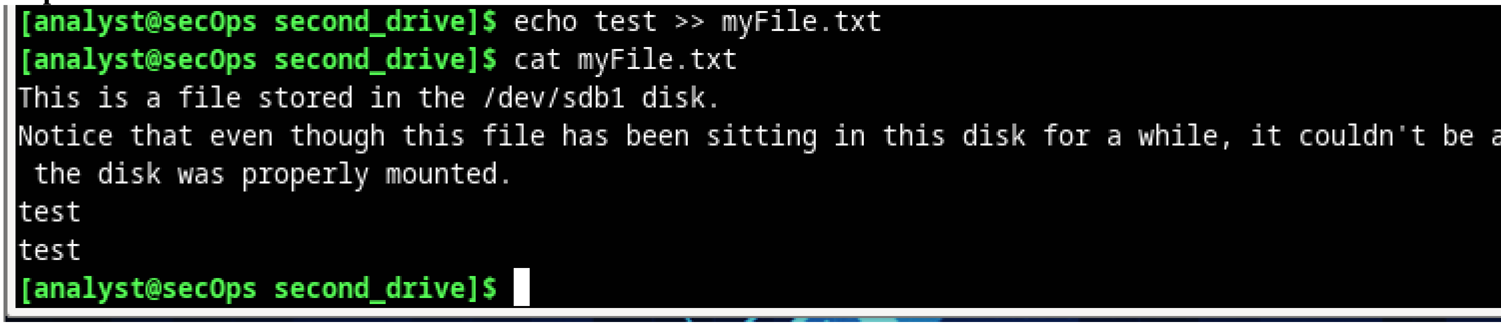
Did the permissions change? What are the permissions of myFile.txt?
Yes, it becomes -rw-rw-r-x.
What command would change the permissions of myFile.txt to rwxrwxrwx, granting any user in the system full access to the file?
The command will be sudo chmod 777 myFile.txt.

Step 1 – g



```
[analyst@secOps second_drive]$ sudo chown analyst myFile.txt
[analyst@secOps second_drive]$ ls -l
total 20
drwx------ 2 root    root 16384 Mar 26  2018 lost+found
-rw-rw-r-x 1 analyst root   188 May 19  2020 myFile.txt
[analyst@secOps second_drive]$ 
```

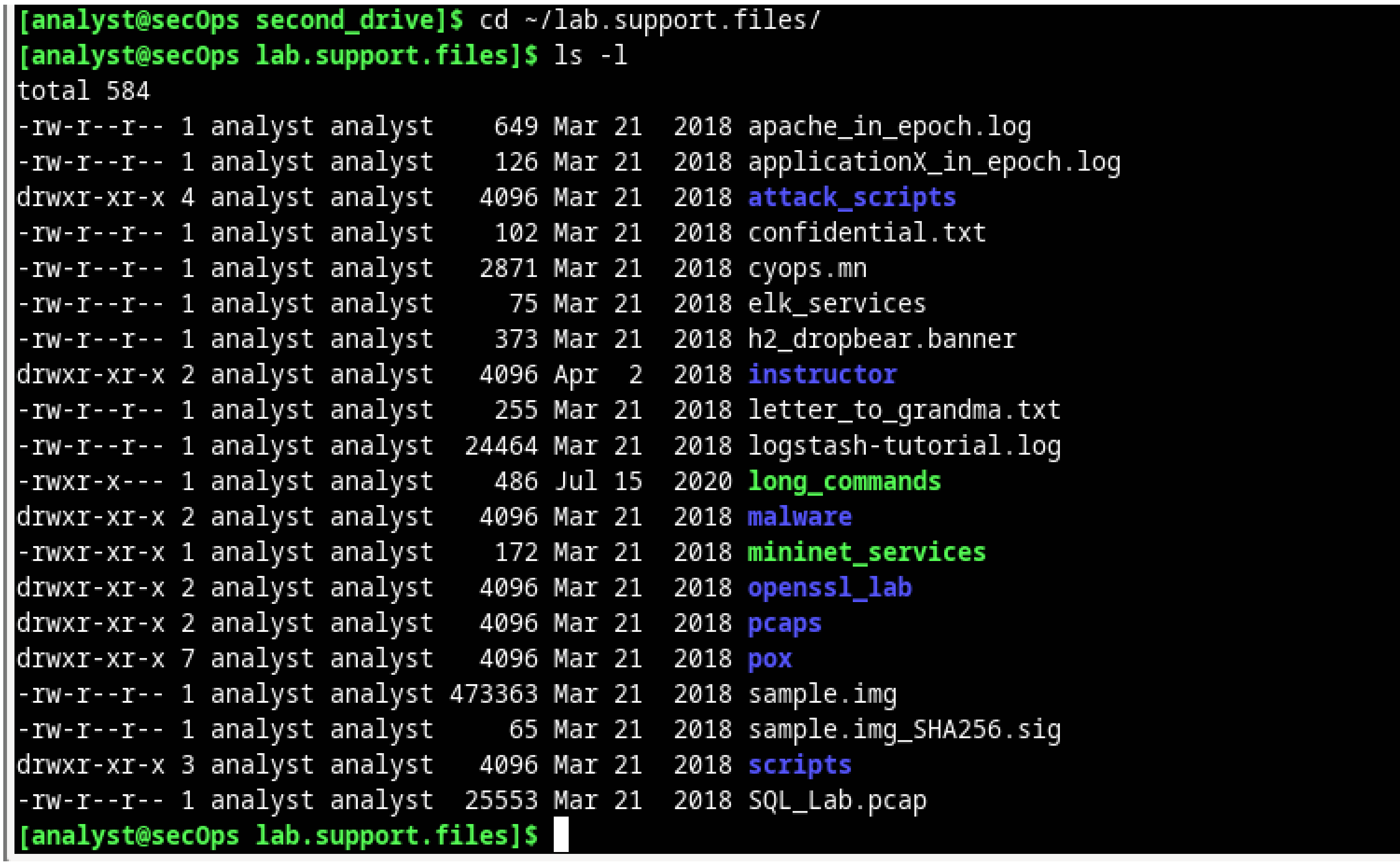
Step 1 – h



```
[analyst@secOps second_drive]$ echo test >> myFile.txt
[analyst@secOps second_drive]$ cat myFile.txt
This is a file stored in the /dev/sdb1 disk.
Notice that even though this file has been sitting in this disk for a while, it couldn't be a
 the disk was properly mounted.
test
test
[analyst@secOps second_drive]$ 
```

Was the operation successful? Explain.
Yes. The owner of the file is analyst, and the previous permissions of 665 are still in place. The owner and users belonging to the root group are able to edit the file due to the current permissions.

Step 2 – a

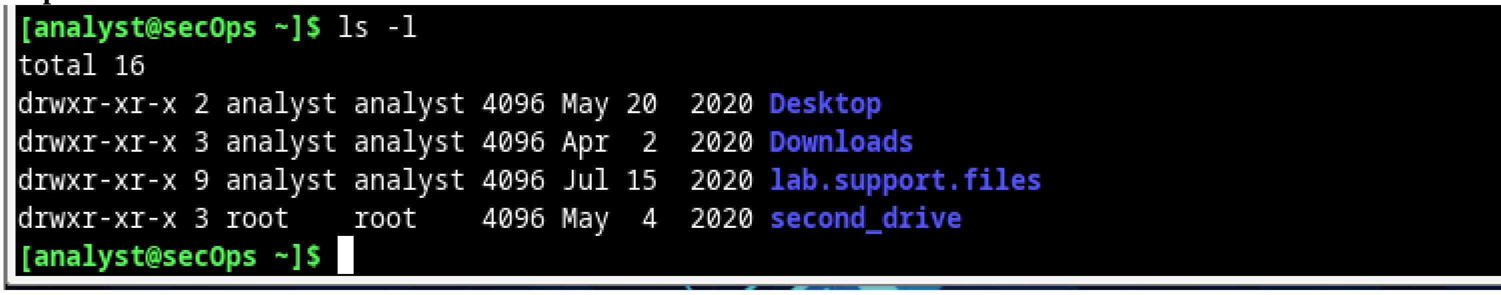


```
[analyst@secOps second_drive]$ cd ~/lab.support.files/
[analyst@secOps lab.support.files]$ ls -l
total 584
-rw-r--r-- 1 analyst analyst    649 Mar 21  2018 apache_in_epoch.log
-rw-r--r-- 1 analyst analyst    126 Mar 21  2018 applicationX_in_epoch.log
drwxr-xr-x 4 analyst analyst   4096 Mar 21  2018 attack_scripts
-rw-r--r-- 1 analyst analyst    102 Mar 21  2018 confidential.txt
-rw-r--r-- 1 analyst analyst   2871 Mar 21  2018 cyops.mn
-rw-r--r-- 1 analyst analyst     75 Mar 21  2018 elk_services
-rw-r--r-- 1 analyst analyst    373 Mar 21  2018 h2_dropbear.banner
drwxr-xr-x 2 analyst analyst   4096 Apr  2  2018 instructor
-rw-r--r-- 1 analyst analyst    255 Mar 21  2018 letter_to_grandma.txt
-rw-r--r-- 1 analyst analyst  24464 Mar 21  2018 logstash-tutorial.log
-rwxr-x--- 1 analyst analyst    486 Jul 15  2020 long_commands
drwxr-xr-x 2 analyst analyst   4096 Mar 21  2018 malware
-rwxr-xr-x 1 analyst analyst    172 Mar 21  2018 mininet_services
drwxr-xr-x 2 analyst analyst   4096 Mar 21  2018 openssl_lab
drwxr-xr-x 2 analyst analyst   4096 Mar 21  2018 pcaps
drwxr-xr-x 7 analyst analyst   4096 Mar 21  2018 pox
-rw-r--r-- 1 analyst analyst 473363 Mar 21  2018 sample.img
-rw-r--r-- 1 analyst analyst     65 Mar 21  2018 sample.img_SHA256.sig
drwxr-xr-x 3 analyst analyst   4096 Mar 21  2018 scripts
-rw-r--r-- 1 analyst analyst  25553 Mar 21  2018 SQL_Lab.pcap
[analyst@secOps lab.support.files]$
```

Compare the permissions of the malware directory with the mininet_services file. What is the difference between beginning part of the malware line and the mininet_services line?
There is a letter d at the beginning before the permissions for the malware directory.

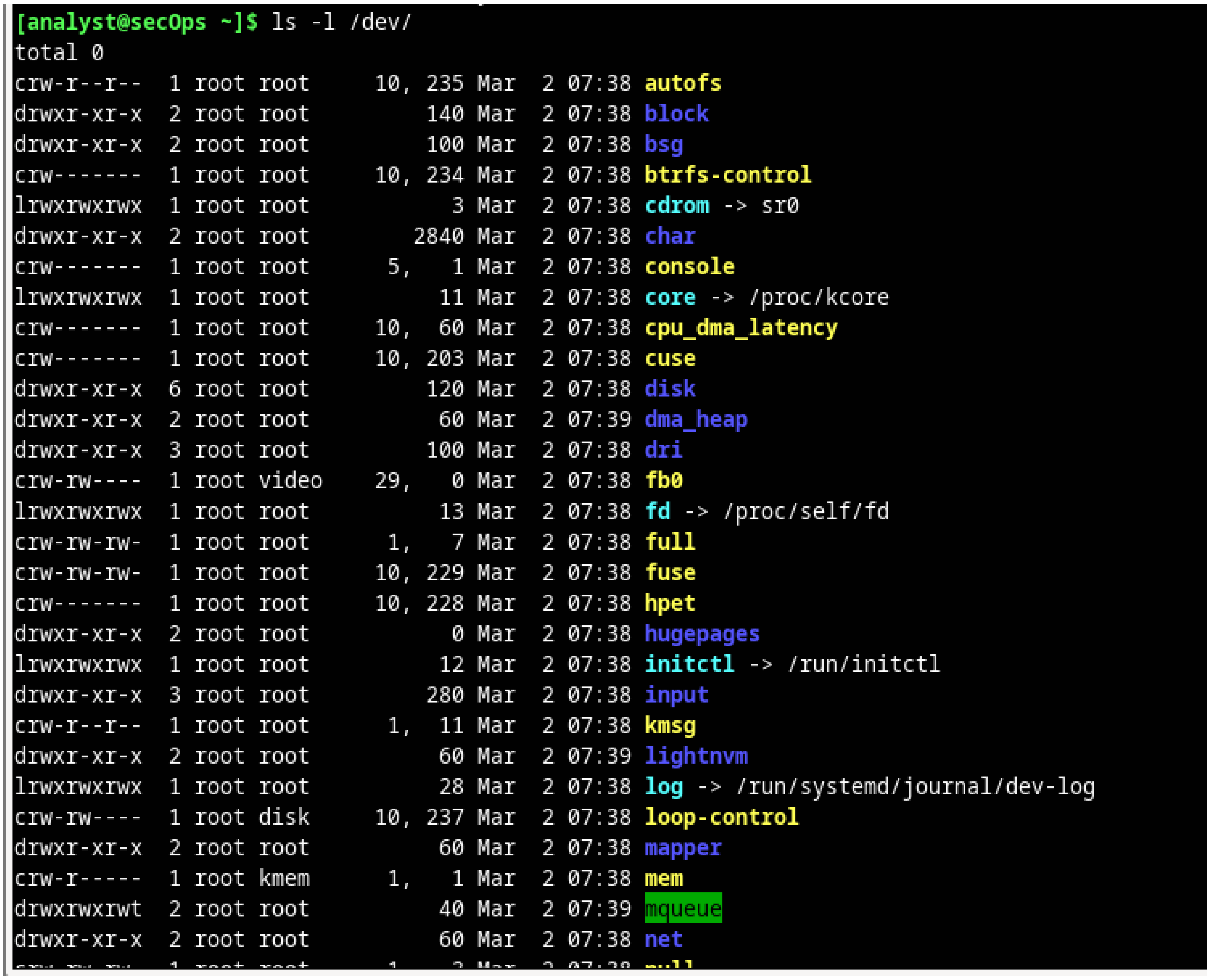
Part 3

Step 1 – a



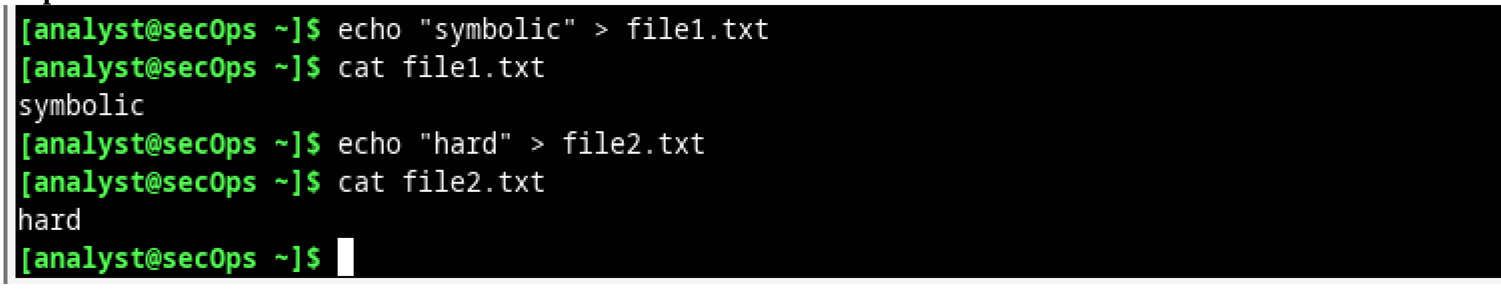
```
[analyst@secOps ~]$ ls -l
total 16
drwxr-xr-x 2 analyst analyst 4096 May 20  2020 Desktop
drwxr-xr-x 3 analyst analyst 4096 Apr  2  2020 Downloads
drwxr-xr-x 9 analyst analyst 4096 Jul 15  2020 lab.support.files
drwxr-xr-x 3 root    root    4096 May  4  2020 second_drive
[analyst@secOps ~]$
```

Step 1 – b



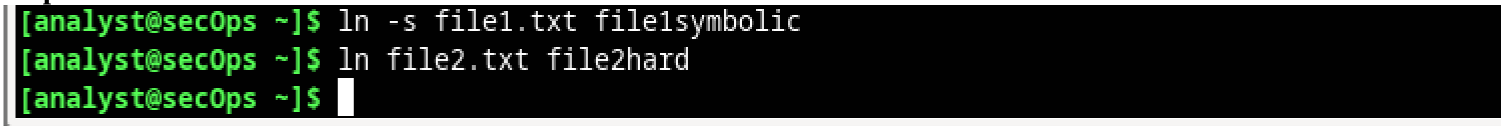
```
[analyst@secOps ~]$ ls -l /dev/
total 0
crw-r--r--  1 root root     10, 235 Mar  2 07:38 autofs
drwxr-xr-x  2 root root         140 Mar  2 07:38 block
drwxr-xr-x  2 root root         100 Mar  2 07:38 bsg
crw-------  1 root root     10, 234 Mar  2 07:38 btrfs-control
lrwxrwxrwx  1 root root           3 Mar  2 07:38 cdrom -> sr0
drwxr-xr-x  2 root root        2840 Mar  2 07:38 char
crw-------  1 root root      5,   1 Mar  2 07:38 console
lrwxrwxrwx  1 root root          11 Mar  2 07:38 core -> /proc/kcore
crw-------  1 root root     10,  60 Mar  2 07:38 cpu_dma_latency
crw-------  1 root root     10, 203 Mar  2 07:38 cuse
drwxr-xr-x  6 root root         120 Mar  2 07:38 disk
drwxr-xr-x  2 root root          60 Mar  2 07:39 dma_heap
drwxr-xr-x  3 root root         100 Mar  2 07:38 dri
crw-rw----  1 root video    29,   0 Mar  2 07:38 fb0
lrwxrwxrwx  1 root root          13 Mar  2 07:38 fd -> /proc/self/fd
crw-rw-rw-  1 root root      1,   7 Mar  2 07:38 full
crw-rw-rw-  1 root root     10, 229 Mar  2 07:38 fuse
crw-------  1 root root     10, 228 Mar  2 07:38 hpet
drwxr-xr-x  2 root root           0 Mar  2 07:38 hugepages
lrwxrwxrwx  1 root root          12 Mar  2 07:38 initctl -> /run/initctl
drwxr-xr-x  3 root root         280 Mar  2 07:38 input
crw-r--r--  1 root root      1,  11 Mar  2 07:38 kmsg
drwxr-xr-x  2 root root          60 Mar  2 07:39 lightnvm
lrwxrwxrwx  1 root root          28 Mar  2 07:38 log -> /run/systemd/journal/dev-log
crw-rw----  1 root disk     10, 237 Mar  2 07:38 loop-control
drwxr-xr-x  2 root root          60 Mar  2 07:38 mapper
crw-r-----  1 root kmem      1,   1 Mar  2 07:38 mem
drwxrwxrwt  2 root root          40 Mar  2 07:39 mqueue
drwxr-xr-x  2 root root          60 Mar  2 07:38 net
```

Step 1 – c



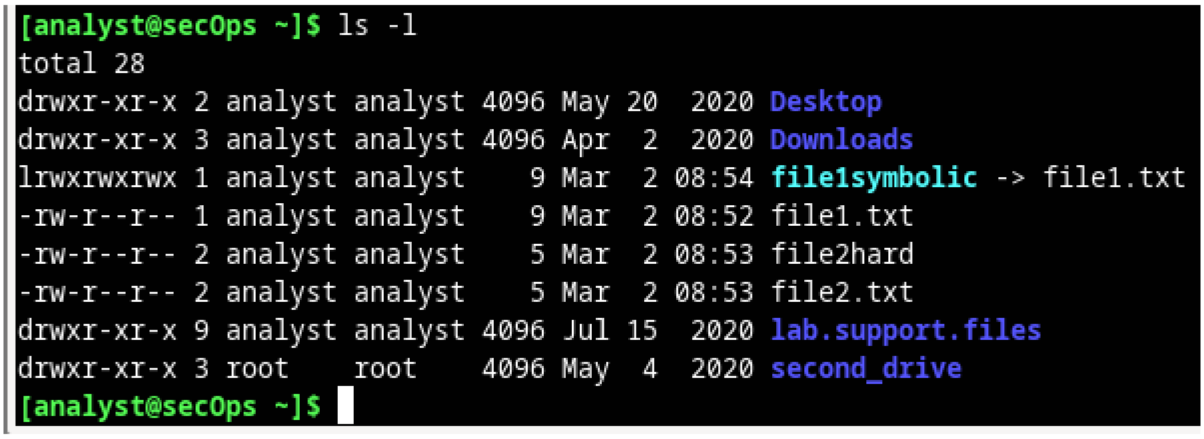
```
[analyst@secOps ~]$ echo "symbolic" > file1.txt
[analyst@secOps ~]$ cat file1.txt
symbolic
[analyst@secOps ~]$ echo "hard" > file2.txt
[analyst@secOps ~]$ cat file2.txt
hard
[analyst@secOps ~]$ 
```

Step 1 – d



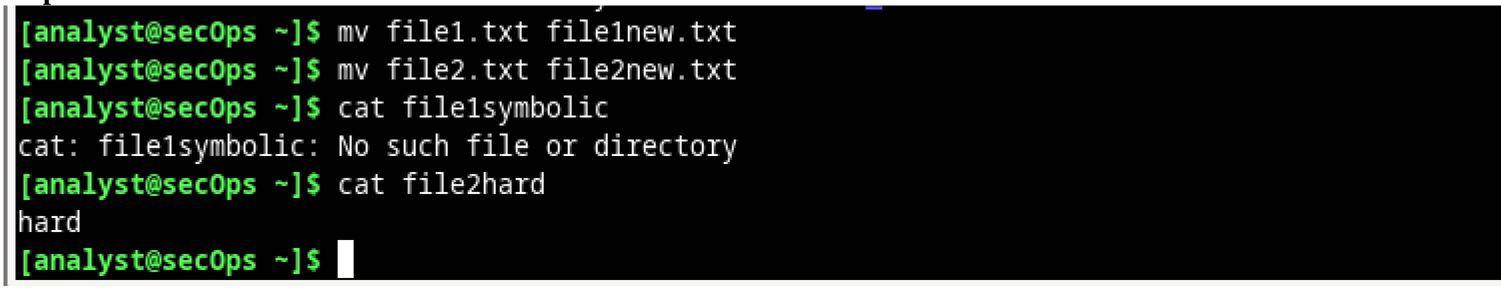
```
[analyst@secOps ~]$ ln -s file1.txt file1symbolic
[analyst@secOps ~]$ ln file2.txt file2hard
[analyst@secOps ~]$ 
```

Step 1 – e



```
[analyst@secOps ~]$ ls -l
total 28
drwxr-xr-x 2 analyst analyst 4096 May 20  2020 Desktop
drwxr-xr-x 3 analyst analyst 4096 Apr  2  2020 Downloads
lrwxrwxrwx 1 analyst analyst    9 Mar  2 08:54 file1symbolic -> file1.txt
-rw-r--r-- 1 analyst analyst    9 Mar  2 08:52 file1.txt
-rw-r--r-- 2 analyst analyst    5 Mar  2 08:53 file2hard
-rw-r--r-- 2 analyst analyst    5 Mar  2 08:53 file2.txt
drwxr-xr-x 9 analyst analyst 4096 Jul 15  2020 lab.support.files
drwxr-xr-x 3 root    root    4096 May  4  2020 second_drive
[analyst@secOps ~]$ 
```

Step 1 – f



```
[analyst@secOps ~]$ mv file1.txt file1new.txt
[analyst@secOps ~]$ mv file2.txt file2new.txt
[analyst@secOps ~]$ cat file1symbolic
cat: file1symbolic: No such file or directory
[analyst@secOps ~]$ cat file2hard
hard
[analyst@secOps ~]$ 
```

What do you think would happen to file2hard if you opened a text editor and changed the text in file2new.txt?
Because they both point to the same inode on the hard disk drive, changing the contents of one file would also modify the contents of the other.

Reflection
File permissions and ownership are two of the most important aspects of Linux. They are also a common cause of problems. A file that has the wrong permissions or ownership set will not be available to the programs that need to access it. In this scenario, the program will usually break, and errors will be encountered.