CS628 Assignment 1 – Principles of Least Privilage
Souvik Mukherjee
Roll Number: 231110405

1. Function categorization as Privilidged or Unprivilidged

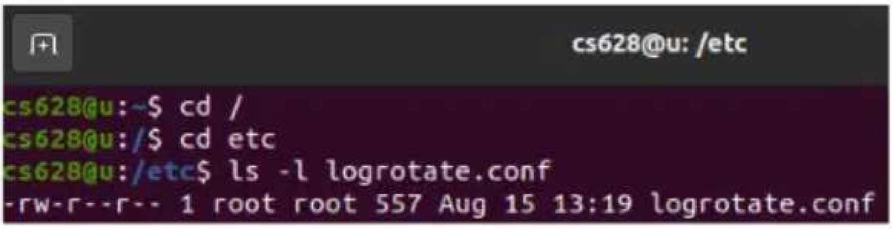
Function	Needs Privilidge	Do not need privilidge	Which command needs privilidge
functionA()	Yes	No	fopen(fp, "a");
functionB()	No	Yes	N/A
functionC()	Yes	No	system(command);
functionD()	Yes	No	fopen(filename, "w");
functionE()	No	Yes	N/A
functionF()	No	Yes	N/A
functionG()	Yes	No	fopen("/etc/crontab", "a");
functionH()	No	Yes	N/A
functionI()	No	Yes	N/A
functionJ()	No	Yes	N/A

2. Explanation, Reasoning & Snippets for the Functions

2.1. functionA()

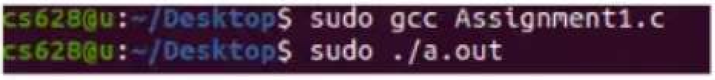
2.1.1. Explanation, Reasoning & Snippets

The functionA() opens the "logrotate.conf" from "/etc" and append the string "Rotate = 90" in the file. But, the file is owned by root and permission granted to others is "only read". So we would need root privilidge to write on the file. To execute the file, we need to set the set-UID=1 or run with super user priviledges.

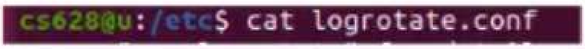


```
cs628@u:~$ cd /
cs628@u:/$ cd etc
cs628@u:/etc$ ls -l logrotate.conf
-rw-r--r-- 1 root root 557 Aug 15 13:19 logrotate.conf
```

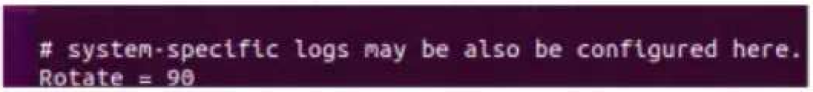
>> cs628@u:/etc$ sudo chmod 4644 logrotate.conf (set uid=1 and compile normally) or run as super user do (sudo)



```
cs628@u:~/Desktop$ sudo gcc Assignment1.c
cs628@u:~/Desktop$ sudo ./a.out
```



```
cs628@u:/etc$ cat logrotate.conf
```



```
  # system-specific logs may be also be configured here.
  Rotate = 90
```

[[We can see the string "Rotate = 90" is appended to the file "logrotate.conf"]]

2.2. functionB()

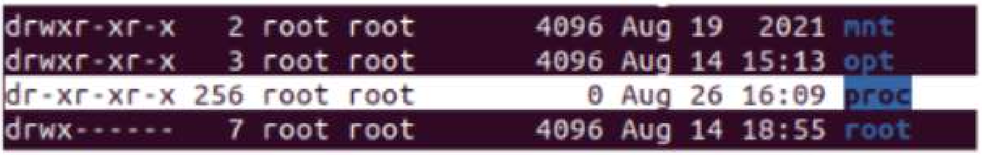
2.2.1. Explanation, Reasoning & Snippets

We are opening the directory "/proc" and we run a while loop untill we reach to the end of entries, in case the entry name (file/folder) is not a number we continue to next iteration, if it's a number, we print it out.

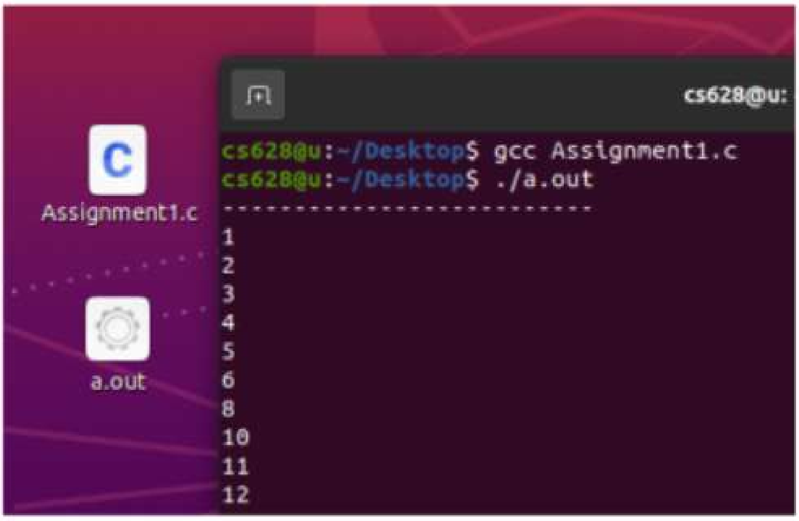
The "/proc" is a directory with user and group as root, and other user (cs628) can execute the directory.

(Read, Write & Execute in directory) Read in directory is reading all contents of directory (ls of directory). Write is user can create, append, modify, delete files. (touch, rm, fopen, fprintf, etc) Execute means we can do "$cd /proc" and enter the directory, and this is what we need.

So as other user, we do not need root permission for 'functionB'



```
drwxr-xr-x   2 root root        4096 Aug 19  2021 mnt
drwxr-xr-x   3 root root        4096 Aug 14 15:13 opt
dr-xr-xr-x 256 root root           0 Aug 26 16:09 proc
drwx------   7 root root        4096 Aug 14 18:55 root
```



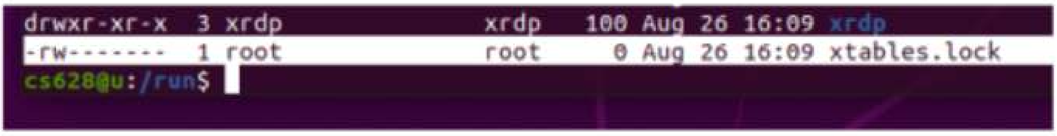
```
cs628@u:~/Desktop$ gcc Assignment1.c
cs628@u:~/Desktop$ ./a.out
------------------------
1
2
3
4
5
6
8
10
11
12
```

2.3. functionC()

2.3.1. Explanation, Reasoning & Snippets

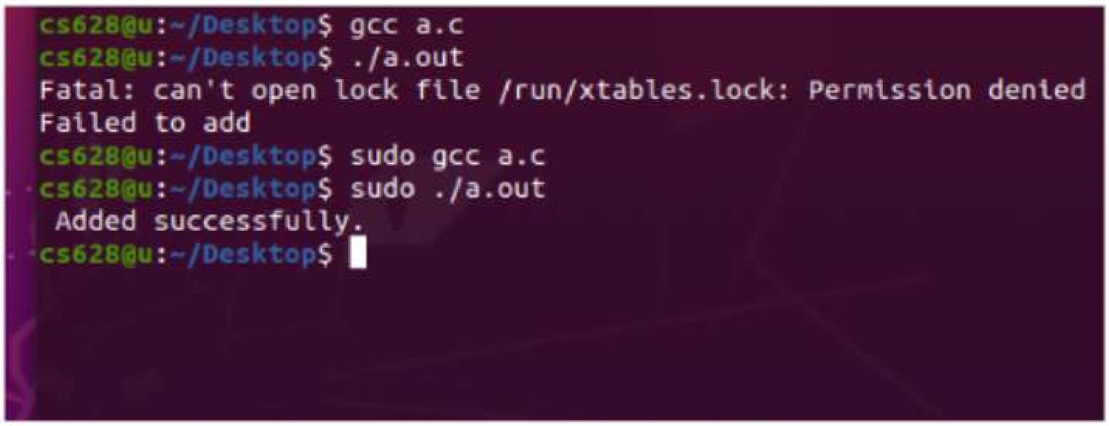
We are iterating the linklist of list of network interfaces using getifaddrs and attempting to add a rule to iptable firewall. "_-A OUTPUT -j DROP" says block all outgoing packets with a certain address, trying to access the internet, using firewall.
The "/run/xtables" is preventing us to do so. We do not have access to it as others.
We can see after a sudo command we are able to do so. Hence we need privilidges.

I have replaced the "google.com" to "172.29.233.235" in sprintf, to execute the file, as our VM do not have an internet access.



```
drwxr-xr-x  3 xrdp              xrdp   100 Aug 26 16:09 xrdp
-rw-------  1 root              root     0 Aug 26 16:09 xtables.lock
cs628@u:/run$ 
```

I have copied the function C() to a file named "a.c" and executed it from there.



```
cs628@u:~/Desktop$ gcc a.c
cs628@u:~/Desktop$ ./a.out
Fatal: can't open lock file /run/xtables.lock: Permission denied
Failed to add
cs628@u:~/Desktop$ sudo gcc a.c
cs628@u:~/Desktop$ sudo ./a.out
 Added successfully.
cs628@u:~/Desktop$ 
```

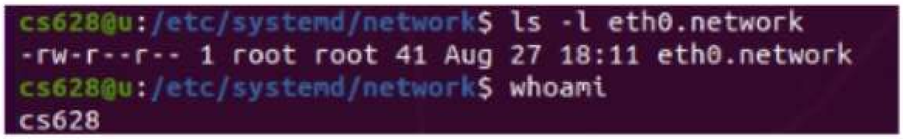
2.4. functionD()

2.4.1. Explanation, Reasoning & Snippets

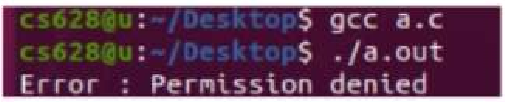
We are storing the directory path "/etc/systemd/network/eth0.network", in the character array "filename".

"eth0" is passed as a string, during the function call in the main function. Sprintf concates the passed value of "eth0" in the string "/etc/systemd/network/%s.network", and places it in the filename char array. Fopen opens the file with write permission and fprintf stores a string in the file.

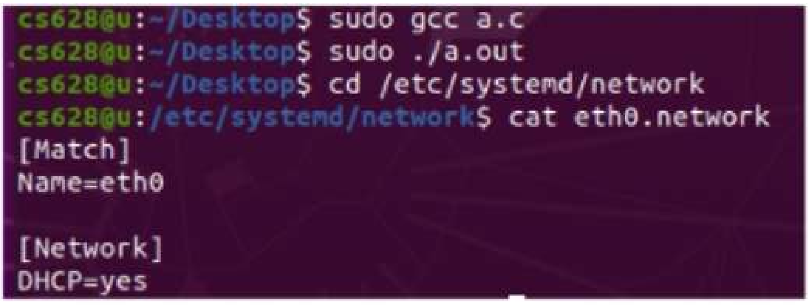
Since the file "eth0.network" is owned by root, as we cs628 fall under others, we only have read permission. So, we do need root privilidges to execute the functionD()



```
cs628@u:/etc/systemd/network$ ls -l eth0.network
-rw-r--r-- 1 root root 41 Aug 27 18:11 eth0.network
cs628@u:/etc/systemd/network$ whoami
cs628
```



```
cs628@u:~/Desktop$ gcc a.c
cs628@u:~/Desktop$ ./a.out
Error : Permission denied
```



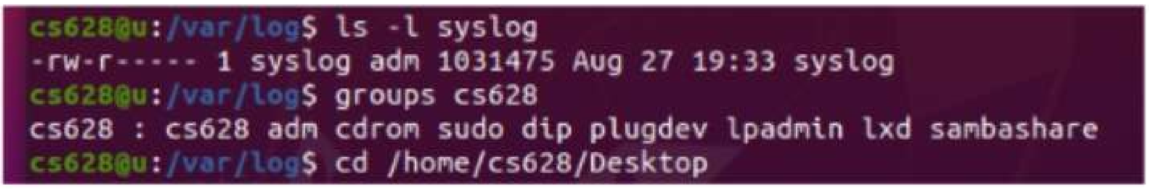
```
cs628@u:~/Desktop$ sudo gcc a.c
cs628@u:~/Desktop$ sudo ./a.out
cs628@u:~/Desktop$ cd /etc/systemd/network
cs628@u:/etc/systemd/network$ cat eth0.network
[Match]
Name=eth0

[Network]
DHCP=yes
```

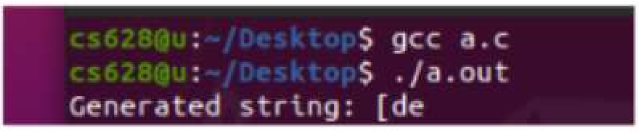
2.5. functionE()

2.5.1. Explanation, Reasoning & Snippets

We are reading a file from "/var/log" named as "syslog"
An ' other user' have no permission, and we are "cs628", apparently an other user.
But 'cs628' is not other user. We can verify by compiling and running the functionE(). It executes without sudo or set UID.
We are actually group member of "adm" so we do have read permission, and we only need the read permission in the functionE().
So we do not need system privilidges.
In the code we are simply checking this constrain"buffer[57] != '\n' && buffer[57] != '\0'" if it is passed, we are adding the buffer[57] value to the CharsString character array and printing it out.



```
cs628@u:/var/log$ ls -l syslog
-rw-r----- 1 syslog adm 1031475 Aug 27 19:33 syslog
cs628@u:/var/log$ groups cs628
cs628 : cs628 adm cdrom sudo dip plugdev lpadmin lxd sambashare
cs628@u:/var/log$ cd /home/cs628/Desktop
```



```
cs628@u:~/Desktop$ gcc a.c
cs628@u:~/Desktop$ ./a.out
Generated string: [de
```

2.6. functionF()

2.6.1. Explanation, Reasoning & Snippets

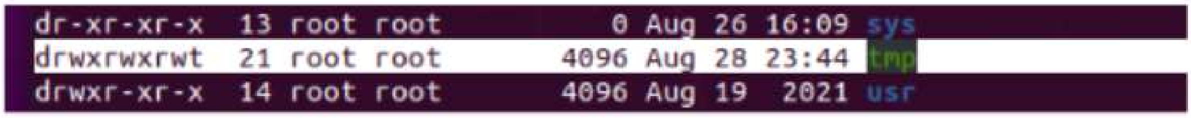
We need to take an input from main function, into functionF() and write it in a file named "temp_cJ" located at "/tmp". After this, we are removing the file. Executing fopen with write will create the file "temp_cJ" at "/tmp".

For this to happen, we (cs628) as owners must have write permission on this file and we need write and execute permissions on the directory ("/tmp") which we do have.

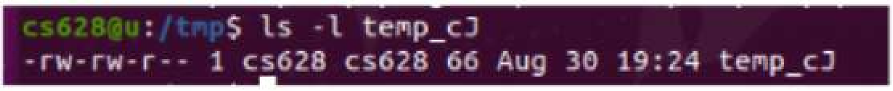
We are also using an executable named "crontab" and putting "/tmp/temp_cJ" in it, and we as other user have permission to execute the crontab.

I have commented the remove file to see if the operation is getting successful or not without sudo, and it is executed successfully.

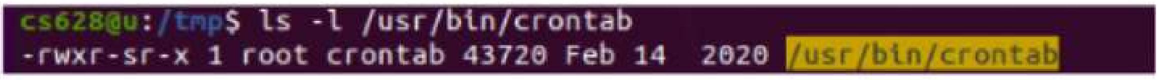
So we do not need any privilidges here.



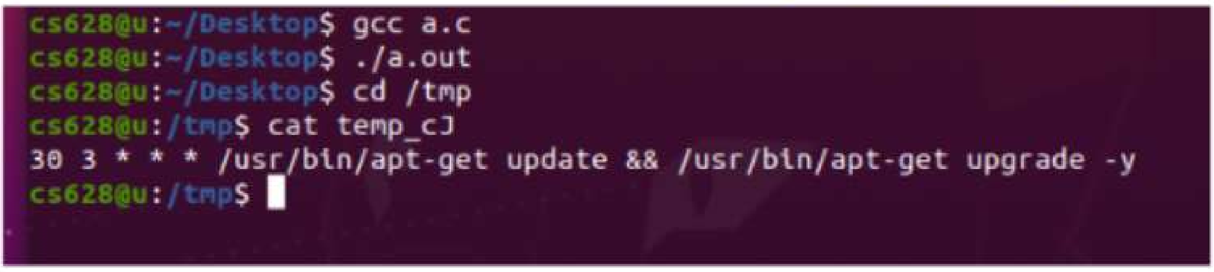
```
dr-xr-xr-x  13 root root        0 Aug 26 16:09 sys
drwxrwxrwt  21 root root     4096 Aug 28 23:44 tmp
drwxr-xr-x  14 root root     4096 Aug 19  2021 usr
```



```
cs628@u:/tmp$ ls -l temp_cJ
-rw-rw-r-- 1 cs628 cs628 66 Aug 30 19:24 temp_cJ
```



```
cs628@u:/tmp$ ls -l /usr/bin/crontab
-rwxr-sr-x 1 root crontab 43720 Feb 14  2020 /usr/bin/crontab
```

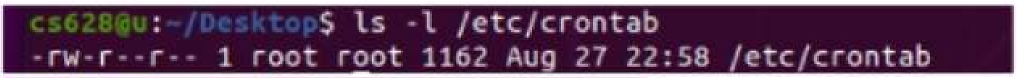


```
cs628@u:~/Desktop$ gcc a.c
cs628@u:~/Desktop$ ./a.out
cs628@u:~/Desktop$ cd /tmp
cs628@u:/tmp$ cat temp_cJ
30 3 * * * /usr/bin/apt-get update && /usr/bin/apt-get upgrade -y
cs628@u:/tmp$ 
```

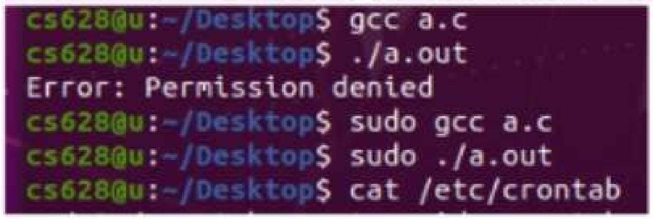
2.7. functionG()

2.7.1. Explanation, Reasoning & Snippets

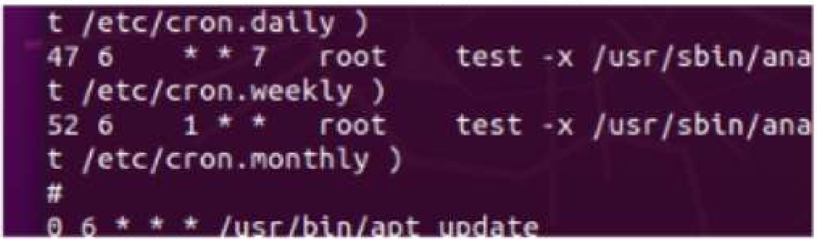
We are opeaning "/etc/crontab" in append mode and passing a string "0 6 * * * /usr/bin/apt update" via main, to append in the file.
The file "cronetab" is owned by root and the group is also root, and we are other user (cs628) and we are only permitted to read, but we need write permission to append something in the file.
Hence, we need privilidges to run the c programme functionG().



```
cs628@u:~/Desktop$ ls -l /etc/crontab
-rw-r--r-- 1 root root 1162 Aug 27 22:58 /etc/crontab
```



```
cs628@u:~/Desktop$ gcc a.c
cs628@u:~/Desktop$ ./a.out
Error: Permission denied
cs628@u:~/Desktop$ sudo gcc a.c
cs628@u:~/Desktop$ sudo ./a.out
cs628@u:~/Desktop$ cat /etc/crontab
```



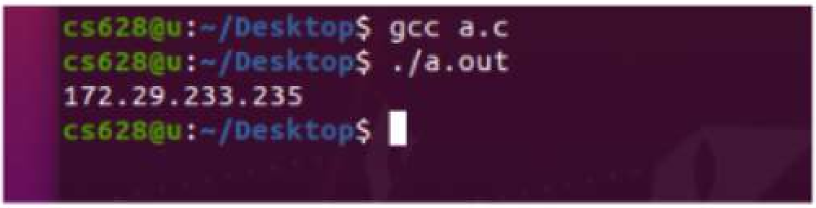
```
t /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/ana
t /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/ana
t /etc/cron.monthly )
#
0 6 * * * /usr/bin/apt update
```

2.8. functionH()

2.8.1. Explanation, Reasoning & Snippets

The functionH(), is a function which resolves IP, for a given hostname.
It is using 'getaddrinfo' to do so.
By executing the function, we can observe we do not need super user privilidges to execute this function.
But our VM do not have internet to use DNS service, so I have used a local IP to prove that our function do not need privilidges.

I have replaced the function call in main, from "functionH("www.iitk.ac.in");" to "functionH("172.29.233.235");".

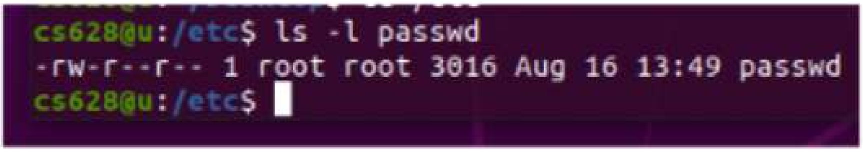


```
cs628@u:~/Desktop$ gcc a.c
cs628@u:~/Desktop$ ./a.out
172.29.233.235
cs628@u:~/Desktop$ 
```

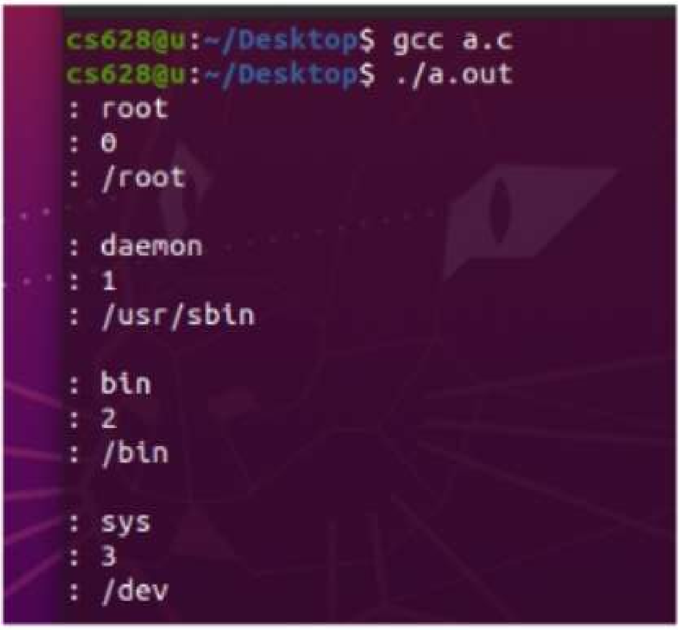
2.9. function "I()"

2.9.1. Explanation, Reasoning & Snippets

setpwent() and getpwent() gets password from "etc/passwd"
The "etc/passwd" file is owned by root and group is also root.
We (cs628) come under 'others' and we are allowed to read the file, and that is what we need in the function "I()".
Hence we do not need privilidges.
In the "I()" function, we are enumerating and displaying usernames, userids and user's home directory for all the users.



```
cs628@u:/etc$ ls -l passwd
-rw-r--r-- 1 root root 3016 Aug 16 13:49 passwd
cs628@u:/etc$ 
```



```
cs628@u:~/Desktop$ gcc a.c
cs628@u:~/Desktop$ ./a.out
: root
: 0
: /root

: daemon
: 1
: /usr/sbin

: bin
: 2
: /bin

: sys
: 3
: /dev
```

2.10. function "J()"

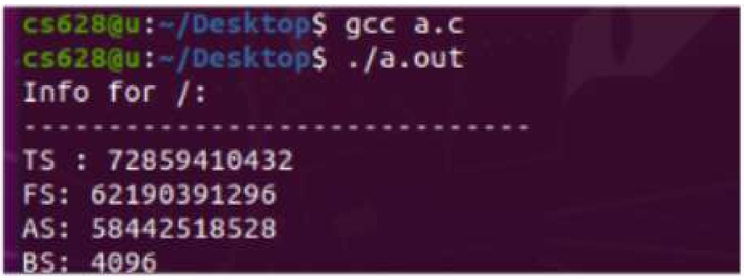
2.10.1. Explanation, Reasoning & Snippets

The function "J()" is multiplying fundamental block size with total number of blocks, free blocks and free for non-prividlidged processes blocks, to find the total size of each category.

We are using the system call "statvfs()", which returns information about a mounted filesystem. We typically do not need any privilidges for running any component of this function, we can verify this by executing the fxn. Even after being an "other" user (cs628) we are able to execute the function "J()" successfully.

Command & their meanings (retrieved from "man statvfs" command);
and abbreviations of the variables (BS,TS,FS, AS):

- f_frsize: fundamental file system block size
- f_blocks: total blocks
- f_bfree: total free blocks
- f_bavail: total free blocks for non-prividlidged process
- BS: Block size
- TS: all blocks alltogether size
- FS: overall free blocks size
- AS: total free block size available for non-prividlidged processes



```
cs628@u:~/Desktop$ gcc a.c
cs628@u:~/Desktop$ ./a.out
Info for /:
-----------------------------
TS : 72859410432
FS: 62190391296
AS: 58442518528
BS: 4096
```

***** Thank You *****