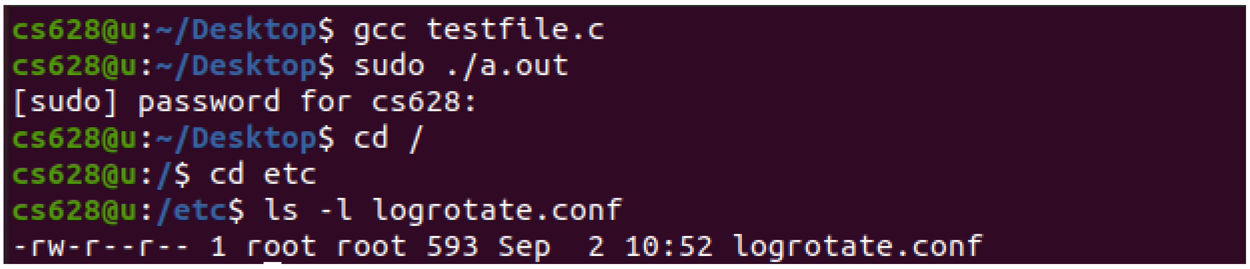
CS628 Assignment 1 – Principles of Least Privilage

Vishal Kumar

Roll Number: 231110058

functionA()

functionA() is attempting to open the "/etc/logrotate.conf" file for appending. If this file can not be opened or doesn't exist then it will print an error message, otherwise, it will append the text "Rotate = 90" to the file and close it.

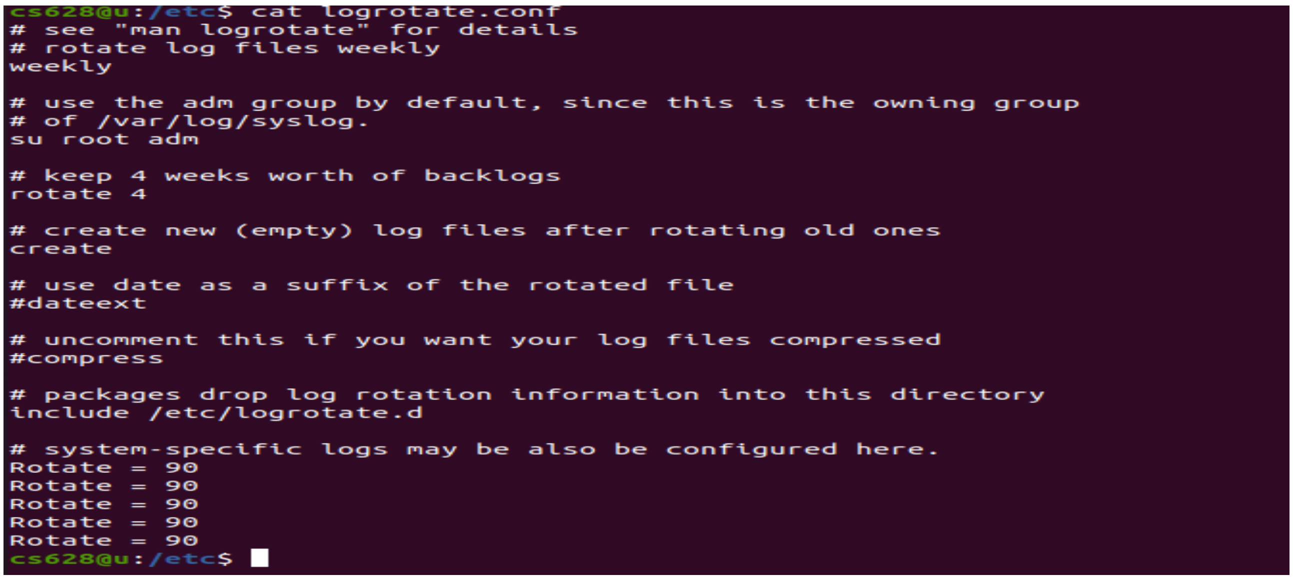


```
cs628@u:~/Desktop$ gcc testfile.c
cs628@u:~/Desktop$ sudo ./a.out
[sudo] password for cs628:
cs628@u:~/Desktop$ cd /
cs628@u:/$ cd etc
cs628@u:/etc$ ls -l logrotate.conf
-rw-r--r-- 1 root root 593 Sep  2 10:52 logrotate.conf
```

Here, we can easily see that to run this code, we needed to execute this with superuser (root) privileges.

```
FILE* file = fopen(fp, "a");
```

Here, fopen() function is used to open the file in append mode, and logrotate.conf file can only be accessed by the root for appending. So yes the functionA() requires root privilege.

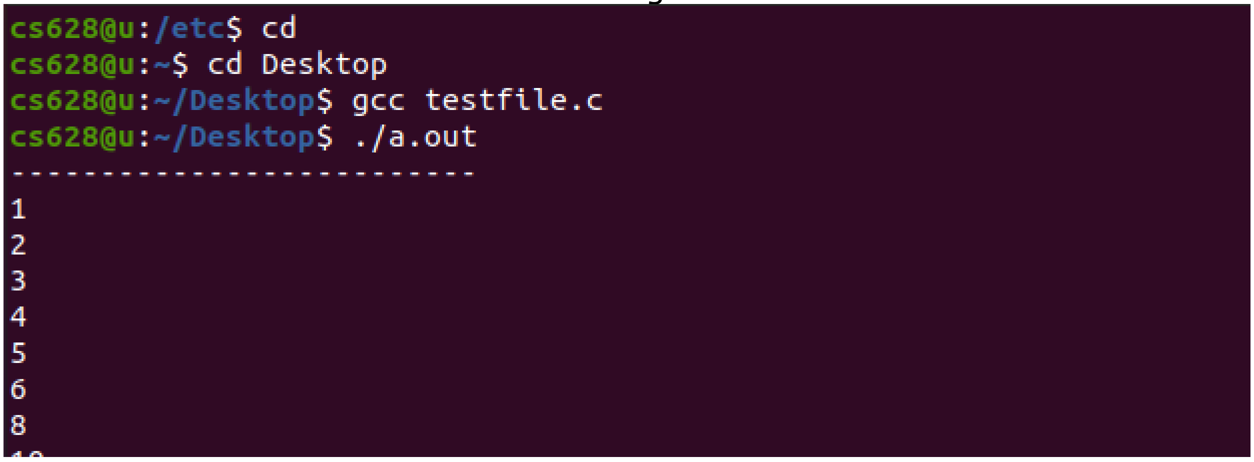


```
cs628@u:/etc$ cat logrotate.conf
# see "man logrotate" for details
# rotate log files weekly
weekly

# use the adm group by default, since this is the owning group
# of /var/log/syslog.
su root adm

# keep 4 weeks worth of backlogs
rotate 4

# create new (empty) log files after rotating old ones
create

# use date as a suffix of the rotated file
#dateext

# uncomment this if you want your log files compressed
#compress

# packages drop log rotation information into this directory
include /etc/logrotate.d

# system-specific logs may be also be configured here.
Rotate = 90
Rotate = 90
Rotate = 90
Rotate = 90
Rotate = 90
cs628@u:/etc$
```

Here, using cat command we can see that "Rotate = 90" text gets appended to the file.

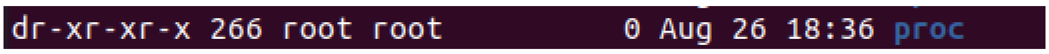
functionB()

The functionB() is designed to read and list the contents of the "/proc" directory. "/proc" is a special directory directory in linux/unix systems that contains information about running processes and some other information.
Here, it is first opening the "/proc" directory and if directory can not be opened then it will print error message. Afterwards it will loop through the directory entries and it will skip entries that do not begin with a digit. It will print the name of the entries starting with digits.

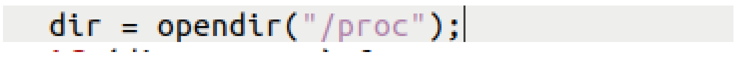


```
cs628@u:/etc$ cd
cs628@u:~$ cd Desktop
cs628@u:~/Desktop$ gcc testfile.c
cs628@u:~/Desktop$ ./a.out
--------------------------
1
2
3
4
5
6
8
```

Here we can see that our code is running without even using sudo command.



```
dr-xr-xr-x 266 root root            0 Aug 26 18:36 proc
```



```
    dir = opendir("/proc");
```

Here we can see that using opendir(), we are trying to open the directory, and since it can be read by everyone, there is no such requrirement of root privileges.
So the functionB() doesn't require root privilege.

functionC()

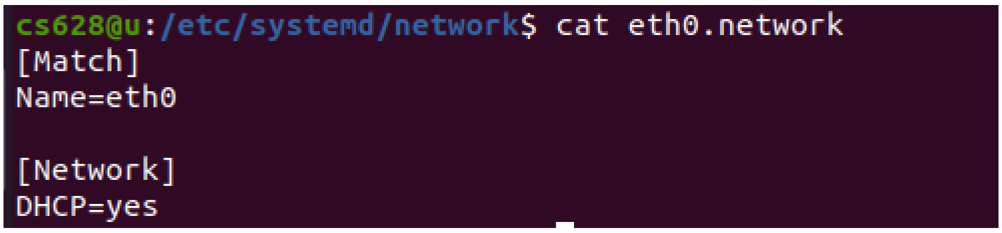
functionC() is attempting to configure a firewall using the iptables command to block outgoing traffic to google.com from a specific network segment.
First of all it is calling getifaddrs to retrieve a list of network interfaces and their addesses. Then it iterates through the list of network interfaces and checks if it's an IPV4 inteface, not a loopback, is up, not point to point, and no ARP. And then it allocates memory for subnet string, and constructs an iptable command to block outgoing traffic to google.com, and then exectues the iptables command, and then finally check the exit status of the system.

```
snprintf(command, sizeof(command), "iptables -A OUTPUT -s %s -d google.com -j DROP", snet);
```

Here we can easily see that this code is using iptables to perform firewall operations, and manipulating firewall rules requires root privileges. So yes functionC() requires root privileges.

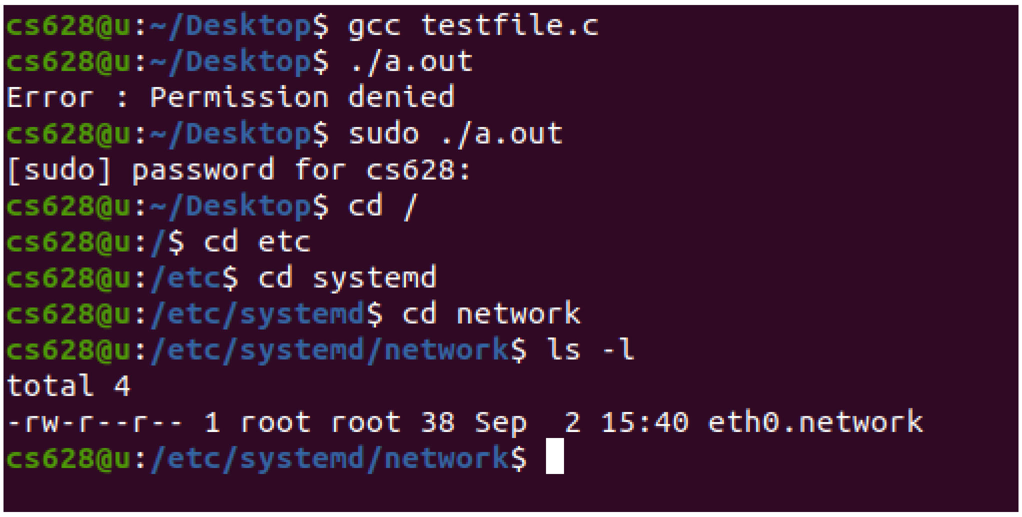
functionD()

functionD() is attempting to create a systemd network configuration file for the "eth0" interface in the "/etc/systemd/network" directory.
First of all it is taking a network interface name as an argument and then it constructs a filename for the systemd network configuration file using snprintf, and then it tries to open the file using fopen command. If the file doesn't open either due to permission issues or non existen directory, it prints an error message, whereas if the file is opened then using fprintf, it writes network configuration settings to the file.



```
cs628@u:/etc/systemd/network$ cat eth0.network
[Match]
Name=eth0

[Network]
DHCP=yes
```

Here using cat command we can see that what changes has been made to this file.
Actually fprintf sets network interface name and specifies DHCP as network configuration.

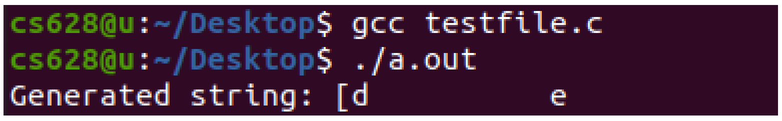


```
cs628@u:~/Desktop$ gcc testfile.c
cs628@u:~/Desktop$ ./a.out
Error : Permission denied
cs628@u:~/Desktop$ sudo ./a.out
[sudo] password for cs628:
cs628@u:~/Desktop$ cd /
cs628@u:/$ cd etc
cs628@u:/etc$ cd systemd
cs628@u:/etc/systemd$ cd network
cs628@u:/etc/systemd/network$ ls -l
total 4
-rw-r--r-- 1 root root 38 Sep  2 15:40 eth0.network
cs628@u:/etc/systemd/network$ 
```

As we know that modifying system configuration files in the "/etc/systemd/network" directory requires root privileges and in the snap we can also see that writing permission is only granted for owner which is root, so again we can easily say that functionD() requries root privileges.

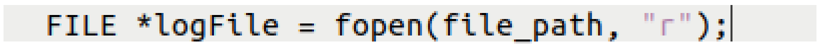
functionE()

functionE() is attempting to read the last 20 characters(if we are assuming 0-based indexing) from each line of the "/var/log/syslog" log file and it generates a string containing those characters. If successful then prints a generated string otherwise prints an error message.



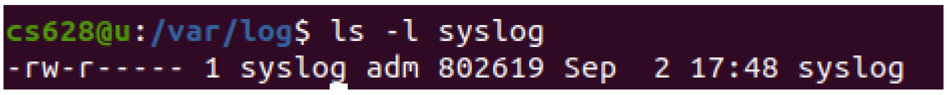
```
cs628@u:~/Desktop$ gcc testfile.c
cs628@u:~/Desktop$ ./a.out
Generated string: [d          e
```

To explain the functionality of the code, there is a custom function named my_strdup that duplicates a given string using malloc, if allocation fails, then it returns null.



```
  FILE *logFile = fopen(file_path, "r");
```

In this section of the code, we are trying to open the log file for reading using fopen().

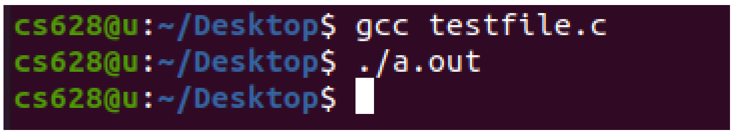


```
cs628@u:/var/log$ ls -l syslog
-rw-r----- 1 syslog adm 802619 Sep  2 17:48 syslog
```

As we can see that here, read permission is granted for the user, so we can say that functionE() doesn't require root privilege.

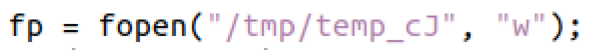
functionF()

functionF() is attempting to set up a new cron job by taking a cron job schedule(which is basically a string) as input and then writing it to a temp file and then set the crontab from the contents of the temp file using the crontab command.



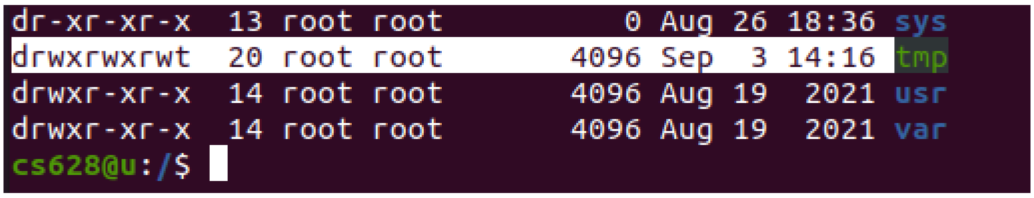
```
cs628@u:~/Desktop$ gcc testfile.c
cs628@u:~/Desktop$ ./a.out
cs628@u:~/Desktop$ 
```

As we can see here, during execution it doesn't require superuser privilege.



```
fp = fopen("/tmp/temp_cJ", "w");
```

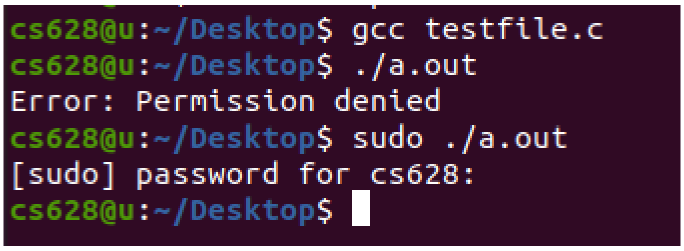
As this code creates a temporary file in the "/tmp" directory and writes the cron job information to it and we can see down below everyone is having permission on the "/tmp" directory, so definitely it doesn't require any root privileges.



```
dr-xr-xr-x  13 root root          0 Aug 26 18:36 sys
drwxrwxrwt  20 root root       4096 Sep  3 14:16 tmp
drwxr-xr-x  14 root root       4096 Aug 19  2021 usr
drwxr-xr-x  14 root root       4096 Aug 19  2021 var
cs628@u:/$ 
```

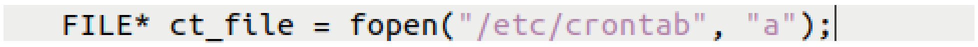
functionG()

functionG() is attemting to add a cron job to the system's crontab file located at "/etc/crontab".



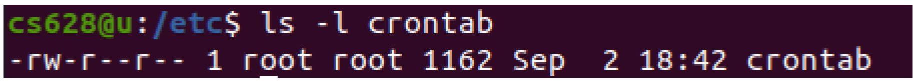
```
cs628@u:~/Desktop$ gcc testfile.c
cs628@u:~/Desktop$ ./a.out
Error: Permission denied
cs628@u:~/Desktop$ sudo ./a.out
[sudo] password for cs628:
cs628@u:~/Desktop$ 
```

As we can see above when we try to run this program it requires superuser privilege.



```
FILE* ct_file = fopen("/etc/crontab", "a");
```

When it tries to open the system's crontab file in append mode, we can see below that it can only be done with root privilege bcz only root has that permission.

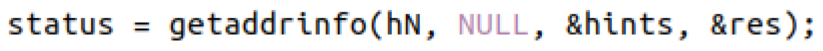


```
cs628@u:/etc$ ls -l crontab
-rw-r--r-- 1 root root 1162 Sep  2 18:42 crontab
```

So, we can say that functionG() requires root privilege.

functionH()

functionH() is attempting to resovle the domain names(hostnames) to their corresponding IP addresses using the getaddrinfo() and print them to console.

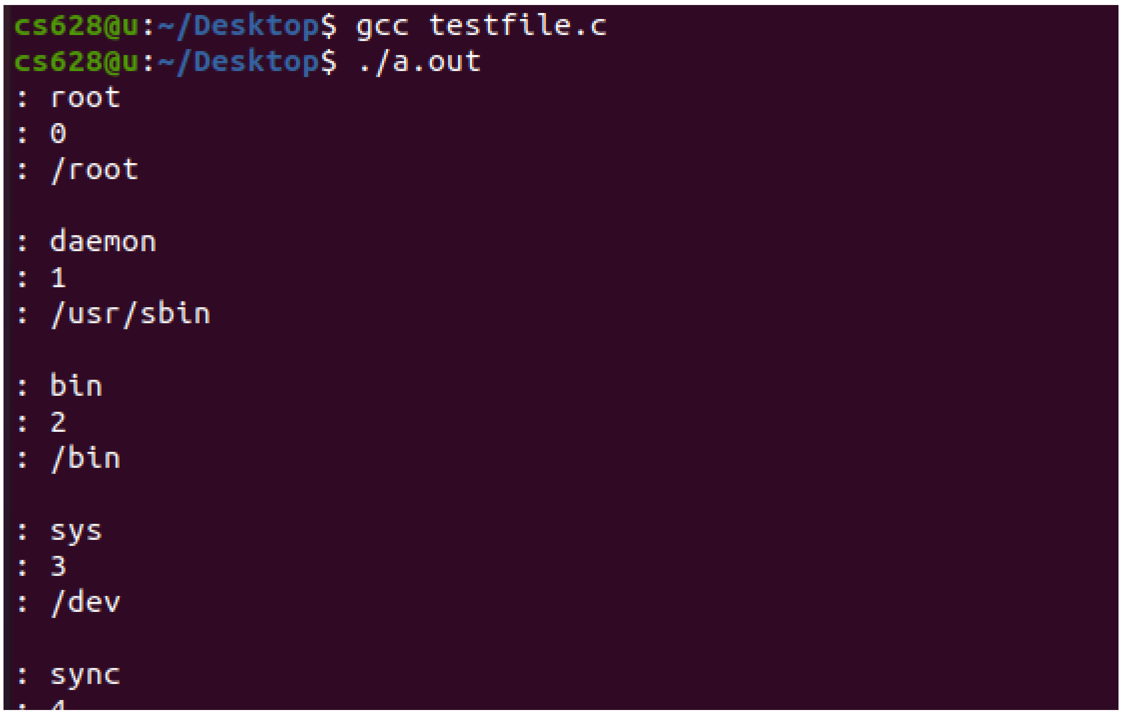


```
status = getaddrinfo(hN, NULL, &hints, &res);
```

This operation doesn't require root privileges because it's a common operation performed by user level applications to map domain names to IP addresses.
Here our code is simply performing a standard DNS resolution operation, which doesn't require root privileges.

I()

The purpose of this function I() is to list and display informations about user accounts on a unix-like system. It uses the function setpwent() to initialise the password file for reading and then using while loop it iterates through user account entries in the password file and then it uses the function getpwent() to get the next user account entry and stroes it in the pointer p. Afterwards, it prints various pieces of information about the user account which are the Username, the Userid, and the user's home directory.

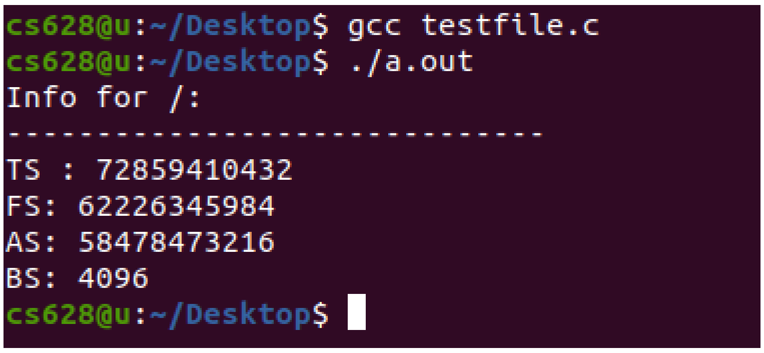


```
cs628@u:~/Desktop$ gcc testfile.c
cs628@u:~/Desktop$ ./a.out
: root
: 0
: /root

: daemon
: 1
: /usr/sbin

: bin
: 2
: /bin

: sys
: 3
: /dev

: sync
```

As I have already discussed the purpose of this code, it is used for informational purposes which basically reads publicy accessible user account data, so this function I() doesn't require any root privileges as such.

J()

The primary purpose of this function J() is retrieve and print information about the filesystem mounted at the path "/" which is root directory.



```
cs628@u:~/Desktop$ gcc testfile.c
cs628@u:~/Desktop$ ./a.out
Info for /:
------------------------------
TS : 72859410432
FS: 62226345984
AS: 58478473216
BS: 4096
cs628@u:~/Desktop$
```

As we can see above in the snap that it calculates and prints the the TS(total size), FS(free space), AS(available space), BS(block size) of the filesystem.

Now, coming to the privilege part, it doesn’t require any root privileges as it is reading publicly accessible filesystem information.

Thank you!