Lab Assignment 1

Shashank Kashyap
17114070

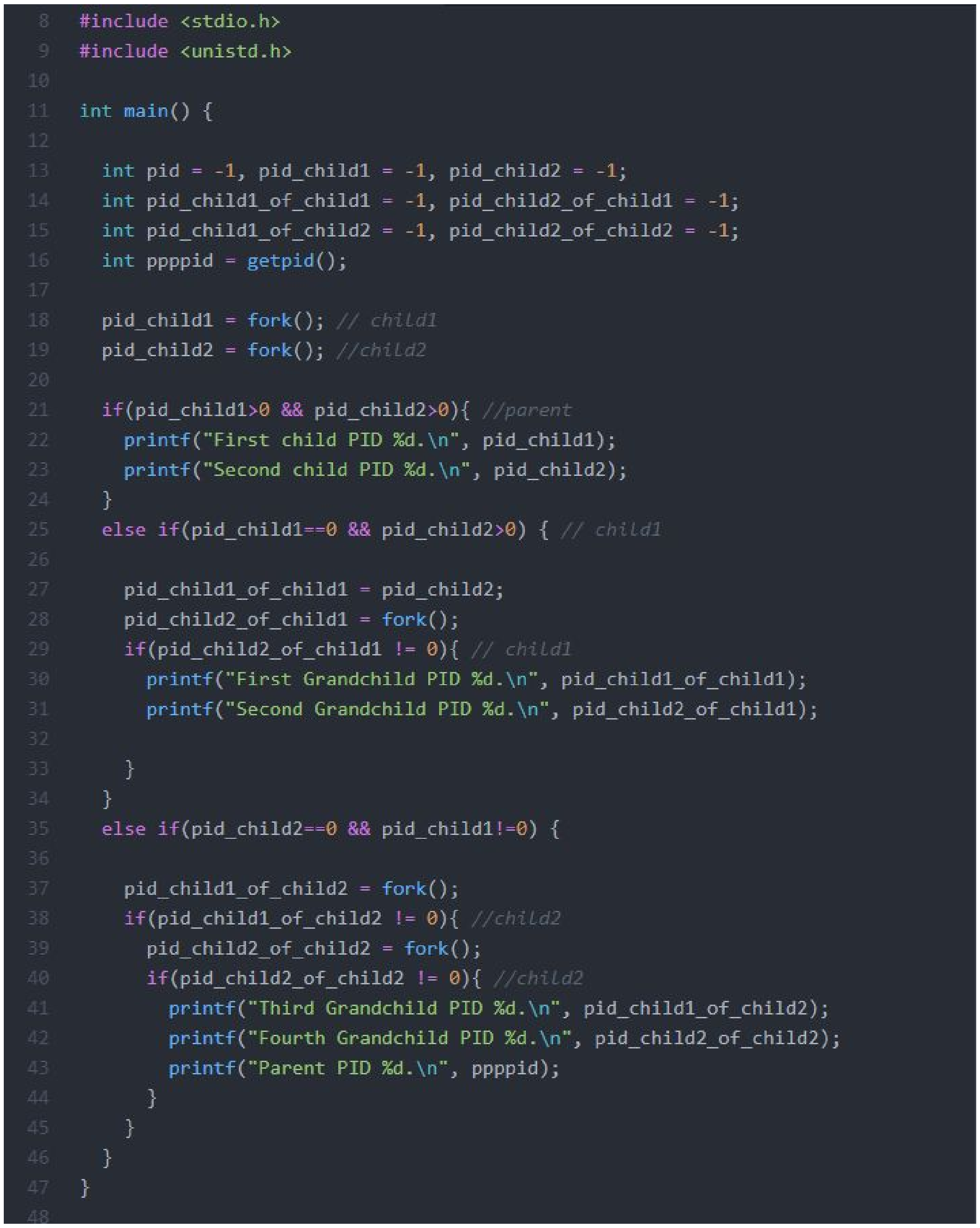
Question 1

<u>Problem Statement</u> - Write a C program in the UNIX system that creates two children and four grandchildren (two for each child). The program should then print the process-IDs of the two children, four grandchildren and the parent in this order.

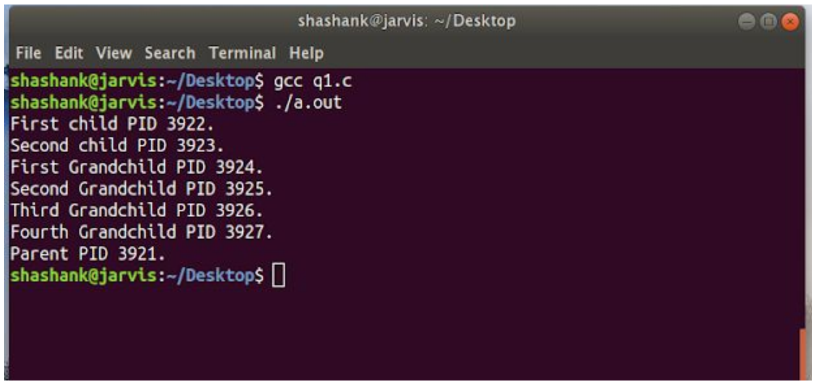
Solution -

- We initialize all PIDs by -1
- We find PIDs of child_1 and child_2 by creating child processes of parent process using **fork()**
- If both child processes were created successfully (PID > 0), we print those PIDs
- Now, if child_1 was created before (assigned lower PID) than child_2, then we create two child processes of child_1 and vice_versa.
- After creating the processes, we find the PIDs of these children processes

Code & Output -



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

int main() {

  int pid = -1, pid_child1 = -1, pid_child2 = -1;
  int pid_child1_of_child1 = -1, pid_child2_of_child1 = -1;
  int pid_child1_of_child2 = -1, pid_child2_of_child2 = -1;
  int ppppid = getpid();

  pid_child1 = fork(); // child1
  pid_child2 = fork(); //child2

  if(pid_child1>0 && pid_child2>0){ //parent
    printf("First child PID %d.\n", pid_child1);
    printf("Second child PID %d.\n", pid_child2);
  }
  else if(pid_child1==0 && pid_child2>0) { // child1

    pid_child1_of_child1 = pid_child2;
    pid_child2_of_child1 = fork();
    if(pid_child2_of_child1 != 0){ // child1
      printf("First Grandchild PID %d.\n", pid_child1_of_child1);
      printf("Second Grandchild PID %d.\n", pid_child2_of_child1);

    }
  }
  else if(pid_child2==0 && pid_child1!=0) {

    pid_child1_of_child2 = fork();
    if(pid_child1_of_child2 != 0){ //child2
      pid_child2_of_child2 = fork();
      if(pid_child2_of_child2 != 0){ //child2
        printf("Third Grandchild PID %d.\n", pid_child1_of_child2);
        printf("Fourth Grandchild PID %d.\n", pid_child2_of_child2);
        printf("Parent PID %d.\n", ppppid);
      }
    }
  }
}
```



shashank@jarvis: ~/Desktop
File Edit View Search Terminal Help

```
shashank@jarvis:~/Desktop$ gcc q1.c
shashank@jarvis:~/Desktop$ ./a.out
First child PID 3922.
Second child PID 3923.
First Grandchild PID 3924.
Second Grandchild PID 3925.
Third Grandchild PID 3926.
Fourth Grandchild PID 3927.
Parent PID 3921.
shashank@jarvis:~/Desktop$
```

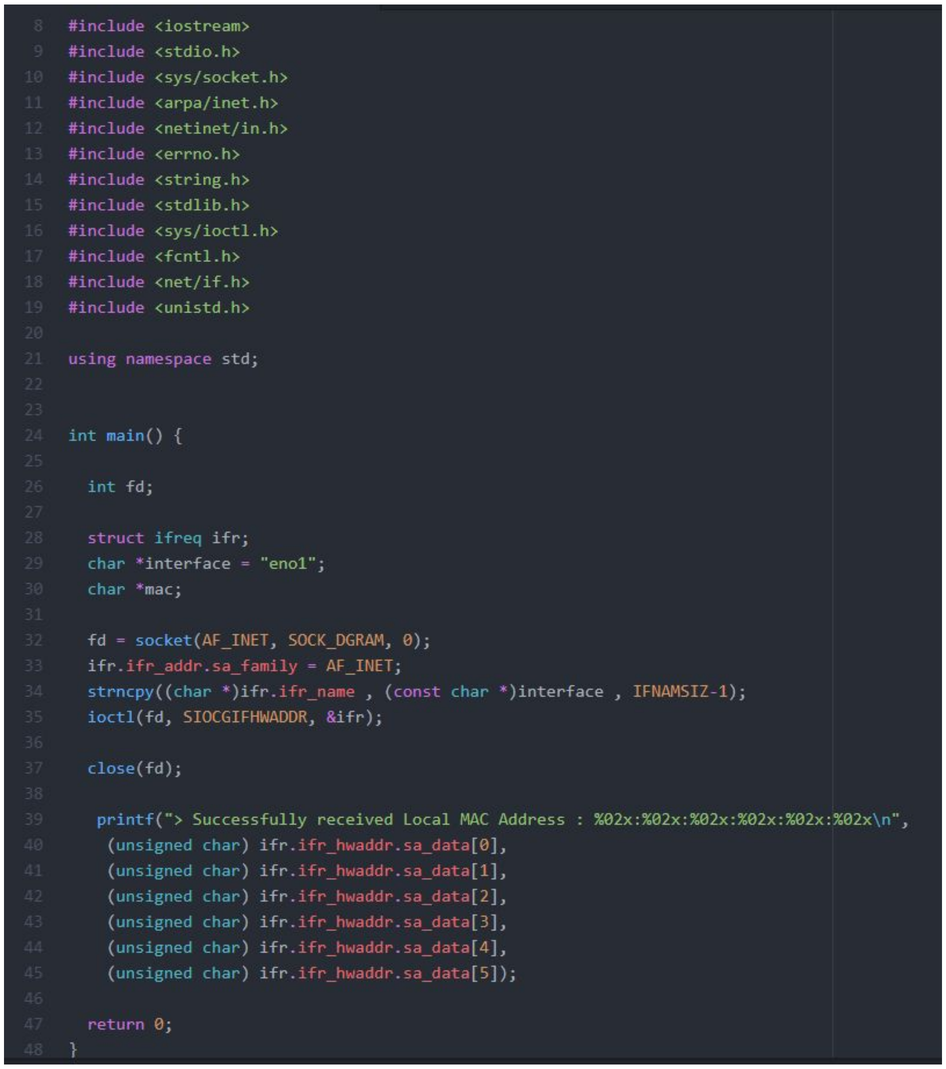
Question 2

<u>Problem Statement</u> - Write a C++ program to print the MAC address of your computer.

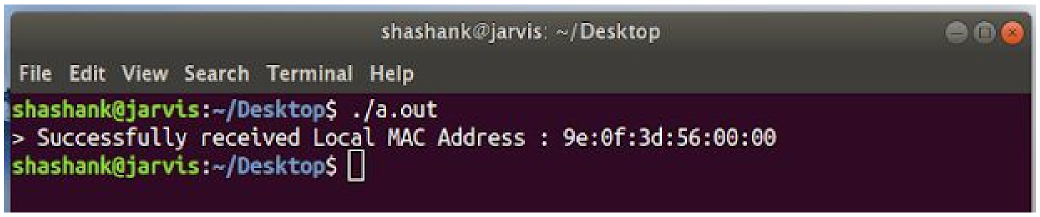
Solution -

- We construct a socket for communication
- Using the name of the interface, we establish the connection
- Use a system_call **ioctl** to query the MAC address of the device and this interface
- We receive the MAC address block by block rather than a single string and is kept in a struct.

Code & Output -



```cpp
#include <iostream>
#include <stdio.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <errno.h>
#include <string.h>
#include <stdlib.h>
#include <sys/ioctl.h>
#include <fcntl.h>
#include <net/if.h>
#include <unistd.h>

using namespace std;


int main() {

  int fd;

  struct ifreq ifr;
  char *interface = "eno1";
  char *mac;

  fd = socket(AF_INET, SOCK_DGRAM, 0);
  ifr.ifr_addr.sa_family = AF_INET;
  strncpy((char *)ifr.ifr_name , (const char *)interface , IFNAMSIZ-1);
  ioctl(fd, SIOCGIFHWADDR, &ifr);

  close(fd);

   printf("> Successfully received Local MAC Address : %02x:%02x:%02x:%02x:%02x:%02x\n",
    (unsigned char) ifr.ifr_hwaddr.sa_data[0],
    (unsigned char) ifr.ifr_hwaddr.sa_data[1],
    (unsigned char) ifr.ifr_hwaddr.sa_data[2],
    (unsigned char) ifr.ifr_hwaddr.sa_data[3],
    (unsigned char) ifr.ifr_hwaddr.sa_data[4],
    (unsigned char) ifr.ifr_hwaddr.sa_data[5]);

  return 0;
}
```


shashank@jarvis: ~/Desktop
File Edit View Search Terminal Help

```
shashank@jarvis:~/Desktop$ ./a.out
> Successfully received Local MAC Address : 9e:0f:3d:56:00:00
shashank@jarvis:~/Desktop$
```

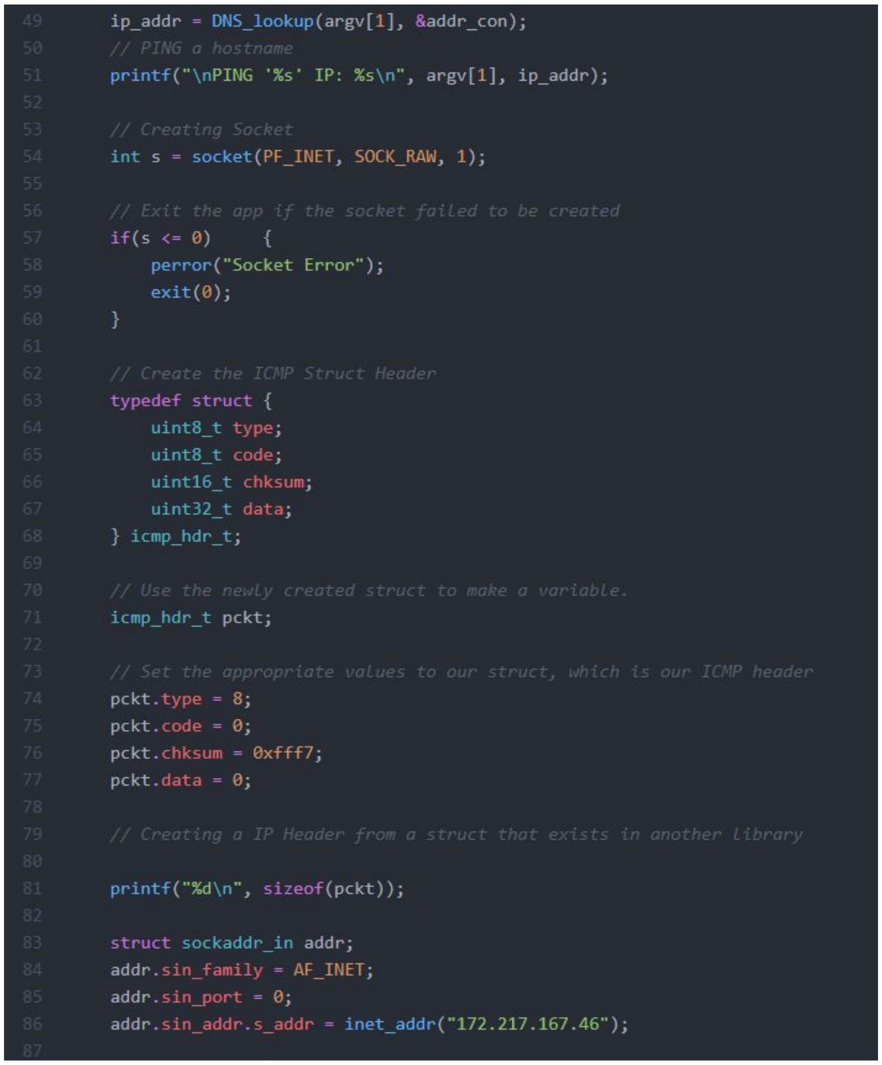
Question 3

<u>Problem Statement</u> - Write your own version of ping program in C language.

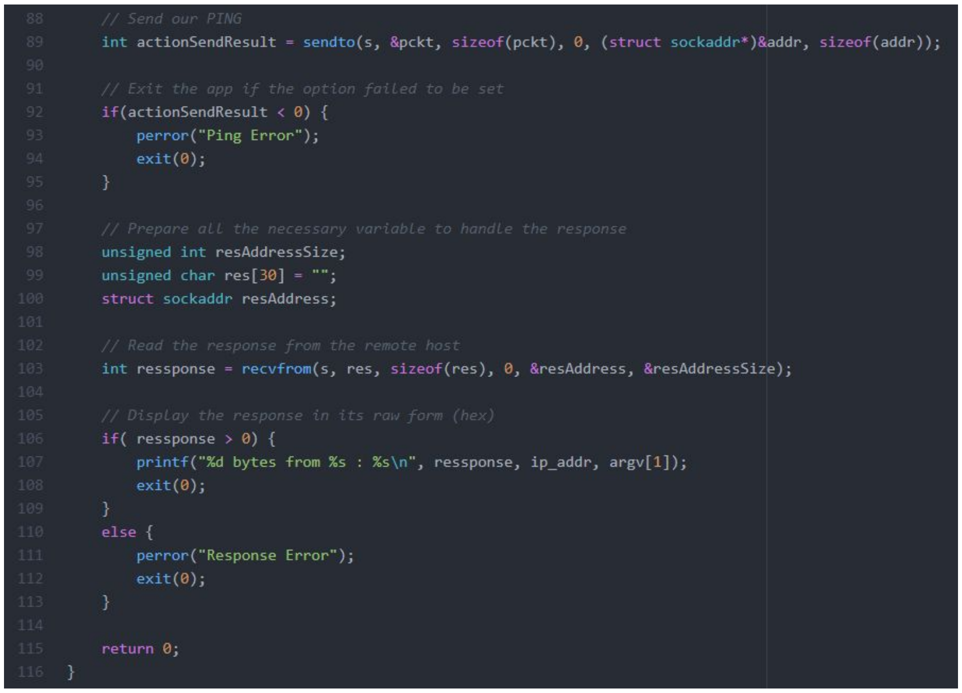
Solution -

- We first provide the hostname, to which we want to send the packet
- Then we use DNS lookup to obtain the IP address to which we want to connect.
- We then create a socket for communication
- Now we create ICMP packet as a struct and store the data(and checksum, etc) to be sent across the connection in that struct
- After this, we send the ping using **ICMP_ECHO**
- Create proper variables to handle the response and wait for it to be received
- Display the information we got in response.

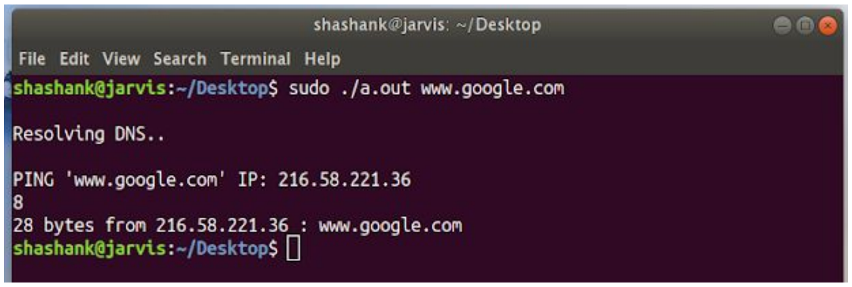
Code & Output -



```c
    ip_addr = DNS_lookup(argv[1], &addr_con);
    // PING a hostname
    printf("\nPING '%s' IP: %s\n", argv[1], ip_addr);

    // Creating Socket
    int s = socket(PF_INET, SOCK_RAW, 1);

    // Exit the app if the socket failed to be created
    if(s <= 0)     {
        perror("Socket Error");
        exit(0);
    }

    // Create the ICMP Struct Header
    typedef struct {
        uint8_t type;
        uint8_t code;
        uint16_t chksum;
        uint32_t data;
    } icmp_hdr_t;

    // Use the newly created struct to make a variable.
    icmp_hdr_t pckt;

    // Set the appropriate values to our struct, which is our ICMP header
    pckt.type = 8;
    pckt.code = 0;
    pckt.chksum = 0xfff7;
    pckt.data = 0;

    // Creating a IP Header from a struct that exists in another library

    printf("%d\n", sizeof(pckt));

    struct sockaddr_in addr;
    addr.sin_family = AF_INET;
    addr.sin_port = 0;
    addr.sin_addr.s_addr = inet_addr("172.217.167.46");

```



```c
    // Send our PING
    int actionSendResult = sendto(s, &pckt, sizeof(pckt), 0, (struct sockaddr*)&addr, sizeof(addr));

    // Exit the app if the option failed to be set
    if(actionSendResult < 0) {
        perror("Ping Error");
        exit(0);
    }

    // Prepare all the necessary variable to handle the response
    unsigned int resAddressSize;
    unsigned char res[30] = "";
    struct sockaddr resAddress;

    // Read the response from the remote host
    int ressponse = recvfrom(s, res, sizeof(res), 0, &resAddress, &resAddressSize);

    // Display the response in its raw form (hex)
    if( ressponse > 0) {
        printf("%d bytes from %s : %s\n", ressponse, ip_addr, argv[1]);
        exit(0);
    }
    else {
        perror("Response Error");
        exit(0);
    }

    return 0;
}
```



```
shashank@jarvis:~/Desktop$ sudo ./a.out www.google.com

Resolving DNS..

PING 'www.google.com' IP: 216.58.221.36
8
28 bytes from 216.58.221.36 : www.google.com
shashank@jarvis:~/Desktop$ 
```

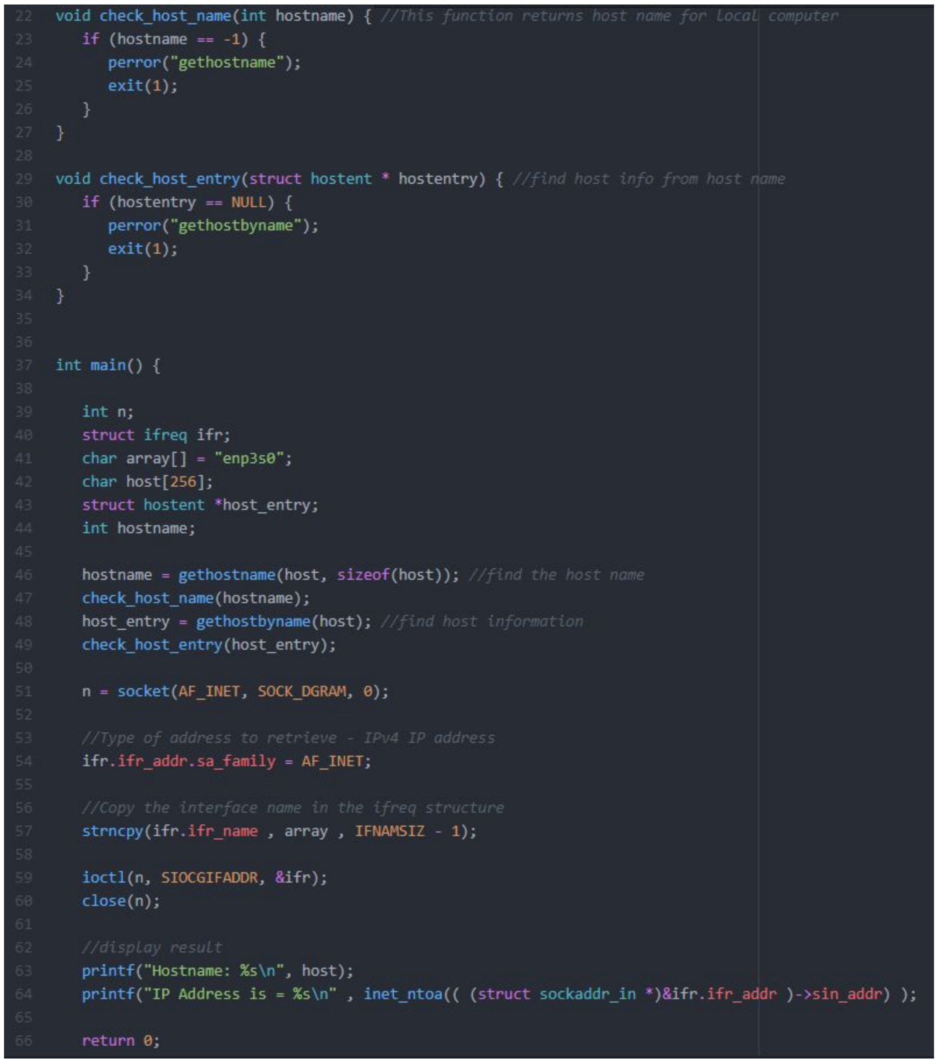
Question 4

<u>Problem Statement</u> - Write a C program to find the host name and the IP address of your computer.

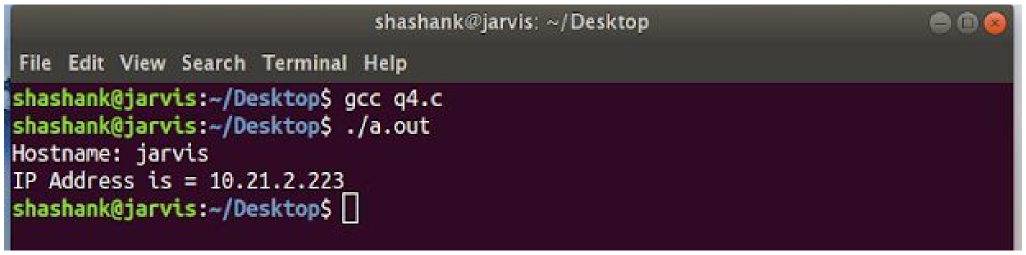
Solution -

- Find hostname of local computer using **gethostbyname**
- Find the name of the interface used by the computer.
- Create a socket for communication
- We do a system call **ioctl** to obtain details about the I/O devices.
- This give details about the IP address of the interface.

Code & Output -



```c
void check_host_name(int hostname) { //This function returns host name for local computer
   if (hostname == -1) {
      perror("gethostname");
      exit(1);
   }
}

void check_host_entry(struct hostent * hostentry) { //find host info from host name
   if (hostentry == NULL) {
      perror("gethostbyname");
      exit(1);
   }
}


int main() {

   int n;
   struct ifreq ifr;
   char array[] = "enp3s0";
   char host[256];
   struct hostent *host_entry;
   int hostname;

   hostname = gethostname(host, sizeof(host)); //find the host name
   check_host_name(hostname);
   host_entry = gethostbyname(host); //find host information
   check_host_entry(host_entry);

   n = socket(AF_INET, SOCK_DGRAM, 0);

   //Type of address to retrieve - IPv4 IP address
   ifr.ifr_addr.sa_family = AF_INET;

   //Copy the interface name in the ifreq structure
   strncpy(ifr.ifr_name , array , IFNAMSIZ - 1);

   ioctl(n, SIOCGIFADDR, &ifr);
   close(n);

   //display result
   printf("Hostname: %s\n", host);
   printf("IP Address is = %s\n" , inet_ntoa(( (struct sockaddr_in *)&ifr.ifr_addr )->sin_addr) );

   return 0;
```


shashank@jarvis: ~/Desktop
File Edit View Search Terminal Help

```
shashank@jarvis:~/Desktop$ gcc q4.c
shashank@jarvis:~/Desktop$ ./a.out
Hostname: jarvis
IP Address is = 10.21.2.223
shashank@jarvis:~/Desktop$ 
```