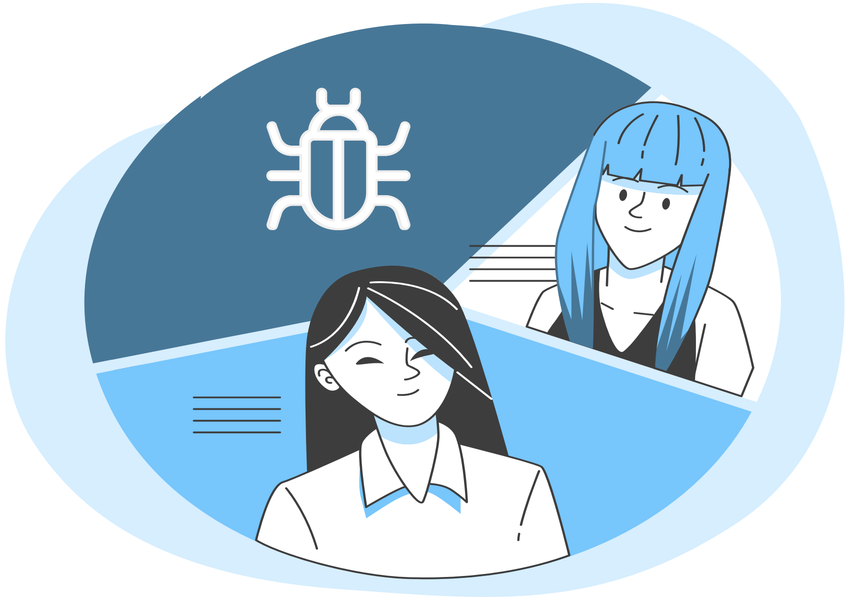
Table of Contents

Table of Contents – Contd..

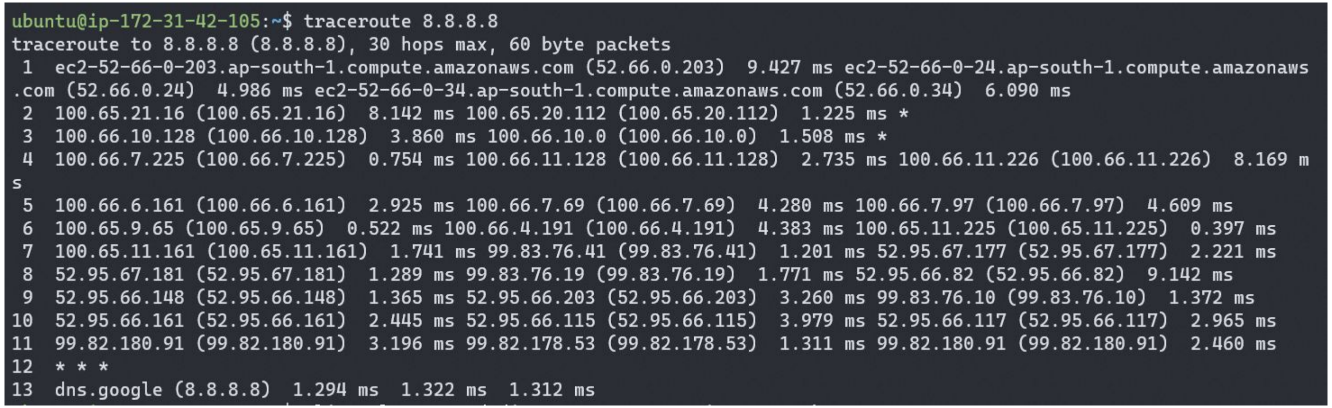
Table of Contents – Contd..



1. TRACEROUTE

- **TRACEROUTE** tracks and prints the route that the packets take from the source to the specified destination host
- The way traceroute traces the path is by incrementing the value of TTL starting from 1 and obtaining the ICMP TIME_EXCEEDED messages when each TTL expires
- Traceroute uses UDP

TRACEROUTE Contd...



```
ubuntu@ip-172-31-42-105:~$ traceroute 8.8.8.8
traceroute to 8.8.8.8 (8.8.8.8), 30 hops max, 60 byte packets
 1  ec2-52-66-0-203.ap-south-1.compute.amazonaws.com (52.66.0.203)  9.427 ms ec2-52-66-0-24.ap-south-1.compute.amazonaws
.com (52.66.0.24)  4.986 ms ec2-52-66-0-34.ap-south-1.compute.amazonaws.com (52.66.0.34)  6.090 ms
 2  100.65.21.16 (100.65.21.16)  8.142 ms 100.65.20.112 (100.65.20.112)  1.225 ms *
 3  100.66.10.128 (100.66.10.128)  3.860 ms 100.66.10.0 (100.66.10.0)  1.508 ms *
 4  100.66.7.225 (100.66.7.225)  0.754 ms 100.66.11.128 (100.66.11.128)  2.735 ms 100.66.11.226 (100.66.11.226)  8.169 m
s
 5  100.66.6.161 (100.66.6.161)  2.925 ms 100.66.7.69 (100.66.7.69)  4.280 ms 100.66.7.97 (100.66.7.97)  4.609 ms
 6  100.65.9.65 (100.65.9.65)  0.522 ms 100.66.4.191 (100.66.4.191)  4.383 ms 100.65.11.225 (100.65.11.225)  0.397 ms
 7  100.65.11.161 (100.65.11.161)  1.741 ms 99.83.76.41 (99.83.76.41)  1.201 ms 52.95.67.177 (52.95.67.177)  2.221 ms
 8  52.95.67.181 (52.95.67.181)  1.289 ms 99.83.76.19 (99.83.76.19)  1.771 ms 52.95.66.82 (52.95.66.82)  9.142 ms
 9  52.95.66.148 (52.95.66.148)  1.365 ms 52.95.66.203 (52.95.66.203)  3.260 ms 99.83.76.10 (99.83.76.10)  1.372 ms
10  52.95.66.161 (52.95.66.161)  2.445 ms 52.95.66.115 (52.95.66.115)  3.979 ms 52.95.66.117 (52.95.66.117)  2.965 ms
11  99.82.180.91 (99.82.180.91)  3.196 ms 99.82.178.53 (99.82.178.53)  1.311 ms 99.82.180.91 (99.82.180.91)  2.460 ms
12  * * *
13  dns.google (8.8.8.8)  1.294 ms  1.322 ms  1.312 ms
```

- Syntax = *traceroute <Dest_IP>*
- The first field represents the hop count. For every hop (every TTL) 3 probes are sent by traceroute in order to determine the average round trip time

TRACEROUTE Contd...

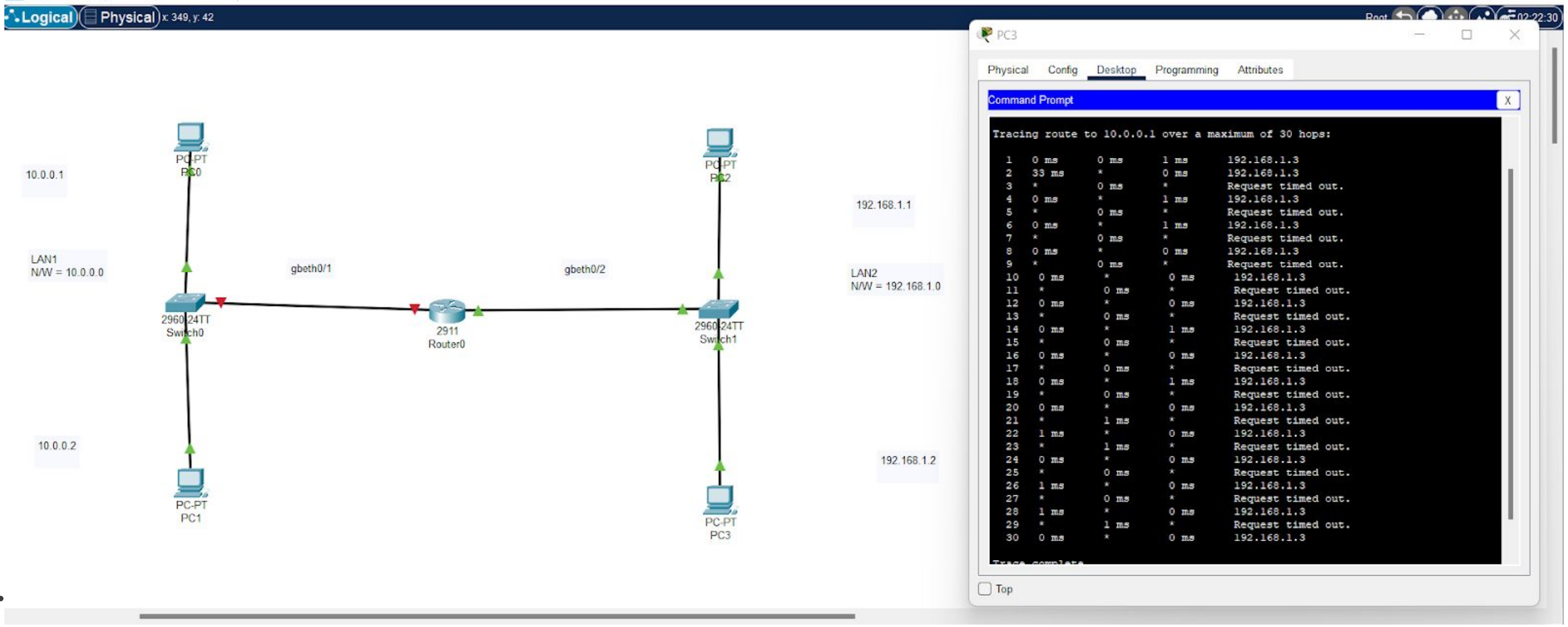
- Therefore, there will be 3 time entries displayed for every hop in the output along with the IP addresses in that hop. The IP addresses can be different if ECMP strategy is in use
- For the final TTL value, another ICMP message Destination Unreachable will be sent back to the source because traceroute makes use of unused ports for tracing

ECMP = Equal Cost Multi Path allows packets to be transmitted across multiple paths of equal costs between same source and destination.

TRACEROUTE Contd...

- **TRACERT** is traceroute's Windows OS counterpart
- By default both traceroute and tracert try for a maximum of 30 hops
- Example: From the below figure, we can infer that the packets are unable to go beyond the gateway router because the interface has been shutdown

Source: 192.168.1.2
Destination: 10.0.0.1



```
Tracing route to 10.0.0.1 over a maximum of 30 hops:

  1   0 ms      0 ms      1 ms      192.168.1.3
  2   33 ms     *         0 ms      192.168.1.3
  3   *         0 ms      *         Request timed out.
  4   0 ms      *         1 ms      192.168.1.3
  5   *         0 ms      *         Request timed out.
  6   0 ms      *         1 ms      192.168.1.3
  7   *         0 ms      *         Request timed out.
  8   0 ms      *         0 ms      192.168.1.3
  9   *         0 ms      *         Request timed out.
  10   0 ms      *         0 ms      192.168.1.3
  11   *         0 ms      *         Request timed out.
  12   0 ms      *         0 ms      192.168.1.3
  13   *         0 ms      *         Request timed out.
  14   0 ms      *         1 ms      192.168.1.3
  15   *         0 ms      *         Request timed out.
  16   0 ms      *         0 ms      192.168.1.3
  17   *         0 ms      *         Request timed out.
  18   0 ms      *         1 ms      192.168.1.3
  19   *         0 ms      *         Request timed out.
  20   0 ms      *         0 ms      192.168.1.3
  21   *         1 ms      *         Request timed out.
  22   1 ms      *         0 ms      192.168.1.3
  23   *         1 ms      *         Request timed out.
  24   0 ms      *         0 ms      192.168.1.3
  25   *         0 ms      *         Request timed out.
  26   1 ms      *         0 ms      192.168.1.3
  27   *         0 ms      *         Request timed out.
  28   1 ms      *         0 ms      192.168.1.3
  29   *         1 ms      *         Request timed out.
  30   0 ms      *         0 ms      192.168.1.3

Trace complete
```

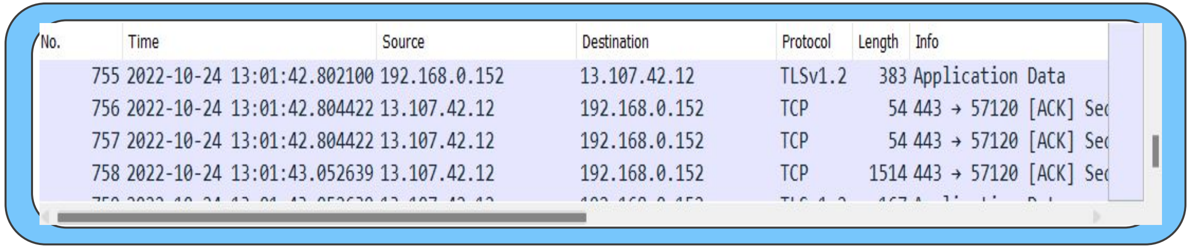
How would someone new to networking believe me if I say that traceroute really sends two types of ICMP messages?

2. WIRESHARK

- **Wireshark** is an open-source, cross-platform packet analyser used primarily for network troubleshooting
- It was previously known as Ethereal
- It puts the network interface controllers to promiscuous mode in order to capture all network traffic
- It requires pcap to run

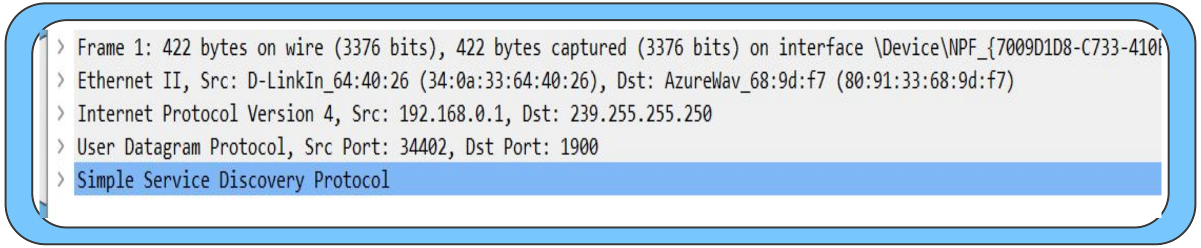
WIRESHARK Contd...

Packet List Pane

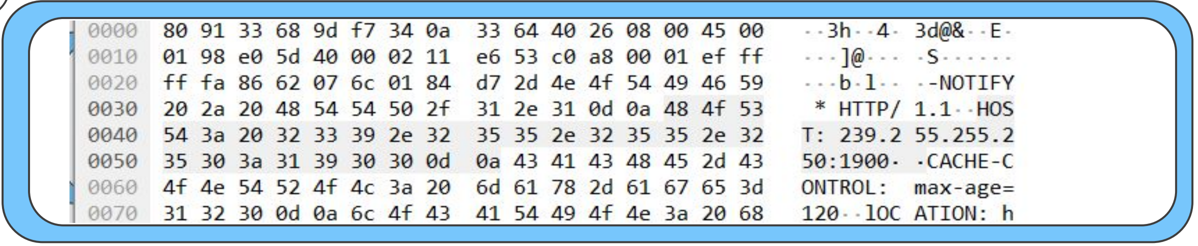


```
No.   Time                         Source           Destination      Protocol  Length  Info
  755 2022-10-24 13:01:42.802100 192.168.0.152    13.107.42.12     TLSv1.2      383 Application Data
  756 2022-10-24 13:01:42.804422 13.107.42.12     192.168.0.152    TCP           54 443 → 57120 [ACK] Se
  757 2022-10-24 13:01:42.804422 13.107.42.12     192.168.0.152    TCP           54 443 → 57120 [ACK] Se
  758 2022-10-24 13:01:43.052639 13.107.42.12     192.168.0.152    TCP         1514 443 → 57120 [ACK] Se
```

Packet Details Pane



```
> Frame 1: 422 bytes on wire (3376 bits), 422 bytes captured (3376 bits) on interface \Device\NPF_{7009D1D8-C733-410
> Ethernet II, Src: D-LinkIn_64:40:26 (34:0a:33:64:40:26), Dst: AzureWav_68:9d:f7 (80:91:33:68:9d:f7)
> Internet Protocol Version 4, Src: 192.168.0.1, Dst: 239.255.255.250
> User Datagram Protocol, Src Port: 34402, Dst Port: 1900
> Simple Service Discovery Protocol
```

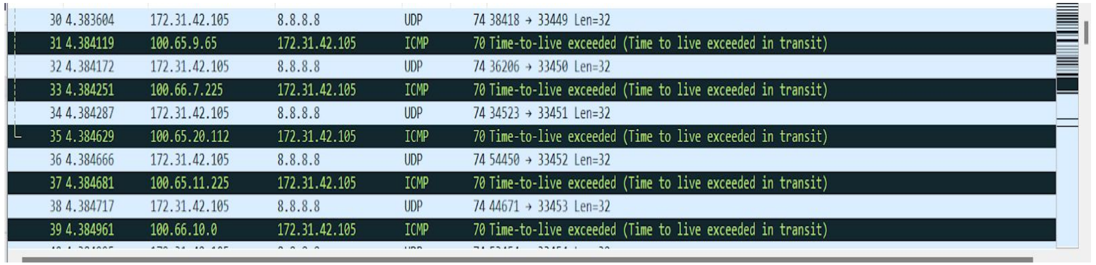


```
0000  80 91 33 68 9d f7 34 0a  33 64 40 26 08 00 45 00   ··3h··4· 3d@&··E·
0010  01 98 e0 5d 40 00 02 11  e6 53 c0 a8 00 01 ef ff   ···]@··· ·S······
0020  ff fa 86 62 07 6c 01 84  d7 2d 4e 4f 54 49 46 59   ···b·l·· ·-NOTIFY
0030  20 2a 20 48 54 54 50 2f  31 2e 31 0d 0a 48 4f 53    * HTTP/ 1.1··HOS
0040  54 3a 20 32 33 39 2e 32  35 35 2e 32 35 35 2e 32   T: 239.2 55.255.2
0050  35 30 3a 31 39 30 30 0d  0a 43 41 43 48 45 2d 43   50:1900· ·CACHE-C
0060  4f 4e 54 52 4f 4c 3a 20  6d 61 78 2d 61 67 65 3d   ONTROL:  max-age=
0070  31 32 30 0d 0a 6c 4f 43  41 54 49 4f 4e 3a 20 68   120··lOC ATION: h
```

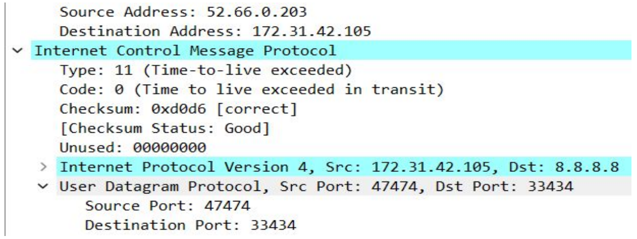
Packet Bytes Pane

WIRESHARK Contd...

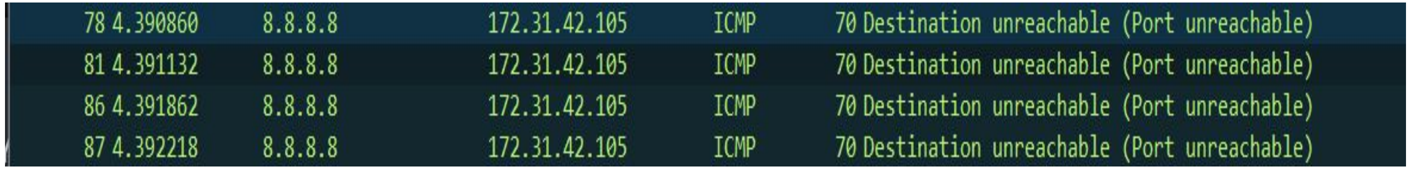
Example of running Wireshark while Traceroute



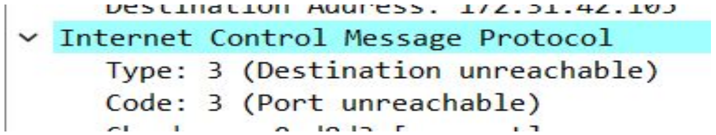
```
30 4.383604  172.31.42.105  8.8.8.8        UDP   74 38418 → 33449 Len=32
31 4.384119  100.65.9.65    172.31.42.105  ICMP  70 Time-to-live exceeded (Time to live exceeded in transit)
32 4.384172  172.31.42.105  8.8.8.8        UDP   74 36206 → 33450 Len=32
33 4.384251  100.66.7.225   172.31.42.105  ICMP  70 Time-to-live exceeded (Time to live exceeded in transit)
34 4.384287  172.31.42.105  8.8.8.8        UDP   74 34523 → 33451 Len=32
35 4.384629  100.65.20.112  172.31.42.105  ICMP  70 Time-to-live exceeded (Time to live exceeded in transit)
36 4.384666  172.31.42.105  8.8.8.8        UDP   74 54450 → 33452 Len=32
37 4.384681  100.65.11.225  172.31.42.105  ICMP  70 Time-to-live exceeded (Time to live exceeded in transit)
38 4.384717  172.31.42.105  8.8.8.8        UDP   74 44671 → 33453 Len=32
39 4.384961  100.66.10.0    172.31.42.105  ICMP  70 Time-to-live exceeded (Time to live exceeded in transit)
```



```
    Source Address: 52.66.0.203
    Destination Address: 172.31.42.105
⌄ Internet Control Message Protocol
    Type: 11 (Time-to-live exceeded)
    Code: 0 (Time to live exceeded in transit)
    Checksum: 0xd0d6 [correct]
    [Checksum Status: Good]
    Unused: 00000000
  > Internet Protocol Version 4, Src: 172.31.42.105, Dst: 8.8.8.8
  ⌄ User Datagram Protocol, Src Port: 47474, Dst Port: 33434
      Source Port: 47474
      Destination Port: 33434
```



```
78 4.390860  8.8.8.8  172.31.42.105  ICMP  70 Destination unreachable (Port unreachable)
81 4.391132  8.8.8.8  172.31.42.105  ICMP  70 Destination unreachable (Port unreachable)
86 4.391862  8.8.8.8  172.31.42.105  ICMP  70 Destination unreachable (Port unreachable)
87 4.392218  8.8.8.8  172.31.42.105  ICMP  70 Destination unreachable (Port unreachable)
```



```
    Destination Address: 172.31.42.105
⌄ Internet Control Message Protocol
    Type: 3 (Destination unreachable)
    Code: 3 (Port unreachable)
```

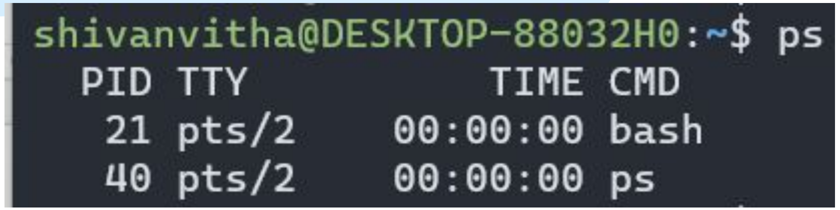
3. PS

- With the help of **ps** command and its options we can retrieve a snapshot of the processes and their related information active on our system
- Syntax = *ps <options>*
- If one wants a real time view of the active processes, top command can be used

PS Contd...

Running **ps** without any options yields 4 fields:

1. **PID:** Process ID
2. **TTY:** Terminal from which the process has started
3. **TIME:** Total CPU time used by the process since it began
4. **CMD:** Command that is used to generate the process

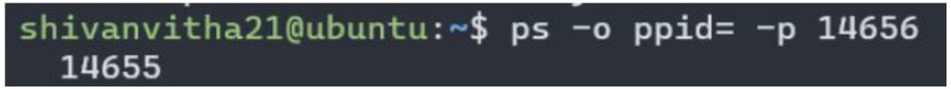


```
shivanvitha@DESKTOP-88032H0:~$ ps
  PID TTY          TIME CMD
   21 pts/2    00:00:00 bash
   40 pts/2    00:00:00 ps
```

PS Contd...

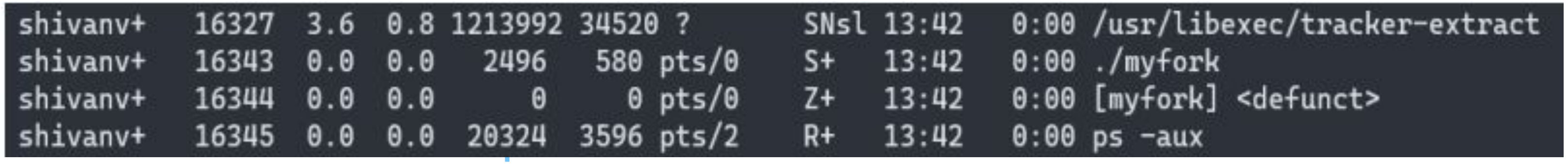
Option	Description
-A	All running processes
-a	All processes except session leaders
-u	Selection by EUID
-p	Select by PID
-o	Print in User defined format

PS Contd...



```
shivanvitha21@ubuntu:~$ ps -o ppid= -p 14656
  14655
```

Fetching the parent PID of the specified PID



```
shivanv+   16327  3.6  0.8 1213992 34520 ?       SNsl 13:42   0:00 /usr/libexec/tracker-extract
shivanv+   16343  0.0  0.0   2496   580 pts/0    S+   13:42   0:00 ./myfork
shivanv+   16344  0.0  0.0      0     0 pts/0    Z+   13:42   0:00 [myfork] <defunct>
shivanv+   16345  0.0  0.0  20324  3596 pts/2    R+   13:42   0:00 ps -aux
```

Spotting zombie processes in STAT field

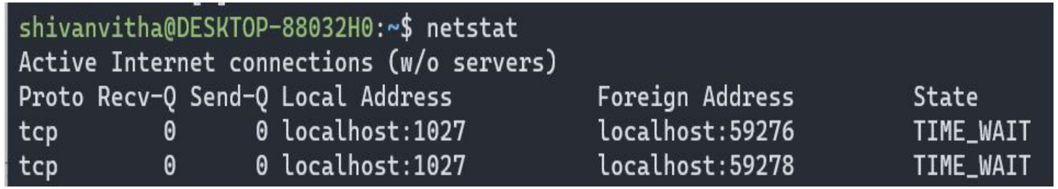
4. NETSTAT

- **Netstat** (Network Statistics) is a command line utility that prints network connections, interface statistics, routing tables, etc..
- On Linux, netstat is part of net-tools package
- Syntax = *netstat <options>*

NETSTAT Contd...

Important fields in the output are:

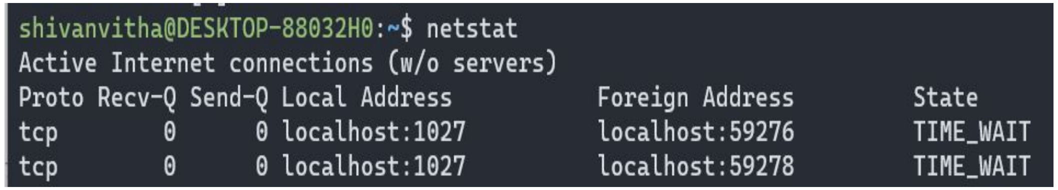
1. **Proto:** Protocol used by the socket
2. **Local Address:** Address and port of the local end of the socket



```
shivanvitha@DESKTOP-88032H0:~$ netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 localhost:1027          localhost:59276         TIME_WAIT
tcp        0      0 localhost:1027          localhost:59278         TIME_WAIT
```

NETSTAT Contd...

3. **Foreign Address:** Address and port of the remote end
4. **State:** State of the socket
5. **Recv-Q:** Current syn backlog
6. **Send-Q:** Maximum size of syn backlog



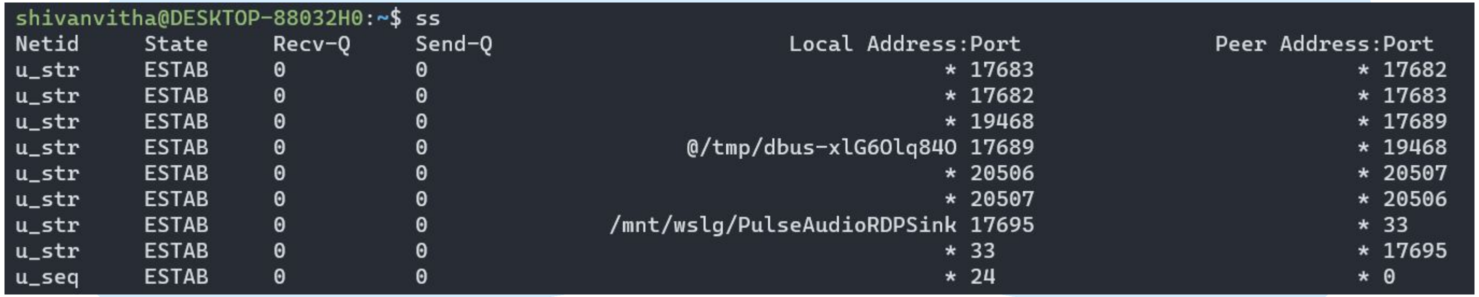
```
shivanvitha@DESKTOP-88032H0:~$ netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 localhost:1027          localhost:59276         TIME_WAIT
tcp        0      0 localhost:1027          localhost:59278         TIME_WAIT
```

NETSTAT Contd...

Option	Description
-r	Prints kernel routing tables
-i	Displays all network interfaces
-s	Shows statistics for each protocol
-l	Displays only the listening sockets
-a	Shows listening and non-listening sockets

SS

- Netstat has almost become obsolete
- It has been superseded by **ss**
- **ss** is part of iproute2



```
shivanvitha@DESKTOP-88032H0:~$ ss
Netid       State       Recv-Q       Send-Q                       Local Address:Port                  Peer Address:Port
u_str       ESTAB       0            0                                      * 17683                            * 17682
u_str       ESTAB       0            0                                      * 17682                            * 17683
u_str       ESTAB       0            0                                      * 19468                            * 17689
u_str       ESTAB       0            0                   @/tmp/dbus-xlG6Olq840 17689                            * 19468
u_str       ESTAB       0            0                                      * 20506                            * 20507
u_str       ESTAB       0            0                                      * 20507                            * 20506
u_str       ESTAB       0            0              /mnt/wslg/PulseAudioRDPSink 17695                            * 33
u_str       ESTAB       0            0                                      * 33                               * 17695
u_seq       ESTAB       0            0                                      * 24                               * 0
```

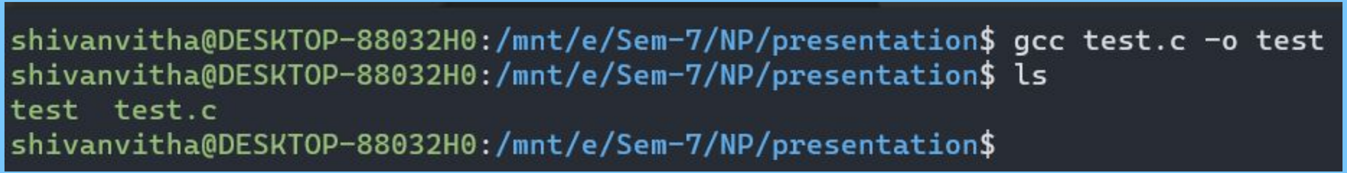
5. GCC

- **GCC (GNU Compiler Collection)** is an integrated distribution of compilers for many major languages
- Some of the major languages include C, C++, Ada, etc..
- For languages other than C, compilers have their own names, g++ for C++ and GNAT for Ada

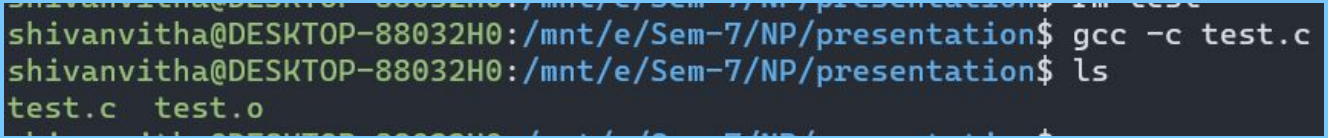
GCC Contd...

Option	Description
-c	Compiles and assembles but does not link
-o	Places the output in specified file
-S	Only compiles; not assembles or links
-Wall	Displays all warnings
–help	Shows help for usage

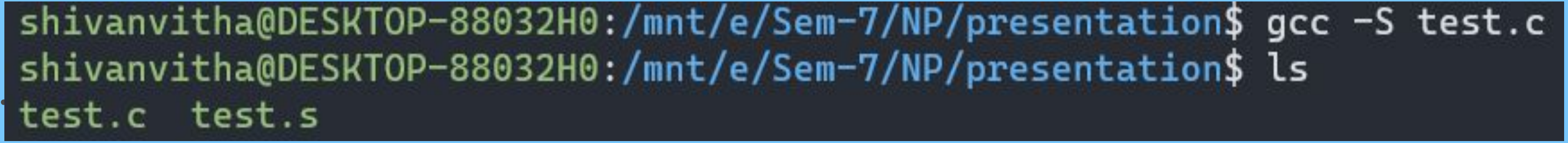
GCC Contd...



```
shivanvitha@DESKTOP-88032H0:/mnt/e/Sem-7/NP/presentation$ gcc test.c -o test
shivanvitha@DESKTOP-88032H0:/mnt/e/Sem-7/NP/presentation$ ls
test  test.c
shivanvitha@DESKTOP-88032H0:/mnt/e/Sem-7/NP/presentation$
```



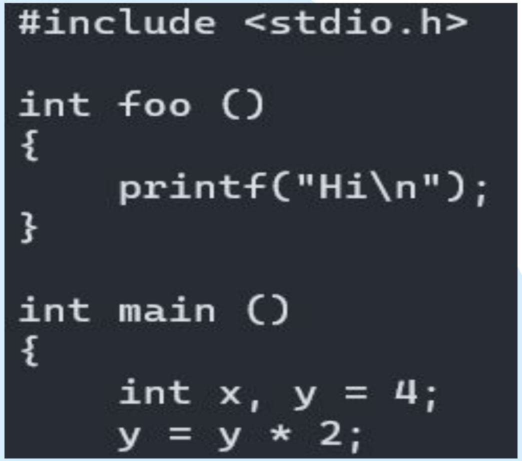
```
shivanvitha@DESKTOP-88032H0:/mnt/e/Sem-7/NP/presentation$ gcc -c test.c
shivanvitha@DESKTOP-88032H0:/mnt/e/Sem-7/NP/presentation$ ls
test.c  test.o
```



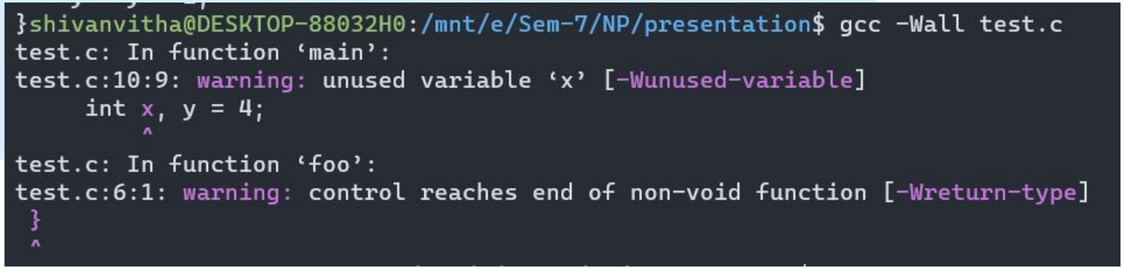
```
shivanvitha@DESKTOP-88032H0:/mnt/e/Sem-7/NP/presentation$ gcc -S test.c
shivanvitha@DESKTOP-88032H0:/mnt/e/Sem-7/NP/presentation$ ls
test.c  test.s
```

GCC Contd...

By using the **Wall** option as shown in the figure, gcc displays the warnings that there is an unused variable and a function which did not return according to the type specified.



```
#include <stdio.h>

int foo ()
{
        printf("Hi\n");
}

int main ()
{
        int x, y = 4;
        y = y * 2;
```



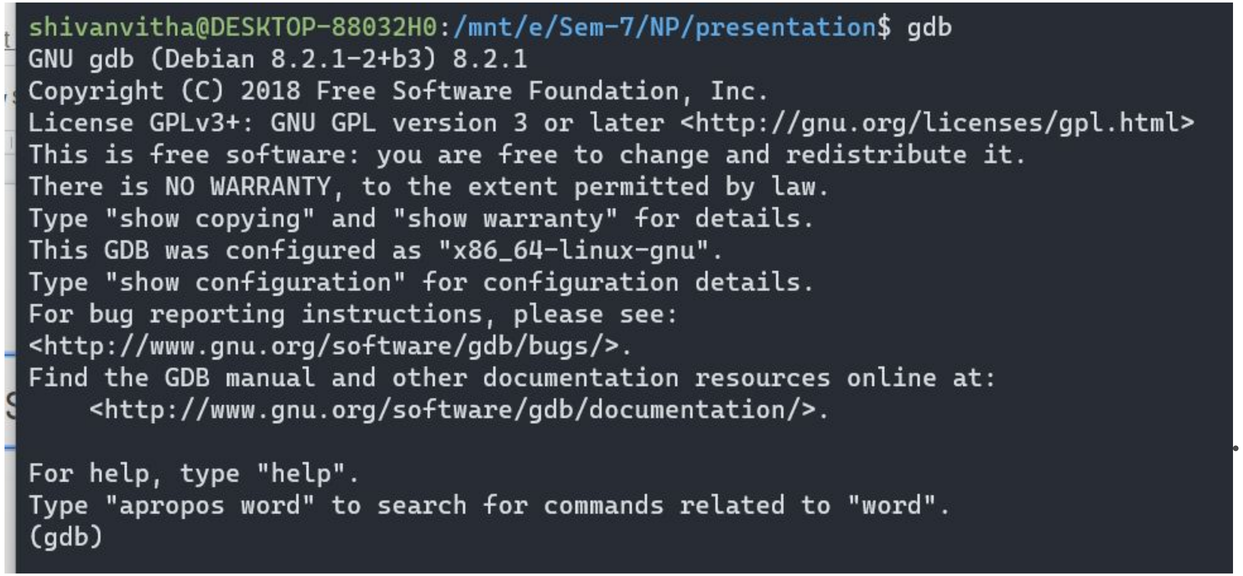
```
}shivanvitha@DESKTOP-88032H0:/mnt/e/Sem-7/NP/presentation$ gcc -Wall test.c
test.c: In function ‘main’:
test.c:10:9: warning: unused variable ‘x’ [-Wunused-variable]
     int x, y = 4;
         ^
test.c: In function ‘foo’:
test.c:6:1: warning: control reaches end of non-void function [-Wreturn-type]
 }
 ^
```

6. GDB

- **GDB (GNU Debugger)** is a portable debugger that works for various languages like C, C++, Objective-C, etc..
- It provides facilities to operate on executable files
- GDB uses **ptrace** (process trace) system call to examine the executing process

GDB Contd...

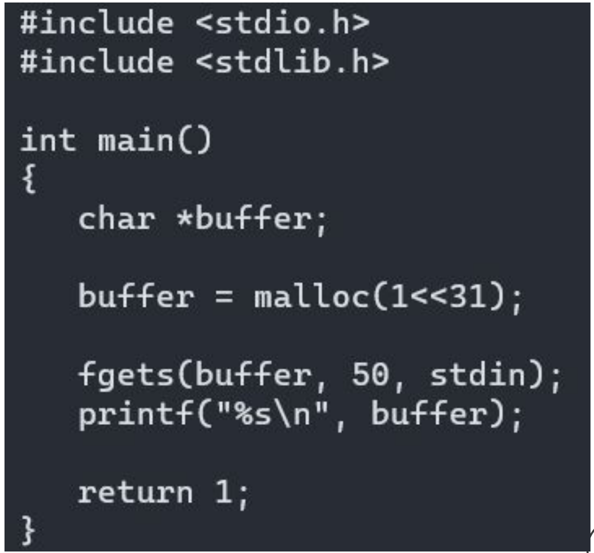
To start up gdb, type “gdb” in the console



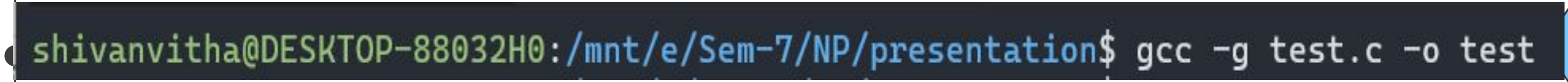
```
shivanvitha@DESKTOP-88032H0:/mnt/e/Sem-7/NP/presentation$ gdb
GNU gdb (Debian 8.2.1-2+b3) 8.2.1
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word".
(gdb)
```

GDB Contd...

We use the **g** option with gcc to include debug information to the executable.
The given program is supposed to generate segmentation fault because the size specified in malloc cannot be allocated.



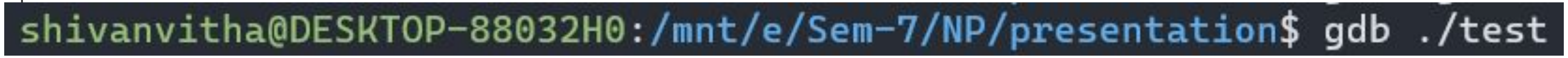
```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    char *buffer;

    buffer = malloc(1<<31);

    fgets(buffer, 50, stdin);
    printf("%s\n", buffer);

    return 1;
}
```



```
shivanvitha@DESKTOP-88032H0:/mnt/e/Sem-7/NP/presentation$ gcc -g test.c -o test
```

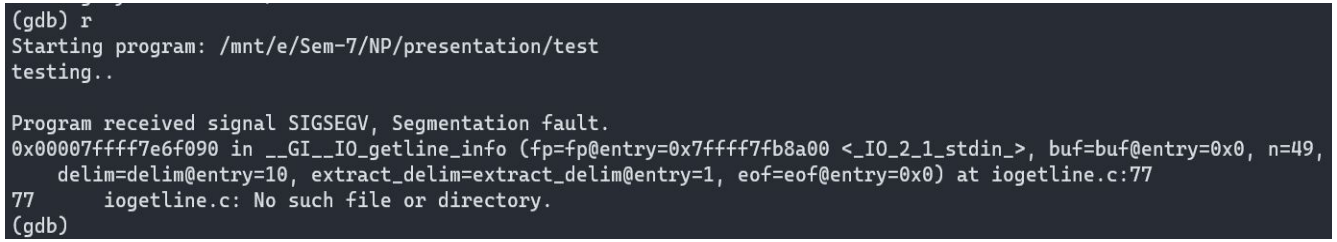
GDB Contd...

The executable should be run along with **gdb** command as shown



```
shivanvitha@DESKTOP-88032H0:/mnt/e/Sem-7/NP/presentation$ gdb ./test
```

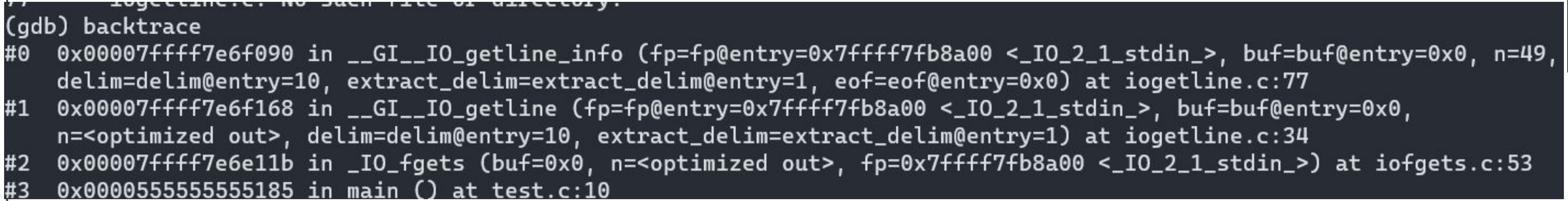
We can see that the program received **SIGSEGV** signal



```
(gdb) r
Starting program: /mnt/e/Sem-7/NP/presentation/test
testing..

Program received signal SIGSEGV, Segmentation fault.
0x00007ffff7e6f090 in __GI__IO_getline_info (fp=fp@entry=0x7ffff7fb8a00 <_IO_2_1_stdin_>, buf=buf@entry=0x0, n=49,
    delim=delim@entry=10, extract_delim=extract_delim@entry=1, eof=eof@entry=0x0) at iogetline.c:77
77      iogetline.c: No such file or directory.
(gdb)
```

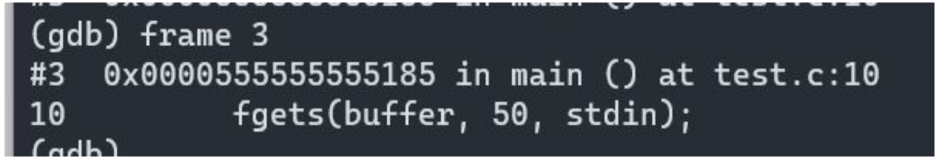
GDB Contd...

The result on **backtrace** is as follows,



```
77      iogetline.c: No such file or directory.
(gdb) backtrace
#0  0x00007ffff7e6f090 in __GI__IO_getline_info (fp=fp@entry=0x7ffff7fb8a00 <_IO_2_1_stdin_>, buf=buf@entry=0x0, n=49,
    delim=delim@entry=10, extract_delim=extract_delim@entry=1, eof=eof@entry=0x0) at iogetline.c:77
#1  0x00007ffff7e6f168 in __GI__IO_getline (fp=fp@entry=0x7ffff7fb8a00 <_IO_2_1_stdin_>, buf=buf@entry=0x0,
    n=<optimized out>, delim=delim@entry=10, extract_delim=extract_delim@entry=1) at iogetline.c:34
#2  0x00007ffff7e6e11b in _IO_fgets (buf=0x0, n=<optimized out>, fp=0x7ffff7fb8a00 <_IO_2_1_stdin_>) at iofgets.c:53
#3  0x0000555555555185 in main () at test.c:10
```

To get only the frame of our program, we use **frame** option

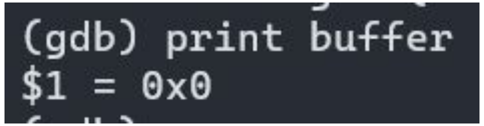


```
#3  0x0000555555555185 in main () at test.c:10
(gdb) frame 3
#3  0x0000555555555185 in main () at test.c:10
10          fgets(buffer, 50, stdin);
(gdb)
```

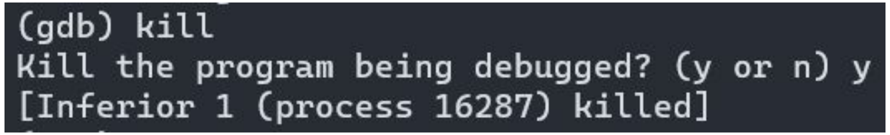
The program must have crashed on call to fgets and most probably on the “buffer” argument

GDB Contd...

We see that the buffer is a NULL pointer



```
(gdb) print buffer
$1 = 0x0
```



```
(gdb) kill
Kill the program being debugged? (y or n) y
[Inferior 1 (process 16287) killed]
```

To kill the program being debugged we use **kill** option

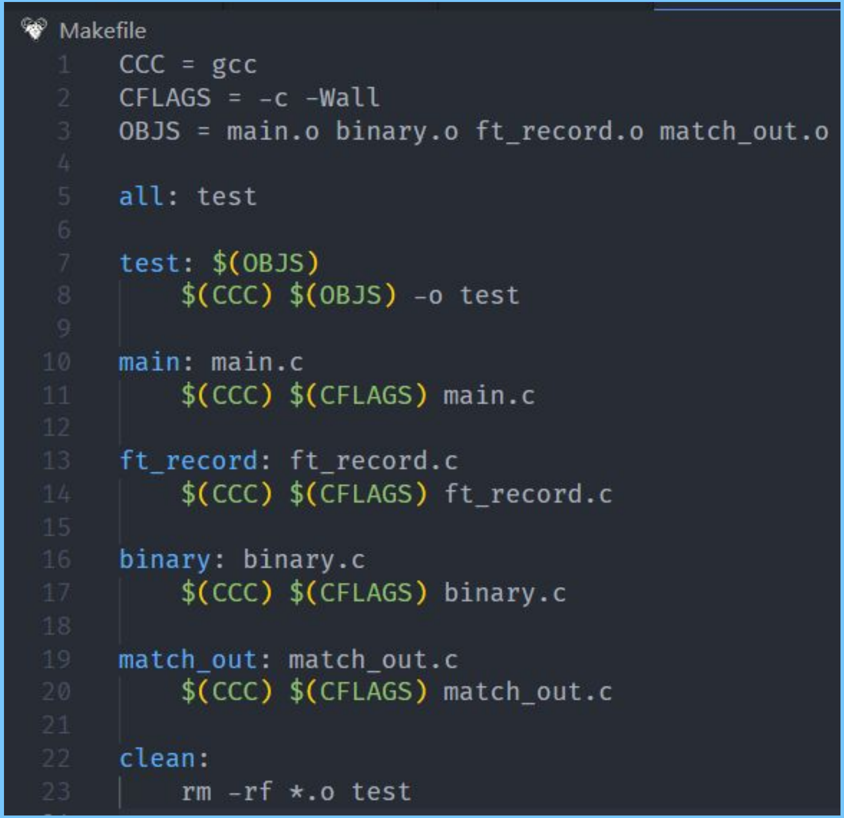
Option	Description
l	Prints 10 lines of code by default
b	To issue a breakpoint
q	To quit gdb

7. MAKE

- Make is a command line utility that helps us to manage and maintain a large number of files with the help of a **Makefile**
- It helps us to compile a bunch of source files at a time and link them to an executable
- By default make looks for GNUmakefile, makefile or Makefile in the same order if custom make file name is not provided with **f** option

MAKE Contd...

- In the Makefile shown, **CCC**, **CFLAGS** and **OBJS** are the makefile variables
- The variable are accessed via the **$** sign

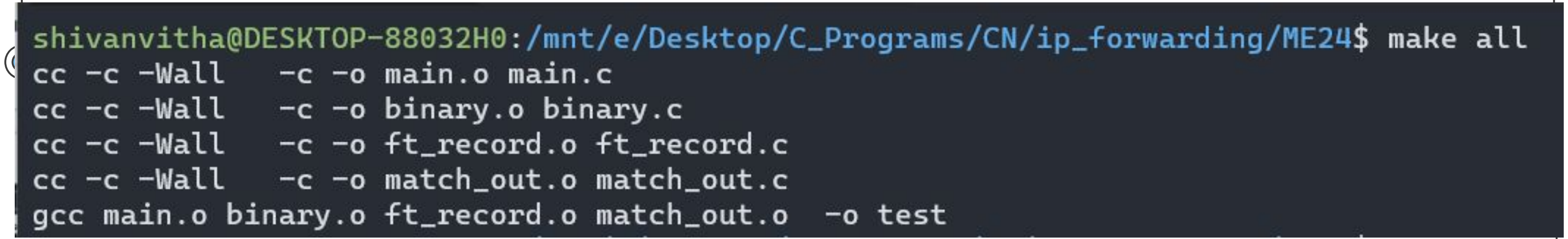


Makefile

```
CCC = gcc
CFLAGS = -c -Wall
OBJS = main.o binary.o ft_record.o match_out.o

all: test

test: $(OBJS)
	$(CCC) $(OBJS) -o test

main: main.c
	$(CCC) $(CFLAGS) main.c

ft_record: ft_record.c
	$(CCC) $(CFLAGS) ft_record.c

binary: binary.c
	$(CCC) $(CFLAGS) binary.c

match_out: match_out.c
	$(CCC) $(CFLAGS) match_out.c

clean:
	rm -rf *.o test
```

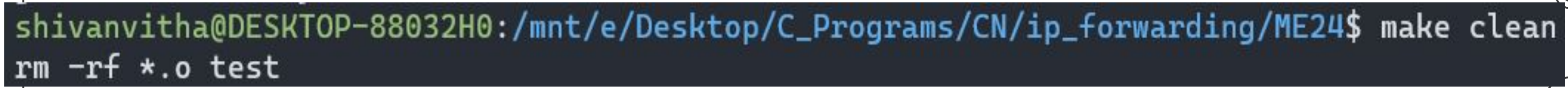
MAKE Contd...

To create all object files and link them,



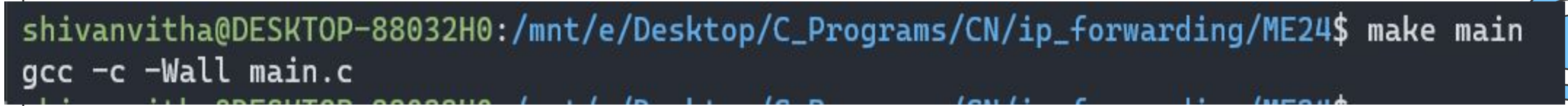
```
shivanvitha@DESKTOP-88032H0:/mnt/e/Desktop/C_Programs/CN/ip_forwarding/ME24$ make all
cc -c -Wall   -c -o main.o main.c
cc -c -Wall   -c -o binary.o binary.c
cc -c -Wall   -c -o ft_record.o ft_record.c
cc -c -Wall   -c -o match_out.o match_out.c
gcc main.o binary.o ft_record.o match_out.o  -o test
```

To clean the object files,



```
shivanvitha@DESKTOP-88032H0:/mnt/e/Desktop/C_Programs/CN/ip_forwarding/ME24$ make clean
rm -rf *.o test
```

To compile only one file (say main.c)



```
shivanvitha@DESKTOP-88032H0:/mnt/e/Desktop/C_Programs/CN/ip_forwarding/ME24$ make main
gcc -c -Wall main.c
```

8. TCPDUMP

- **tcpdump** is a network data packet analyzer that runs on a command line interface.
- It allows the user to display TCP/IP and other packets being transmitted or received over a network.
- It works on most Unix-like operating systems, and uses the libpcap library to capture packets.
- The port of tcpdump for Windows is called WinDump; it uses WinPcap, the Windows version of libpcap.
- Syntax = *tcpdump [OPTIONS]*

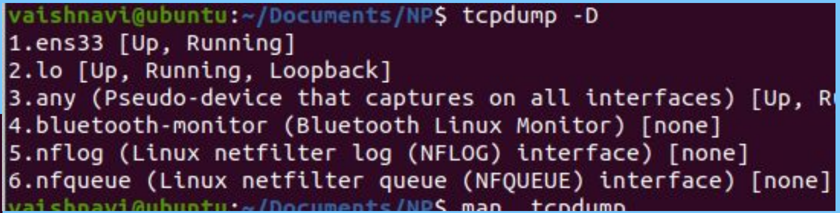
TCPDUMP Contd...

- tcpdump can save captured information in the form of a .pcap file, which can be viewed by tcpdump command or GUI Wireshark
- In some operating systems, a user must have superuser privileges to use tcpdump

TCPDUMP Contd...

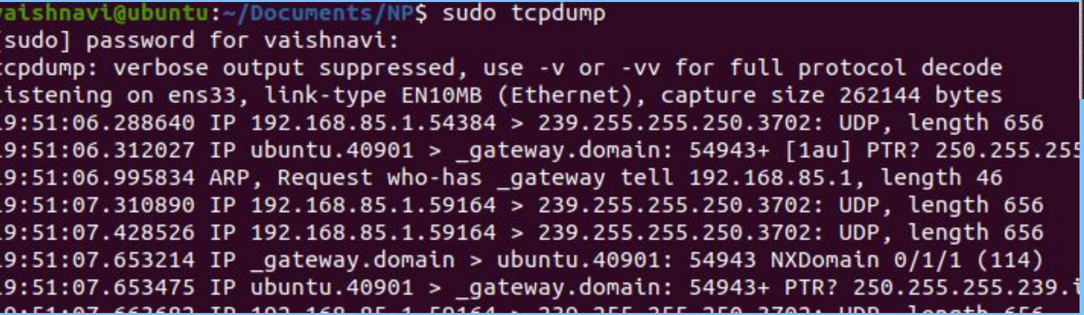
Option	Description
-D	Prints a list of available interfaces
-i	Prints packets from a specific interface
-A	Prints captured packets in ASCII format
-c	Captures only a specific number of packets
-w	Saves a captured packets into a given file
-r	Reads captured packets from a given file
tcp	To capture only TCP packets

TCPDUMP Contd...



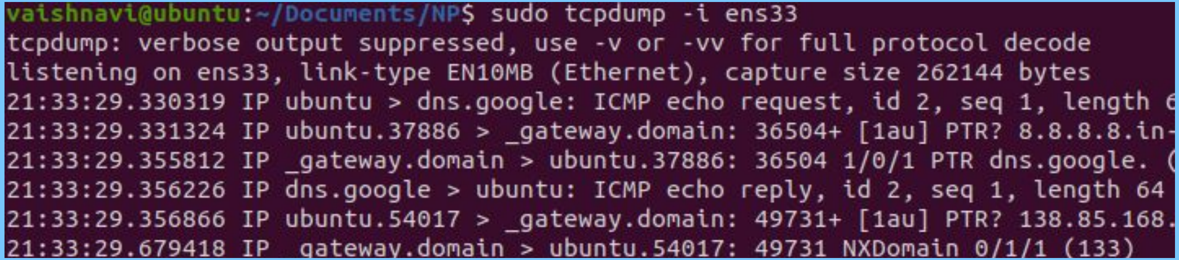
```
vaishnavi@ubuntu:~/Documents/NP$ tcpdump -D
1.ens33 [Up, Running]
2.lo [Up, Running, Loopback]
3.any (Pseudo-device that captures on all interfaces) [Up, R
4.bluetooth-monitor (Bluetooth Linux Monitor) [none]
5.nflog (Linux netfilter log (NFLOG) interface) [none]
6.nfqueue (Linux netfilter queue (NFQUEUE) interface) [none]
vaishnavi@ubuntu:~/Documents/NP$ man  tcpdump
```

Listing all network interfaces



```
aishnavi@ubuntu:~/Documents/NP$ sudo tcpdump
sudo] password for vaishnavi:
cpdump: verbose output suppressed, use -v or -vv for full protocol decode
istening on ens33, link-type EN10MB (Ethernet), capture size 262144 bytes
9:51:06.288640 IP 192.168.85.1.54384 > 239.255.255.250.3702: UDP, length 656
9:51:06.312027 IP ubuntu.40901 > _gateway.domain: 54943+ [1au] PTR? 250.255.255
9:51:06.995834 ARP, Request who-has _gateway tell 192.168.85.1, length 46
9:51:07.310890 IP 192.168.85.1.59164 > 239.255.255.250.3702: UDP, length 656
9:51:07.428526 IP 192.168.85.1.59164 > 239.255.255.250.3702: UDP, length 656
9:51:07.653214 IP _gateway.domain > ubuntu.40901: 54943 NXDomain 0/1/1 (114)
9:51:07.653475 IP ubuntu.40901 > _gateway.domain: 54943+ PTR? 250.255.255.239.i
9:51:07.663683 IP 192.168.85.1.59164 > 239.255.255.250.3702: UDP, length 656
```

Starting tcpdump



```
vaishnavi@ubuntu:~/Documents/NP$ sudo tcpdump -i ens33
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ens33, link-type EN10MB (Ethernet), capture size 262144 bytes
21:33:29.330319 IP ubuntu > dns.google: ICMP echo request, id 2, seq 1, length 6
21:33:29.331324 IP ubuntu.37886 > _gateway.domain: 36504+ [1au] PTR? 8.8.8.8.in-
21:33:29.355812 IP _gateway.domain > ubuntu.37886: 36504 1/0/1 PTR dns.google. (
21:33:29.356226 IP dns.google > ubuntu: ICMP echo reply, id 2, seq 1, length 64
21:33:29.356866 IP ubuntu.54017 > _gateway.domain: 49731+ [1au] PTR? 138.85.168.
21:33:29.679418 IP _gateway.domain > ubuntu.54017: 49731 NXDomain 0/1/1 (133)
```

Specifying only ens33 interface

9. TCPFLOW

- **tcpflow** is a program that captures data transmitted as part of TCP connections, and stores the data for protocol analysis or debugging
- It understands TCP sequence numbers and will correctly reconstruct data streams regardless of retransmissions or out-of-order delivery.
- However, it does not understand IP fragments; flows containing IP fragments will not be recorded properly.
- Syntax = *sudo tcpflow [OPTIONS]*

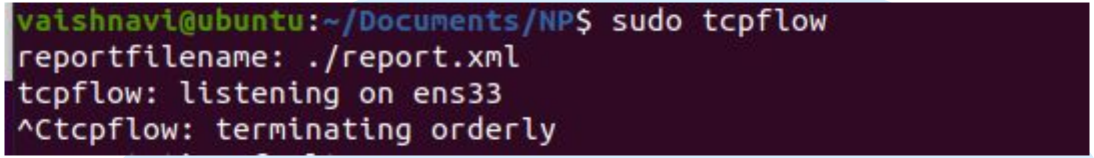
TCPFLOW Contd...

- tcpflow also includes an advanced plug-in system for decompressing compressed HTTP connections, undoing MIME encoding, or invoking third-party programs for post-processing and much more.
- There are many use cases for tcpflow which include to understand network packet flows and also supports for performing network forensics and divulge the contents of HTTP sessions.
- An XML report generated after running tcpflow, which contains information about the program

TCPFLOW Contd…

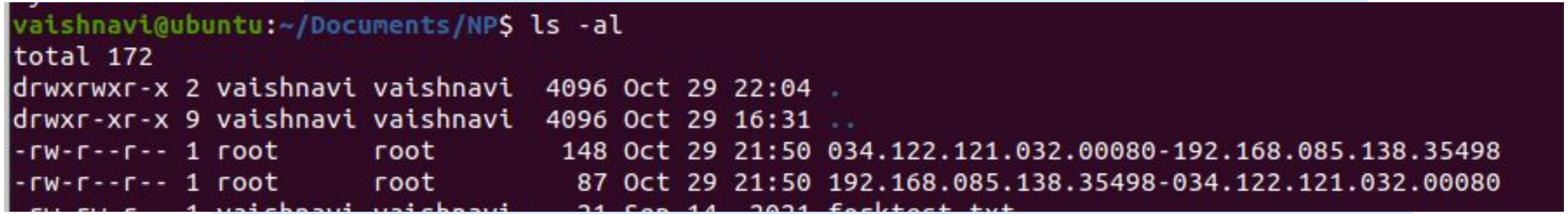
Option	Description
-b	Capture no more than given bytes per flow
-i	Captures for a specific interface
-s	Strip non-printables characters to “.”
-p	Set to no promiscuous mode
-c	Print the contents of packets to stdout
-g	Output each flow in alternating colors

TCPFLOW Contd...



```
vaishnavi@ubuntu:~/Documents/NP$ sudo tcpflow
reportfilename: ./report.xml
tcpflow: listening on ens33
^Ctcpflow: terminating orderly
```

Calling tcpflow which automatically stores in these files



```
vaishnavi@ubuntu:~/Documents/NP$ ls -al
total 172
drwxrwxr-x 2 vaishnavi vaishnavi  4096 Oct 29 22:04 .
drwxr-xr-x 9 vaishnavi vaishnavi  4096 Oct 29 16:31 ..
-rw-r--r-- 1 root      root        148 Oct 29 21:50 034.122.121.032.00080-192.168.085.138.35498
-rw-r--r-- 1 root      root         87 Oct 29 21:50 192.168.085.138.35498-034.122.121.032.00080
```

The saved output files, where the format is

sourceip.sourceport-destip.destport

TCPDUMP vs TCPFLOW

- tcpdump shows a summary of packets seen on the network, but usually doesn't store the data that's actually being transmitted.
- tcpflow reconstructs the actual data streams and stores each flow in a separate file for later analysis.
- tcpflow supports the same filtering expressions that programs like tcpdump.
- tcpflow can also rebuild flows from data captured with tcpdump -w.

10. TOP

- **top** is an abbreviation of **T**able **O**f **P**rocesses.
- It displays a real-time view of running processes in Linux and displays kernel-managed tasks.
- The command also provides a system information summary that shows resource utilization, including CPU and memory usage.
- Syntax = *top [OPTIONS]*

TOP Contd...

Columns headers:

1. **PID**: Process identifier of the task
2. **PR**: Process' priority. The lower the number, the higher the priority.
3. **VIRT**: Total virtual memory used by the task
4. **USER**: Owner of task's username
5. **%CPU**: CPU usage
6. **TIME+**: CPU Time (in 100th of a second)
7. **SHR**: Shared Memory size in KB , used by the task

TOP Contd...

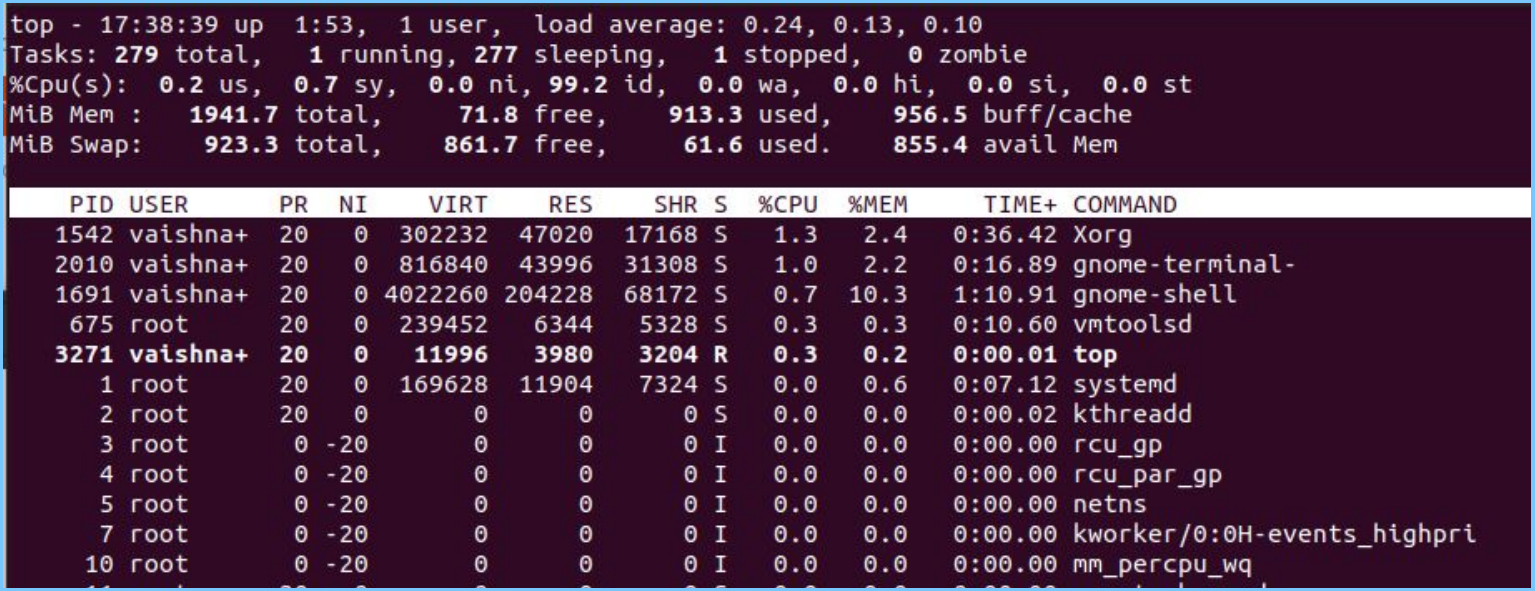
8. **NI**:Nice Value of task. -ve value = higher priority; +ve value= lower priority.
9. **%MEM**: Memory used by the task
10. **RES**: Physical RAM in KB used by the task
11. **COMMAND**: Command that is used to generate the process

The main difference between NI and PRis that PR is the real priority of a process as seen by the kernel, while NI is just a priority hint for the kernel.

TOP Contd...

Option	Description
-n	Exiting after “n” number of repetitions
-u	Print a specified user’s process
-d	Specifies the delay time of screen updates.
-c	Starts top with last closed state.
-p	Monitors specified process IDs
-s	Starts top in secure mode, even for root.

TOP Contd...



```
top - 17:38:39 up  1:53,  1 user,  load average: 0.24, 0.13, 0.10
Tasks: 279 total,   1 running, 277 sleeping,   1 stopped,   0 zombie
%Cpu(s):  0.2 us,  0.7 sy,  0.0 ni, 99.2 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :  1941.7 total,    71.8 free,   913.3 used,   956.5 buff/cache
MiB Swap:   923.3 total,   861.7 free,    61.6 used.   855.4 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   1542 vaishna+  20   0  302232  47020  17168 S   1.3   2.4   0:36.42 Xorg
   2010 vaishna+  20   0  816840  43996  31308 S   1.0   2.2   0:16.89 gnome-terminal-
   1691 vaishna+  20   0 4022260 204228  68172 S   0.7  10.3   1:10.91 gnome-shell
    675 root      20   0  239452   6344   5328 S   0.3   0.3   0:10.60 vmtoolsd
   3271 vaishna+  20   0   11996   3980   3204 R   0.3   0.2   0:00.01 top
      1 root      20   0  169628  11904   7324 S   0.0   0.6   0:07.12 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.02 kthreadd
      3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
      4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_par_gp
      5 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 netns
      7 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/0:0H-events_highpri
     10 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 mm_percpu_wq
```

Output of top command

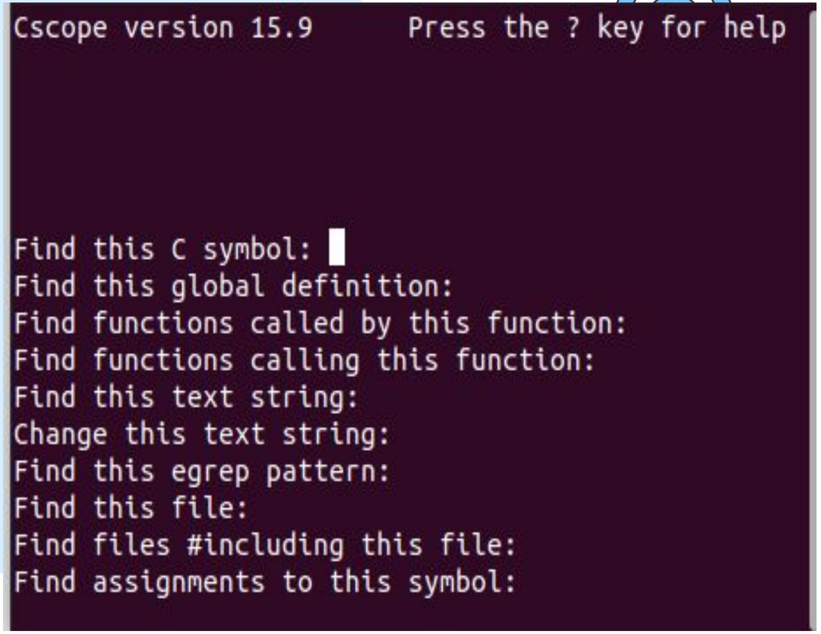
11. CSCOPE

- **cscope** is a Linux tool for browsing source code in a terminal environment.
- It was originally built to work with C code, but also works well with C++, Java, and some other languages.
- cscope support has been built into vim.
- It runs on all flavors of Unix, plus most monopoly-controlled operating systems.
- To exit cscope , use CTRL + D

CSCOPE Contd...

cscope allows searching code for:

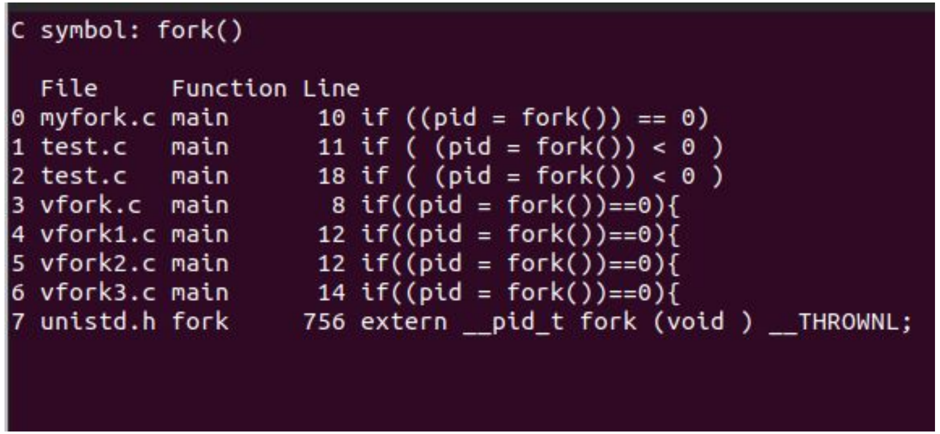
1. all references to a symbol
2. global definitions
3. functions called by a function
4. functions calling a function
5. text string
6. regular expression pattern
7. a file
8. files including a file



```
Cscope version 15.9       Press the ? key for help




Find this C symbol:
Find this global definition:
Find functions called by this function:
Find functions calling this function:
Find this text string:
Change this text string:
Find this egrep pattern:
Find this file:
Find files #including this file:
Find assignments to this symbol:
```

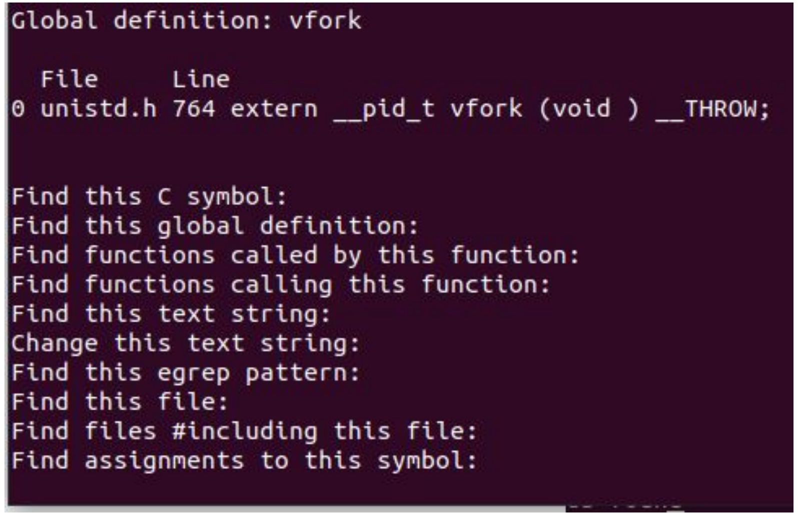
cscope -R command in a directory with .c files it

CSCOPE Contd...



```
C symbol: fork()

  File     Function Line
0 myfork.c main       10 if ((pid = fork()) == 0)
1 test.c   main       11 if ( (pid = fork()) < 0 )
2 test.c   main       18 if ( (pid = fork()) < 0 )
3 vfork.c  main        8 if((pid = fork())==0){
4 vfork1.c main       12 if((pid = fork())==0){
5 vfork2.c main       12 if((pid = fork())==0){
6 vfork3.c main       14 if((pid = fork())==0){
7 unistd.h fork      756 extern __pid_t fork (void ) __THROWNL;
```

Searching for the C symbol fork() using *cscope -R*



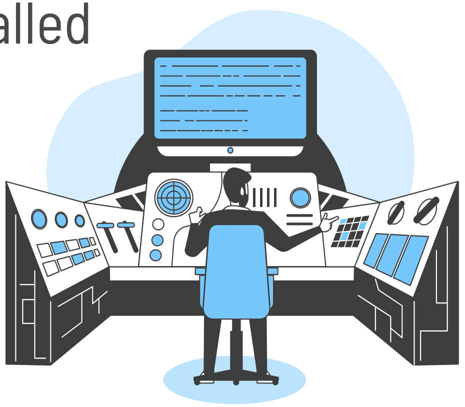
```
Global definition: vfork

  File     Line
0 unistd.h 764 extern __pid_t vfork (void ) __THROW;


Find this C symbol:
Find this global definition:
Find functions called by this function:
Find functions calling this function:
Find this text string:
Change this text string:
Find this egrep pattern:
Find this file:
Find files #including this file:
Find assignments to this symbol:
```

Searching for the Global definition of vfork using *cscope -R*

CSCOPE Contd...

For large projects, way to use cscope is:

1. Build the cscope database:
   a. The developer can often use find or other Unix tools to get the list of filenames needed to index into a file called cscope.files.
   b. The developer then builds a database using the command *cscope -b -q -k*
2. Second, the developer can now search those files using the command *cscope -d*. Often an index must be rebuilt whenever changes are made to files.



12. VI/EMACS

- vi is an abbreviation for **VI**sual (editor)
- emacs is an abbreviation for **E**ditor **MAC**ro**S**
- Both are powerful text editors.
- vi is the standard command-line text editor prebuilt in most Linux distributions. Its origins come from the Unix text editor for the command line, called ed.
- vim stands for “vi improved”, and is an implementation of vi with extra features that improve the user experience and increase the effectiveness of the text editor.

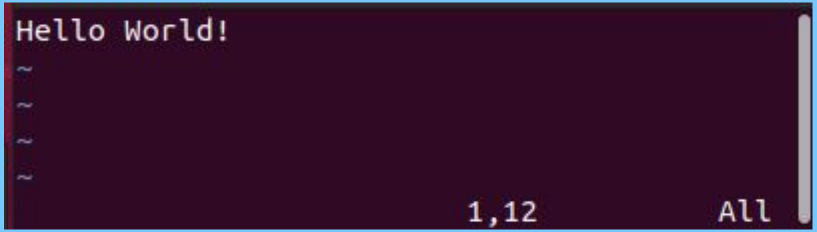
VI/EMACS Contd...

- vi categorises user interface into 3 modes of operations :
    - Command Mode - using ESC key
    - Insert Mode - using i,I,a,A, o, O characters
    - Escape Mode - using : character
- Syntax = *vi filename* or *vim filename*

VI/EMACS Contd...

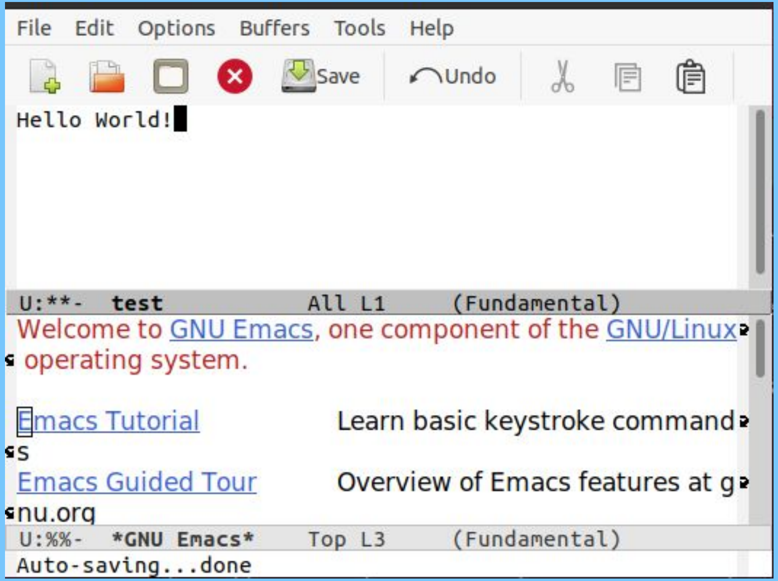
- emacs is modeless, unlike vi
- Its keyboard commands often start with the CTRL key or the Meta key, so that the system can distinguish actual edits from commands.
- emacs is said to resembles editors like Microsoft Word and Google Docs more than vim because of its modelessness, and this fact may make it easier to get used to than vim.
- Syntax = *emacs filename*

VI/EMACS Contd...



```
Hello World!
~
~
~
~
                        1,12           All
```

Editing in vi



```
Hello World!

U:**-  test            All L1      (Fundamental)
Welcome to GNU Emacs, one component of the GNU/Linux
operating system.

Emacs Tutorial          Learn basic keystroke commands
Emacs Guided Tour       Overview of Emacs features at gnu.org
U:%%-  *GNU Emacs*     Top L3      (Fundamental)
Auto-saving...done
```

Editing in emacs

The EDITOR WAR

The editor war is the rivalry between users of the emacs and vi. This rivalry has become a lasting part of hacker culture and the free software community.

The vi versus emacs debate was one of the original "holy wars" conducted on **Usenet** groups,with many flame wars fought between those insisting that their editor of choice is the paragon of editing perfection, and insulting the other, since at least 1985.

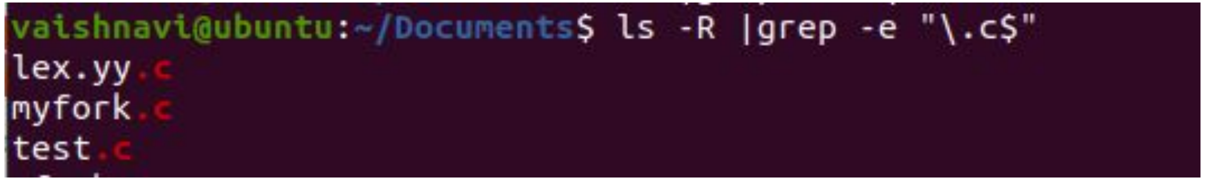
13. GREP

- **grep** is an abbreviation for it’s utility, that is **G**lobal **R**egular **E**xpression **P**rint.
- Given one or more patterns, grep searches input files for matches to the patterns.
- Syntax = *grep [OPTIONS] PATTERNS [FILES]*

GREP Contd...

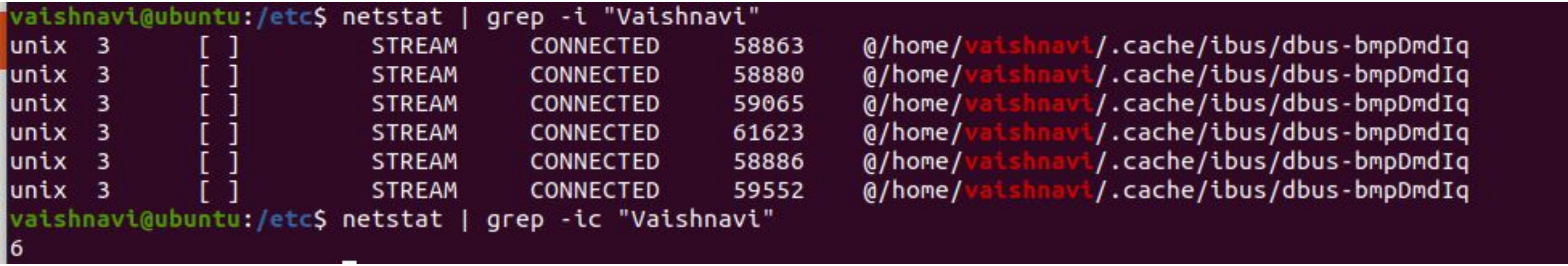
Option	Description
-i	Ignores case distinctions
-c	Prints a count of matching lines
-f	Takes patterns from file,one per line.
-v	Prints non-matching lines.
-l	Prints list of a filenames only.

GREP Contd...



```
vaishnavi@ubuntu:~/Documents$ ls -R |grep -e "\.c$"
lex.yy.c
myfork.c
test.c
```

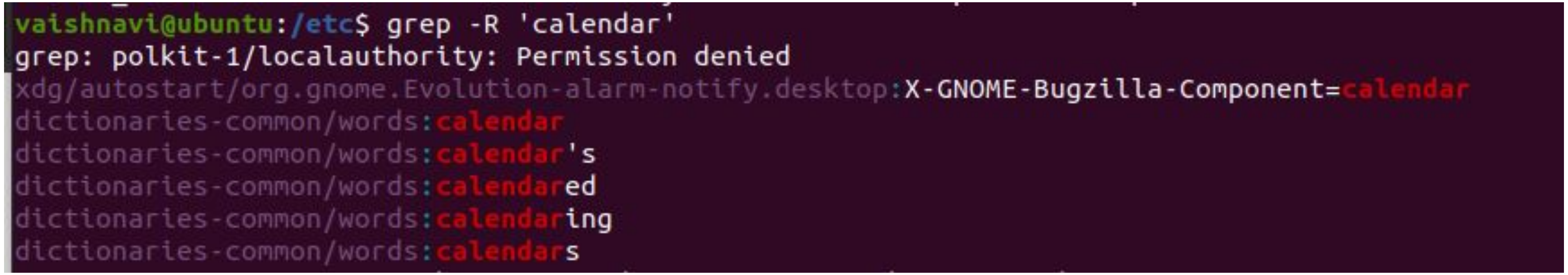
Printing all .c files using regex



```
vaishnavi@ubuntu:/etc$ netstat | grep -i "Vaishnavi"
unix  3      [ ]         STREAM     CONNECTED     58863    @/home/vaishnavi/.cache/ibus/dbus-bmpDmdIq
unix  3      [ ]         STREAM     CONNECTED     58880    @/home/vaishnavi/.cache/ibus/dbus-bmpDmdIq
unix  3      [ ]         STREAM     CONNECTED     59065    @/home/vaishnavi/.cache/ibus/dbus-bmpDmdIq
unix  3      [ ]         STREAM     CONNECTED     61623    @/home/vaishnavi/.cache/ibus/dbus-bmpDmdIq
unix  3      [ ]         STREAM     CONNECTED     58886    @/home/vaishnavi/.cache/ibus/dbus-bmpDmdIq
unix  3      [ ]         STREAM     CONNECTED     59552    @/home/vaishnavi/.cache/ibus/dbus-bmpDmdIq
vaishnavi@ubuntu:/etc$ netstat | grep -ic "Vaishnavi"
6
```

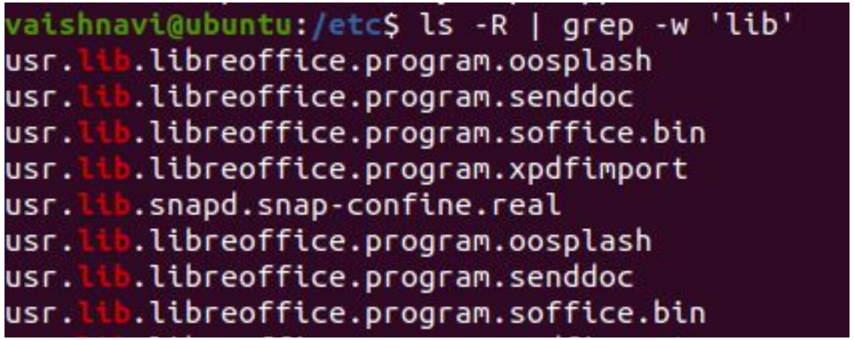
Printing all programs with path having “Vaishnavi ” in them, using case insensitivity and count

GREP Contd...



```
vaishnavi@ubuntu:/etc$ grep -R 'calendar'
grep: polkit-1/localauthority: Permission denied
xdg/autostart/org.gnome.Evolution-alarm-notify.desktop:X-GNOME-Bugzilla-Component=calendar
dictionaries-common/words:calendar
dictionaries-common/words:calendar's
dictionaries-common/words:calendared
dictionaries-common/words:calendaring
dictionaries-common/words:calendars
```

Recursively searching the /etc directory for the word calendar



```
vaishnavi@ubuntu:/etc$ ls -R | grep -w 'lib'
usr.lib.libreoffice.program.oosplash
usr.lib.libreoffice.program.senddoc
usr.lib.libreoffice.program.soffice.bin
usr.lib.libreoffice.program.xpdfimport
usr.lib.snapd.snap-confine.real
usr.lib.libreoffice.program.oosplash
usr.lib.libreoffice.program.senddoc
usr.lib.libreoffice.program.soffice.bin
```

Printing matches of whole word 'lib' in the /etc directory

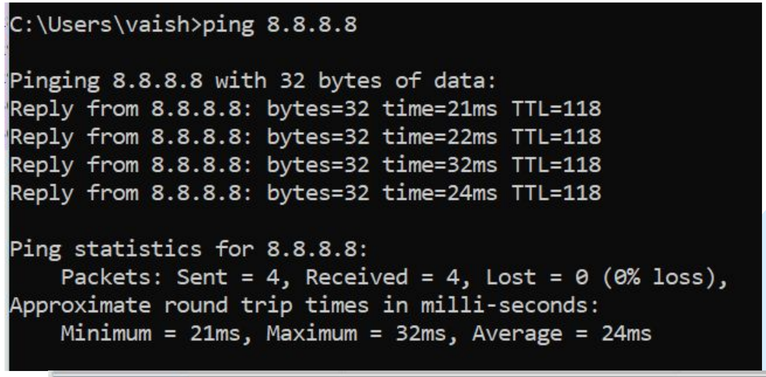
14. PING

- **ping** is an abbreviation for **P**acket **I**nter**N**et **G**roper.
- It is used to troubleshoot networking and connectivity.
- The command sends ICMP ECHO_REQUEST datagram to elicit an ICMP ECHO_RESPONSE from a host or gateway.
- Syntax = *ping [OPTIONS] destinationaddress*

PING Contd...

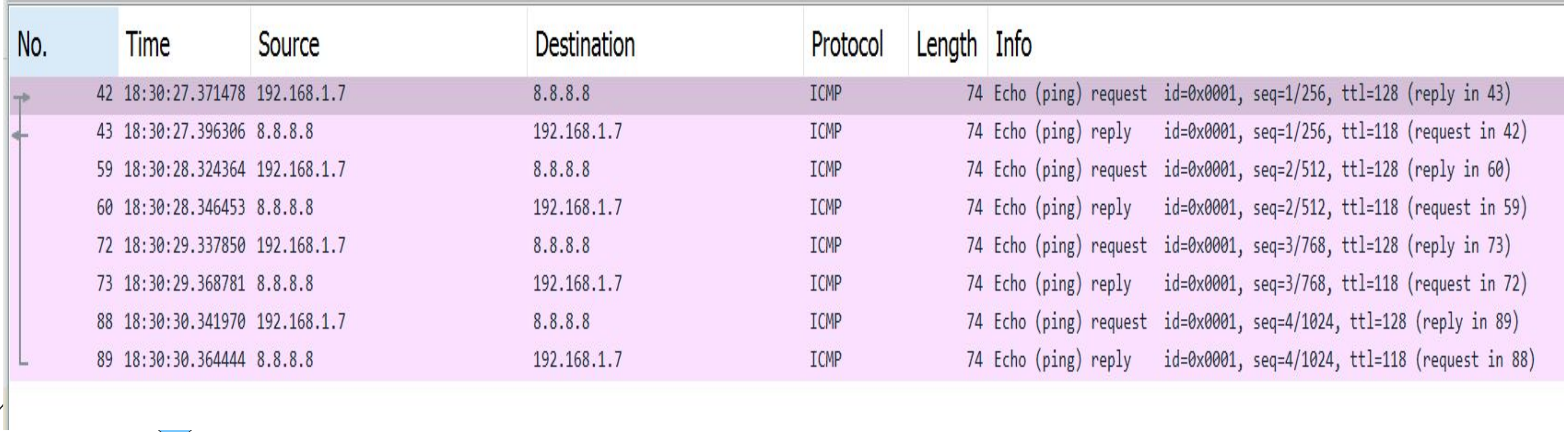
Option	Description
-c	Controls the number of packets to send to the destination
-4	Uses IPv4 only
-6	Uses IPv6 Only
-i	Waiting interval(in seconds) between sending packets
-q	Display quiet output
-s	Changes packet size for sending request

Ping Contd...



```
C:\Users\vaish>ping 8.8.8.8

Pinging 8.8.8.8 with 32 bytes of data:
Reply from 8.8.8.8: bytes=32 time=21ms TTL=118
Reply from 8.8.8.8: bytes=32 time=22ms TTL=118
Reply from 8.8.8.8: bytes=32 time=32ms TTL=118
Reply from 8.8.8.8: bytes=32 time=24ms TTL=118

Ping statistics for 8.8.8.8:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 21ms, Maximum = 32ms, Average = 24ms
```

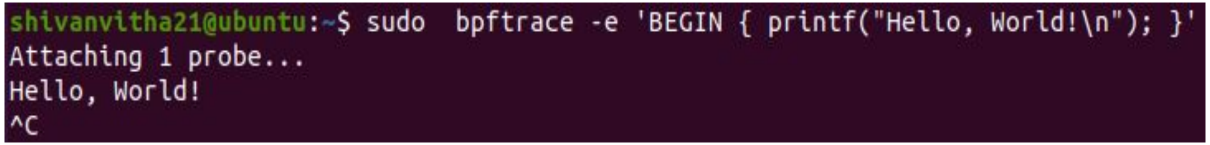
ping-ing the google server, and the wireshark depicting **ping**'s working



No.	Time	Source	Destination	Protocol	Length	Info
42	18:30:27.371478	192.168.1.7	8.8.8.8	ICMP	74	Echo (ping) request id=0x0001, seq=1/256, ttl=128 (reply in 43)
43	18:30:27.396306	8.8.8.8	192.168.1.7	ICMP	74	Echo (ping) reply id=0x0001, seq=1/256, ttl=118 (request in 42)
59	18:30:28.324364	192.168.1.7	8.8.8.8	ICMP	74	Echo (ping) request id=0x0001, seq=2/512, ttl=128 (reply in 60)
60	18:30:28.346453	8.8.8.8	192.168.1.7	ICMP	74	Echo (ping) reply id=0x0001, seq=2/512, ttl=118 (request in 59)
72	18:30:29.337850	192.168.1.7	8.8.8.8	ICMP	74	Echo (ping) request id=0x0001, seq=3/768, ttl=128 (reply in 73)
73	18:30:29.368781	8.8.8.8	192.168.1.7	ICMP	74	Echo (ping) reply id=0x0001, seq=3/768, ttl=118 (request in 72)
88	18:30:30.341970	192.168.1.7	8.8.8.8	ICMP	74	Echo (ping) request id=0x0001, seq=4/1024, ttl=128 (reply in 89)
89	18:30:30.364444	8.8.8.8	192.168.1.7	ICMP	74	Echo (ping) reply id=0x0001, seq=4/1024, ttl=118 (request in 88)

15. BPFTRACE

- **BPFtrace** is a new high-level tracing language for **eBPF** (enhanced Berkeley Packet Filter) that is made available in kernel versions (4.x)+ inspired by C and awk
- **LLVM** is used as the backend for compiling scripts to BPF-bytecode
- **BCC** acts as the interface between Linux BPF system and BPFtrace



```
shivanvitha21@ubuntu:~$ sudo  bpftrace -e 'BEGIN { printf("Hello, World!\n"); }'
Attaching 1 probe...
Hello, World!
^C
```

BPFTRACE Contd...

Linux tracing systems can be classified into,

Data sources

Where the tracing data comes from
Ex: kprobe, uprobe, tracepoint

Mechanisms for Data Collection

Means by which data collection is done
Ex: eBPF

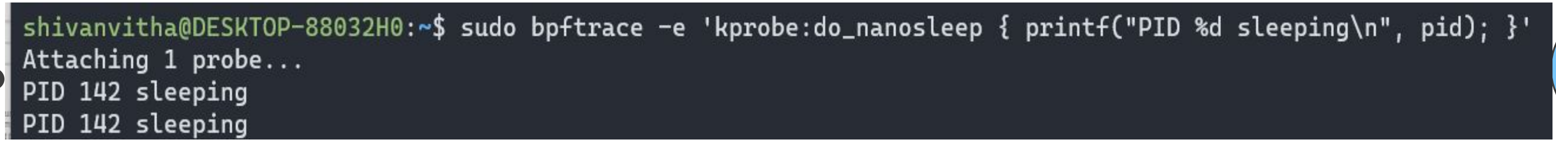
Tracing Frontends

Tool used to collect and analyse the data
Ex: BPFtrace

BPFTRACE Contd...

Probe: It is an instrumentation point that generates events that can execute bpftrace programs

- **Kprobe:** Attaches a BPFtrace script to a kernel function i.e., it creates and manages probe points in kernel code.
  Ex: Trace the processes that call sleep



```
shivanvitha@DESKTOP-88032H0:~$ sudo bpftrace -e 'kprobe:do_nanosleep { printf("PID %d sleeping\n", pid); }'
Attaching 1 probe...
PID 142 sleeping
PID 142 sleeping
```

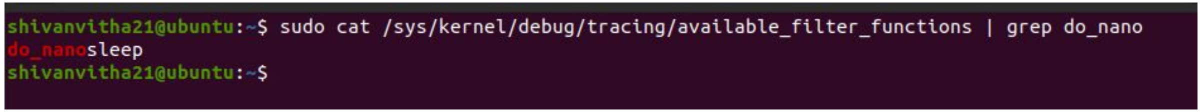
BPFTRACE Contd...

A list of kprobes can be found in

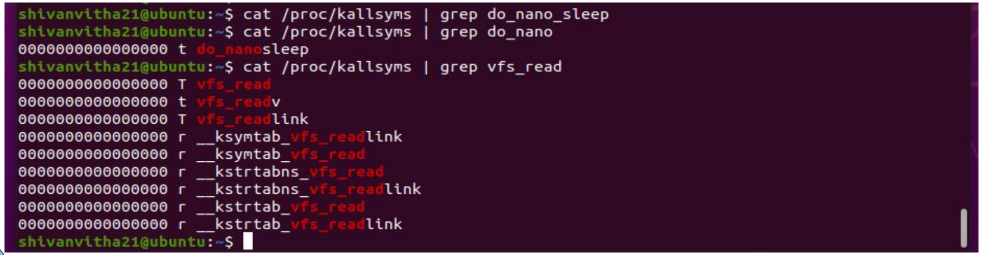
/proc/kallsyms

OR

/sys/kernel/debug/tracing/available_filter_functions



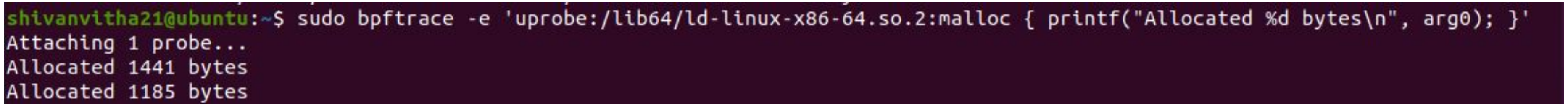
```
shivanvitha21@ubuntu:~$ sudo cat /sys/kernel/debug/tracing/available_filter_functions | grep do_nano
do_nanosleep
shivanvitha21@ubuntu:~$
```



```
shivanvitha21@ubuntu:~$ cat /proc/kallsyms | grep do_nano_sleep
shivanvitha21@ubuntu:~$ cat /proc/kallsyms | grep do_nano
0000000000000000 t do_nanosleep
shivanvitha21@ubuntu:~$ cat /proc/kallsyms | grep vfs_read
0000000000000000 T vfs_read
0000000000000000 t vfs_readv
0000000000000000 T vfs_readlink
0000000000000000 r __ksymtab_vfs_readlink
0000000000000000 r __ksymtab_vfs_read
0000000000000000 r __kstrtabns_vfs_read
0000000000000000 r __kstrtabns_vfs_readlink
0000000000000000 r __kstrtab_vfs_read
0000000000000000 r __kstrtab_vfs_readlink
shivanvitha21@ubuntu:~$
```

BPFTRACE Contd...

- **Uprobe:** Attaches a BPFtrace script to a userland function
  Ex: Tracing malloc



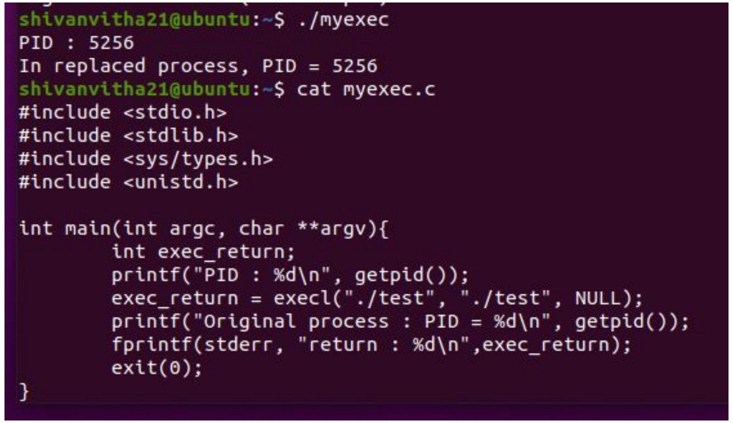
```
shivanvitha21@ubuntu:~$ sudo bpftrace -e 'uprobe:/lib64/ld-linux-x86-64.so.2:malloc { printf("Allocated %d bytes\n", arg0); }'
Attaching 1 probe...
Allocated 1441 bytes
Allocated 1185 bytes
```

- **Tracepoint:** Attaches a BPFtrace script to a statically defined tracepoint in the kernel. These are more stable than kprobes between kernel versions

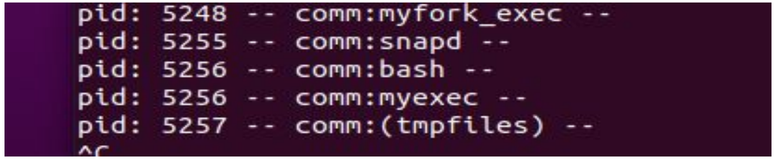
BPFTRACE Contd...

Ex: Trace the processes running exec family

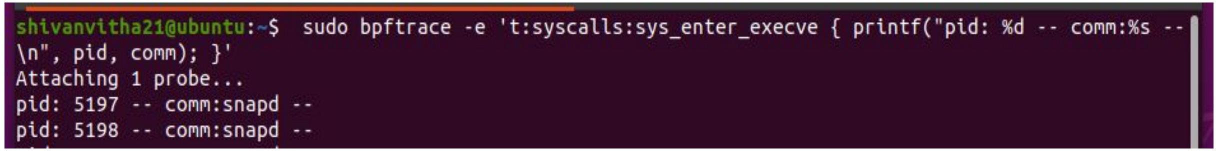
Custom program running exec is shown



```
shivanvitha21@ubuntu:~$ ./myexec
PID : 5256
In replaced process, PID = 5256
shivanvitha21@ubuntu:~$ cat myexec.c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>

int main(int argc, char **argv){
        int exec_return;
        printf("PID : %d\n", getpid());
        exec_return = execl("./test", "./test", NULL);
        printf("Original process : PID = %d\n", getpid());
        fprintf(stderr, "return : %d\n",exec_return);
        exit(0);
}
```



```
pid: 5248 -- comm:myfork_exec --
pid: 5255 -- comm:snapd --
pid: 5256 -- comm:bash --
pid: 5256 -- comm:myexec --
pid: 5257 -- comm:(tmpfiles) --
^C
```

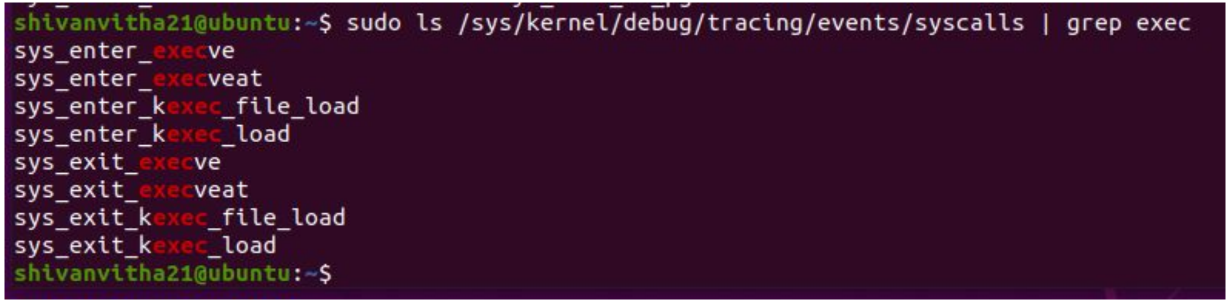


```
shivanvitha21@ubuntu:~$  sudo bpftrace -e 't:syscalls:sys_enter_execve { printf("pid: %d -- comm:%s --
\n", pid, comm); }'
Attaching 1 probe...
pid: 5197 -- comm:snapd --
pid: 5198 -- comm:snapd --
```

BPFTRACE Contd...

A list of tracepoints can be found in,
/sys/kernel/debug/tracing/events

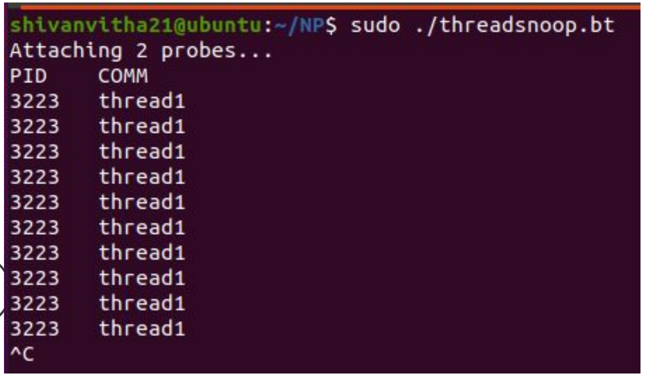
“syscalls:sys_enter_execve” is found as shown below:



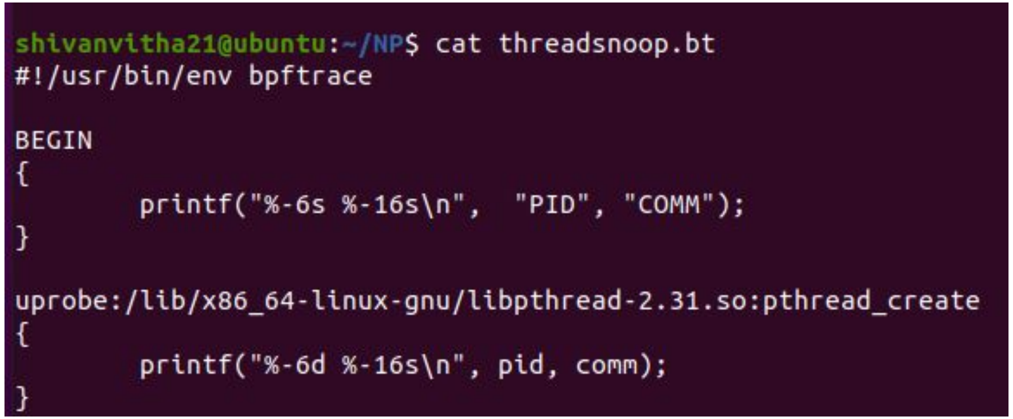
```
shivanvitha21@ubuntu:~$ sudo ls /sys/kernel/debug/tracing/events/syscalls | grep exec
sys_enter_execve
sys_enter_execveat
sys_enter_kexec_file_load
sys_enter_kexec_load
sys_exit_execve
sys_exit_execveat
sys_exit_kexec_file_load
sys_exit_kexec_load
shivanvitha21@ubuntu:~$
```

BPFTRACE Contd...

Ex: To list creation of new threads with a bpftrace script file

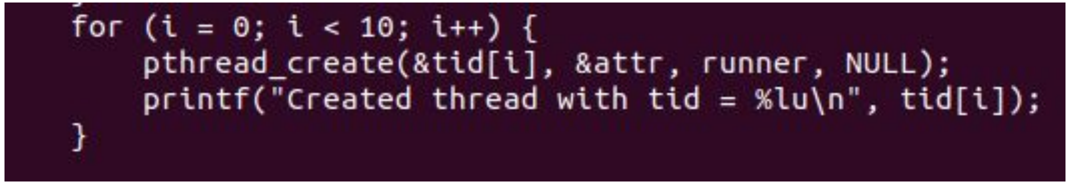


```
shivanvitha21@ubuntu:~/NP$ sudo ./threadsnoop.bt
Attaching 2 probes...
PID    COMM
3223   thread1
3223   thread1
3223   thread1
3223   thread1
3223   thread1
3223   thread1
3223   thread1
3223   thread1
3223   thread1
3223   thread1
^C
```



```
shivanvitha21@ubuntu:~/NP$ cat threadsnoop.bt
#!/usr/bin/env bpftrace

BEGIN
{
        printf("%-6s %-16s\n",  "PID", "COMM");
}

uprobe:/lib/x86_64-linux-gnu/libpthread-2.31.so:pthread_create
{
        printf("%-6d %-16s\n", pid, comm);
}
```

...



```
    for (i = 0; i < 10; i++) {
        pthread_create(&tid[i], &attr, runner, NULL);
        printf("Created thread with tid = %lu\n", tid[i]);
    }
```

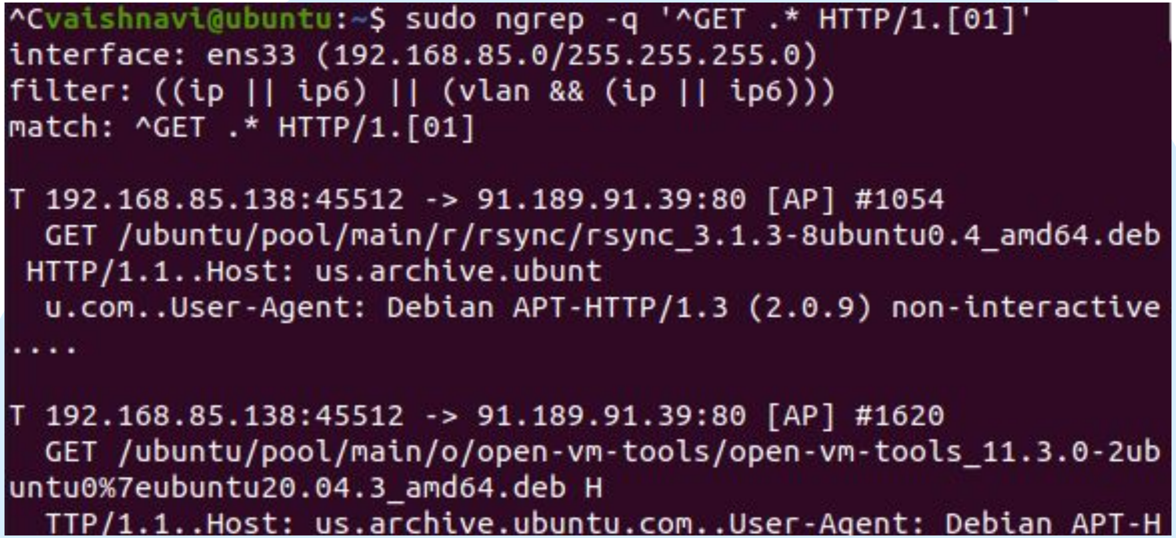
16. NGREP

- **ngrep** is a abbreviation for **N**etwork **GREP** data packet analyzer that runs on a command line interface.
- It is a grep-like tool applied to the network layer
- It allows you to specify an extended regular or hexadecimal expression to match against data payloads
- It works with various types of protocols
- It operates in the same fashion as tcpdump
- Syntax = *ngrep [OPTIONS]*

NGREP Contd...

Option	Description
-v	Display packets that don't match
-N	Prints sub-protocol number along with single-character identifier
-p	Set to no promiscuous mode
-i	Ignore case for the regex expression
-q	Display quiet output
-x	Print packet contents as hexadecimal as well as ASCII.

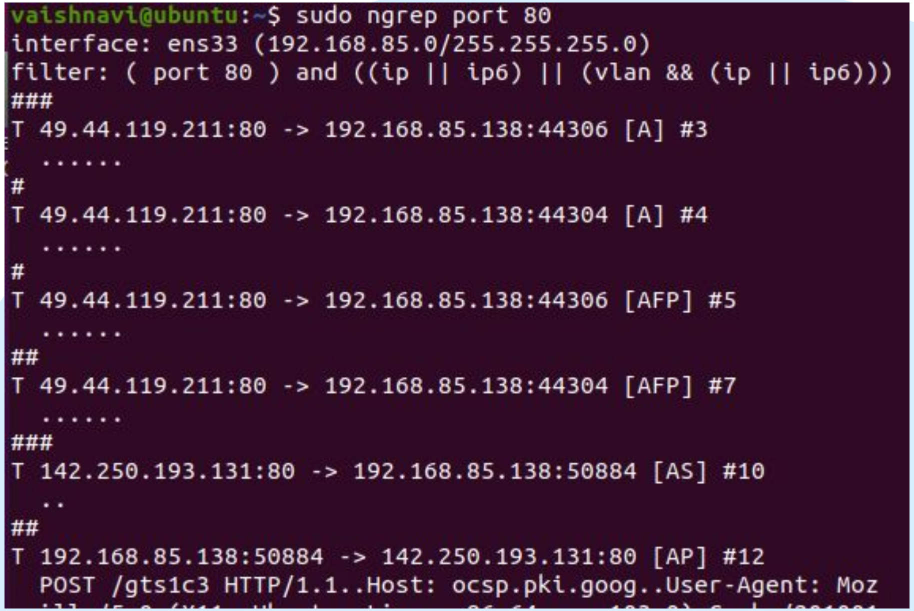
NGREP Contd...



```
^Cvaishnavi@ubuntu:~$ sudo ngrep -q '^GET .* HTTP/1.[01]'
interface: ens33 (192.168.85.0/255.255.255.0)
filter: ((ip || ip6) || (vlan && (ip || ip6)))
match: ^GET .* HTTP/1.[01]

T 192.168.85.138:45512 -> 91.189.91.39:80 [AP] #1054
  GET /ubuntu/pool/main/r/rsync/rsync_3.1.3-8ubuntu0.4_amd64.deb
 HTTP/1.1..Host: us.archive.ubunt
  u.com..User-Agent: Debian APT-HTTP/1.3 (2.0.9) non-interactive
....

T 192.168.85.138:45512 -> 91.189.91.39:80 [AP] #1620
  GET /ubuntu/pool/main/o/open-vm-tools/open-vm-tools_11.3.0-2ub
untu0%7eubuntu20.04.3_amd64.deb H
  TTP/1.1..Host: us.archive.ubuntu.com..User-Agent: Debian APT-H
```

The following command is monitoring which files my browser is requesting

NGREP Contd...



```
vaishnavi@ubuntu:~$ sudo ngrep port 80
interface: ens33 (192.168.85.0/255.255.255.0)
filter: ( port 80 ) and ((ip || ip6) || (vlan && (ip || ip6)))
###
T 49.44.119.211:80 -> 192.168.85.138:44306 [A] #3
  ......
#
T 49.44.119.211:80 -> 192.168.85.138:44304 [A] #4
  ......
#
T 49.44.119.211:80 -> 192.168.85.138:44306 [AFP] #5
  ......
##
T 49.44.119.211:80 -> 192.168.85.138:44304 [AFP] #7
  ......
###
T 142.250.193.131:80 -> 192.168.85.138:50884 [AS] #10
  ..
##
T 192.168.85.138:50884 -> 142.250.193.131:80 [AP] #12
  POST /gts1c3 HTTP/1.1..Host: ocsp.pki.goog..User-Agent: Moz
```

The following command is monitoring which files are received on port on 80

Other Honorable Mentions

ipconfig/ifconfig - Internet Protocol configuration/ Interface configuration

route - Network route tables

nslookup - Name Server Lookup

nmap - Network Mapper

dig - Domain Information Groper

REFERENCES

- https://www.wireshark.org/docs/wsug_html_chunked/ChUseMainWindowSection.html
- https://man7.org/linux/man-pages/man8/netstat.8.html
- https://www.redhat.com/sysadmin/ss-command
- https://gcc.gnu.org/onlinedocs/gcc/G_002b_002b-and-GCC.html
- https://man7.org/linux/man-pages/man1/gdb.1.html
- http://www.unknownroad.com/rtfm/gdbtut/gdbsegfault.html
- https://linux.die.net/man/1/make
- https://jvns.ca/blog/2017/07/05/linux-tracing-systems/#kprobes
- https://opensource.com/article/19/8/introduction-bpftrace
- https://github.com/iovisor/bpftrace
- https://github.com/iovisor/bpftrace/blob/master/docs/reference_guide.md
- https://cscope.sourceforge.net/
- https://linux.die.net/man/1/cscope
- https://www.tcpdump.org/manpages/tcpdump.1.html
- https://man.openbsd.org/tcpdump
- https://opensource.com/article/18/10/introduction-tcpdump
- https://www.kali.org/tools/tcpflow/
- https://linux.die.net/man/1/tcpflow
- https://www.tecmint.com/tcpflow-analyze-debug-network-traffic-in-linux/
- https://www.cs.colostate.edu/helpdocs/vi.html
- https://man7.org/linux/man-pages/man1/vi.1p.html
- https://opensource.com/resources/what-emacs
- https://www.redhat.com/sysadmin/beginners-guide-emacs
- https://linux.die.net/man/1/emacs
- https://www.linux.com/news/emacs-vs-vi-endless-geek-holy-war/
- https://man7.org/linux/man-pages/man1/grep.1.html
- https://linux.die.net/man/8/ping
- https://docs.oracle.com/cd/E88353_01/html/E72487/ping-8.html
- https://linux.die.net/man/8/ngrep

Thank You!

Do you have any questions?