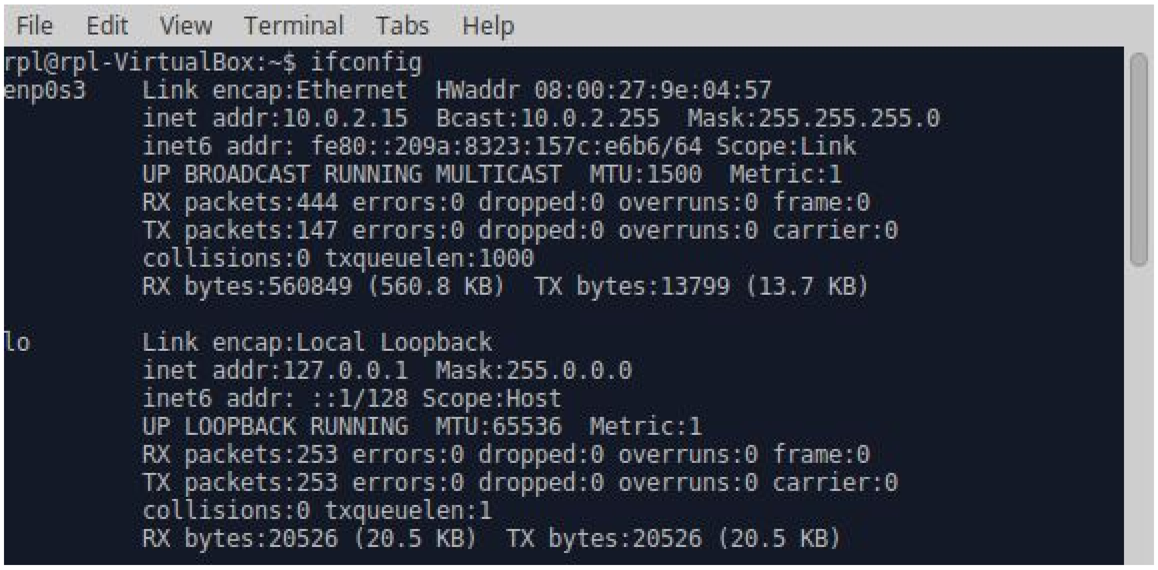
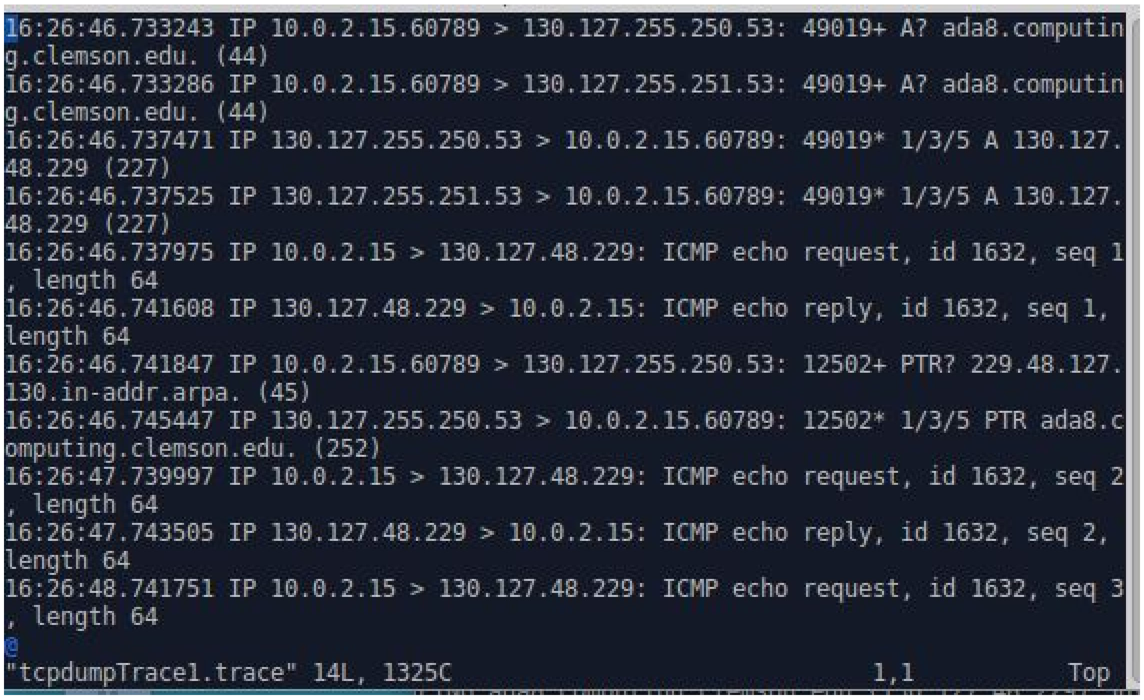
- Q1.1:
    - Network interface name: enp0s3
    - IPV4 address: fe80::209a:8323:157c:e6b6/64

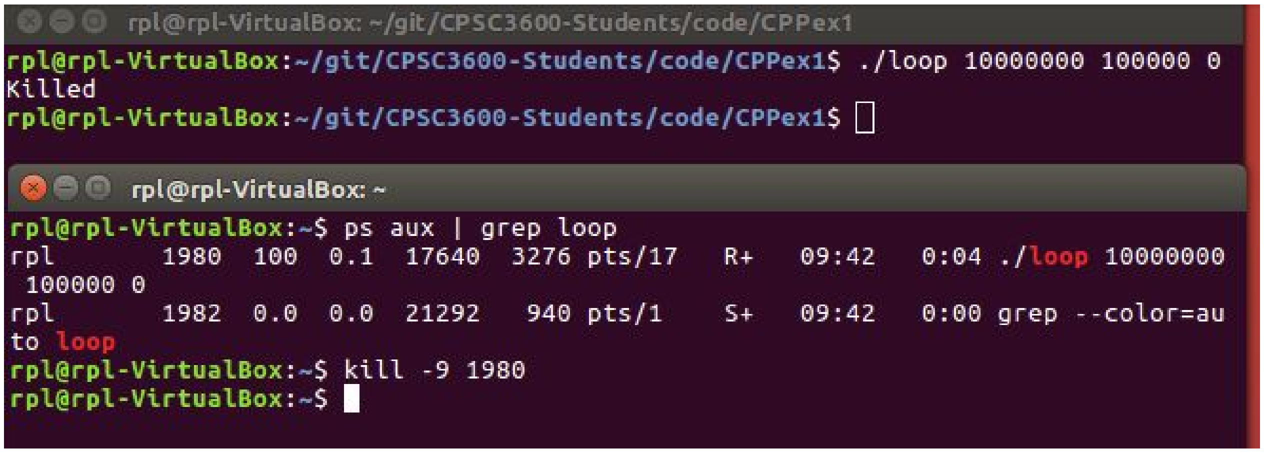


```
rpl@rpl-VirtualBox:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:9e:04:57
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::209a:8323:157c:e6b6/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:444 errors:0 dropped:0 overruns:0 frame:0
          TX packets:147 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:560849 (560.8 KB)  TX bytes:13799 (13.7 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:253 errors:0 dropped:0 overruns:0 frame:0
          TX packets:253 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:20526 (20.5 KB)  TX bytes:20526 (20.5 KB)
```



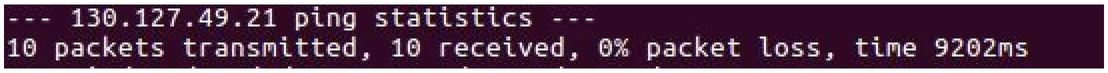
```
16:26:46.733243 IP 10.0.2.15.60789 > 130.127.255.250.53: 49019+ A? ada8.computin
g.clemson.edu. (44)
16:26:46.733286 IP 10.0.2.15.60789 > 130.127.255.251.53: 49019+ A? ada8.computin
g.clemson.edu. (44)
16:26:46.737471 IP 130.127.255.250.53 > 10.0.2.15.60789: 49019* 1/3/5 A 130.127.
48.229 (227)
16:26:46.737525 IP 130.127.255.251.53 > 10.0.2.15.60789: 49019* 1/3/5 A 130.127.
48.229 (227)
16:26:46.737975 IP 10.0.2.15 > 130.127.48.229: ICMP echo request, id 1632, seq 1
, length 64
16:26:46.741608 IP 130.127.48.229 > 10.0.2.15: ICMP echo reply, id 1632, seq 1,
length 64
16:26:46.741847 IP 10.0.2.15.60789 > 130.127.255.250.53: 12502+ PTR? 229.48.127.
130.in-addr.arpa. (45)
16:26:46.745447 IP 130.127.255.250.53 > 10.0.2.15.60789: 12502* 1/3/5 PTR ada8.c
omputing.clemson.edu. (252)
16:26:47.739997 IP 10.0.2.15 > 130.127.48.229: ICMP echo request, id 1632, seq 2
, length 64
16:26:47.743505 IP 130.127.48.229 > 10.0.2.15: ICMP echo reply, id 1632, seq 2,
length 64
16:26:48.741751 IP 10.0.2.15 > 130.127.48.229: ICMP echo request, id 1632, seq 3
, length 64
@
"tcpdumpTrace1.trace" 14L, 1325C                                1,1          Top
```

- Q1.2:
    - issue ‘man 7 signal’ in the terminal
- Q1.3:

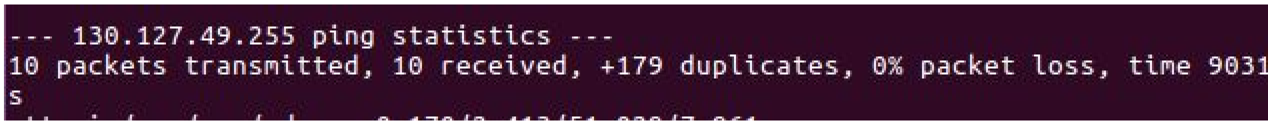


```
rpl@rpl-VirtualBox:~/git/CPSC3600-Students/code/CPPex1$ ./loop 10000000 100000 0
Killed
rpl@rpl-VirtualBox:~/git/CPSC3600-Students/code/CPPex1$
```

```
rpl@rpl-VirtualBox:~$ ps aux | grep loop
rpl       1980  100  0.1  17640  3276 pts/17   R+   09:42   0:04 ./loop 10000000
 100000 0
rpl       1982  0.0  0.0  21292   940 pts/1    S+   09:42   0:00 grep --color=au
to loop
rpl@rpl-VirtualBox:~$ kill -9 1980
rpl@rpl-VirtualBox:~$
```

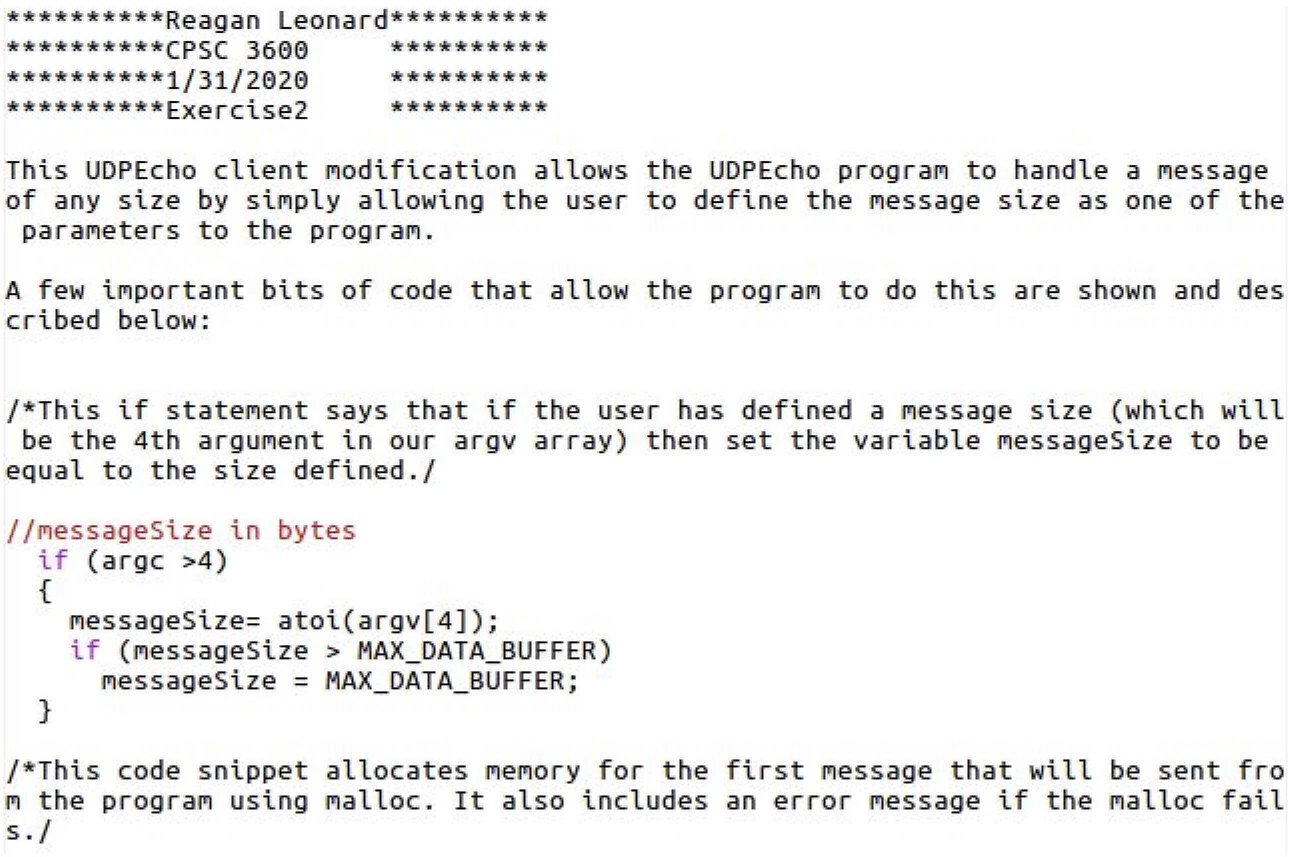
- Q2.1:
    - IPV4: 130.127.49.21
        - 

```
--- 130.127.49.21 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9202ms
```

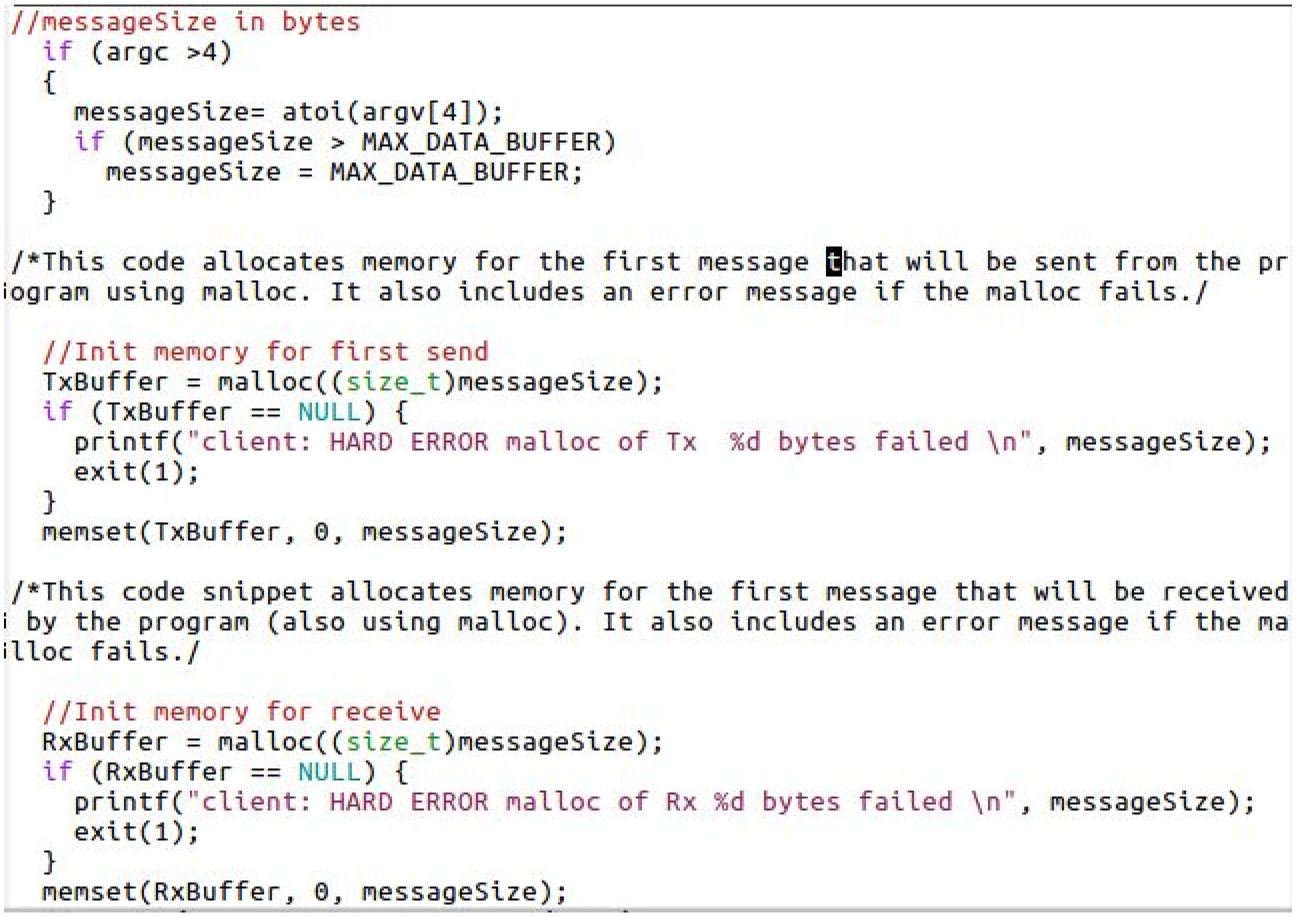
    - Local broadcast: 130.127.49.255
        - 

```
--- 130.127.49.255 ping statistics ---
10 packets transmitted, 10 received, +179 duplicates, 0% packet loss, time 9031
s
```

        - More duplicates
    - IPV6: fe80::42b0:34ff:fef9:2aa0/64
        - Failed to recognize host
    - IPV6: 2620:103:a000:401:42b0:34ff:fef9:2aa0/64
        - Failed to recognize host
- Q2.2:
    - Class A
    - 130.127.49.21
    - 65535
- Q2.3:
    - 198.21.240.166
- Q3.1:



```
**********Reagan Leonard**********
**********CPSC 3600     **********
**********1/31/2020     **********
**********Exercise2     **********

This UDPEcho client modification allows the UDPEcho program to handle a message
of any size by simply allowing the user to define the message size as one of the
 parameters to the program.

A few important bits of code that allow the program to do this are shown and des
cribed below:


/*This if statement says that if the user has defined a message size (which will
 be the 4th argument in our argv array) then set the variable messageSize to be
equal to the size defined./

//messageSize in bytes
  if (argc >4)
  {
    messageSize= atoi(argv[4]);
    if (messageSize > MAX_DATA_BUFFER)
      messageSize = MAX_DATA_BUFFER;
  }

/*This code snippet allocates memory for the first message that will be sent fro
m the program using malloc. It also includes an error message if the malloc fail
s./
```



```
//messageSize in bytes
  if (argc >4)
  {
    messageSize= atoi(argv[4]);
    if (messageSize > MAX_DATA_BUFFER)
      messageSize = MAX_DATA_BUFFER;
  }

/*This code allocates memory for the first message that will be sent from the pr
ogram using malloc. It also includes an error message if the malloc fails./

  //Init memory for first send
  TxBuffer = malloc((size_t)messageSize);
  if (TxBuffer == NULL) {
    printf("client: HARD ERROR malloc of Tx  %d bytes failed \n", messageSize);
    exit(1);
  }
  memset(TxBuffer, 0, messageSize);

/*This code snippet allocates memory for the first message that will be received
 by the program (also using malloc). It also includes an error message if the ma
lloc fails./

  //Init memory for receive
  RxBuffer = malloc((size_t)messageSize);
  if (RxBuffer == NULL) {
    printf("client: HARD ERROR malloc of Rx %d bytes failed \n", messageSize);
    exit(1);
  }
  memset(RxBuffer, 0, messageSize);
```