

System Programing 4

PING.C

This program will send an ICMP (Internet Control Message Protocol) echo request packet to the specified host and wait for an ICMP echo reply.

The "calculate_checksum" function is used to compute the checksum for the ICMP packet. The checksum is used to verify the integrity of the packet.

The "main" function first checks if the correct number of arguments (a hostname or IP address) has been provided. If not, it prints a usage message and exits.

The program then attempts to resolve the hostname provided as an argument to an IP address using the "getHostByName" function.

Next, a raw socket is created with the "socket" function. This type of socket allows the program to send and receive raw data at the IP level. The socket's Time-To-Live (TTL) value is then set to 255 using the "setsockopt" function. The TTL value determines the maximum number of hops that a packet can make before it is discarded.

The program then constructs an ICMP packet containing an echo request and some data. It calculates the checksum of the packet and sends it to the destination using the "sendto" function.

The program then waits for a reply using the "recvfrom" function and calculates the round-trip time taken for the packet to be sent and received. It then prints this time and some other information about the reply packet. The program then waits for 1 second before sending another echo request.

This process continues until the program is interrupted.

NEWPING.C

This program is similar to the ping.c program, but with some additional features. It also sends ICMP echo request packets to a specified host and waits for an ICMP echo reply.

In addition to the raw socket used to send and receive the ICMP packets, this program also creates a TCP socket. It then uses the TCP socket to connect to a "watchdog" program running on the local host (IP address "127.0.0.1") on port 3000.

The program then forks, creating a child process. The child process executes the "watchdog" program using the "exevp" function. The parent process waits for 2 seconds, then sends the hostname or IP address of the destination to the watchdog program using the TCP socket.

The program then constructs an ICMP packet and sends it to the destination in the same way as the first program. However, after sending the packet, the program waits for a response from the watchdog program using the TCP socket. If the watchdog program sends a message, the program knows that the watchdog has received a reply to the ICMP echo request and exits. If the watchdog program does not send a message within a certain timeout period, the program sends another ICMP echo request.

This process continues until the watchdog program sends a message, indicating that it has received a reply.

Watchdog.c

This is the "watchdog" program that is run by the second ping program. It creates a TCP socket and listens for incoming connections on port 3000. When a client connects, it receives a message from the client indicating that the client is ready to start sending ICMP echo request packets.

The watchdog program then enters a loop where it waits for messages from the client. If it receives a message containing the string "ping", it sets the socket to non-blocking mode and starts a timer. It then waits for a message containing the string "pong". If it receives such a message, it resets the timer and sets the socket back to blocking mode. If it does not receive a "pong" message within 10 seconds, it assumes that the client has disconnected and closes the socket.

If the watchdog program receives any other message, it ignores it and continues waiting for a "ping" message.

How to run

To build the executables, run "make all" in the command line while in the same directory as the Makefile.

This will build all three executables.

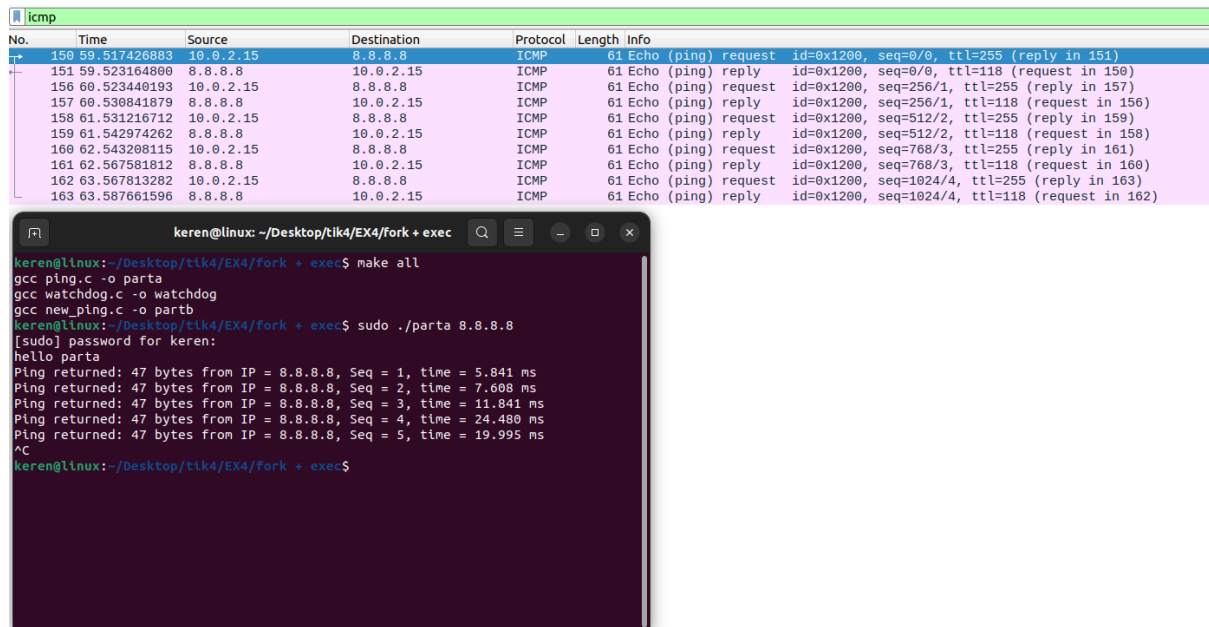
For part one of the assignment use "sudo ./parta <ip address>" command

For part two of the assignment use "sudo ./partb <ip address>" command

You can run "make clean" to remove the executables files.these

Part a:

part a runs in an infinite loop, we interrupted the code after 5 iteration and this is why wireshark cached 5 requests and 5 replies.




part b:

like the part a, runs in an infinite loop.

It communicates with the 'watchdog' in an infinite internal loop by sending and receiving requests, after 5 iterations we stopped receiving the response for 10 seconds, as you can see in the code, the screenshot* and the pcapng recording. then it stops the program and disconnects.

*We can see sending and receiving 5 requests and reply, and then there is no response, and then the connection is closed by the watchdog program



```
keren@linux: ~/Desktop/tik4/EX4/fork + exec
keren@linux:~/Desktop/tik4/EX4/fork + exec$ sudo ./partb 8.8.8.8
In child
hello partb
Ping returned: 35 bytes from IP = 8.8.8.8, Seq = 1, time = 17.945 ms
Ping returned: 35 bytes from IP = 8.8.8.8, Seq = 2, time = 7.913 ms
Ping returned: 35 bytes from IP = 8.8.8.8, Seq = 3, time = 5.798 ms
Ping returned: 35 bytes from IP = 8.8.8.8, Seq = 4, time = 30.731 ms
Ping returned: 35 bytes from IP = 8.8.8.8, Seq = 5, time = 6.363 ms
Ping returned: 35 bytes from IP = 8.8.8.8, Seq = 6, time = 10001.845 ms
Disconnected
keren@linux:~/Desktop/tik4/EX4/fork + exec$
```