LAB ASSIGNMENT-1

CSN-361
Computer Networks Laboratory

Submitted by - Prateek Mali
Enrollment no. - 17114059

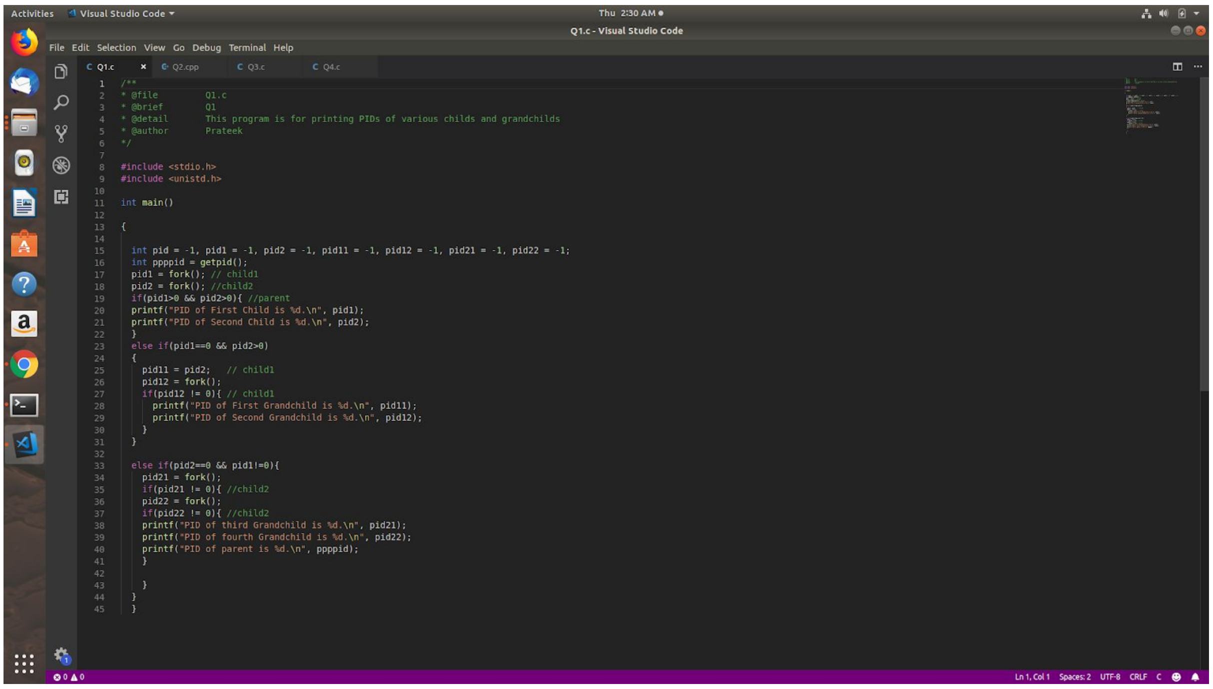
Problem Statements

1. 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.
2. Write a C++ program to print the MAC address of your computer.
3. Write your own version of ping program in C language.
4. Write a C program to find the host name from IP address.

Implementations details (Data Structures and Algorithms)

Q1

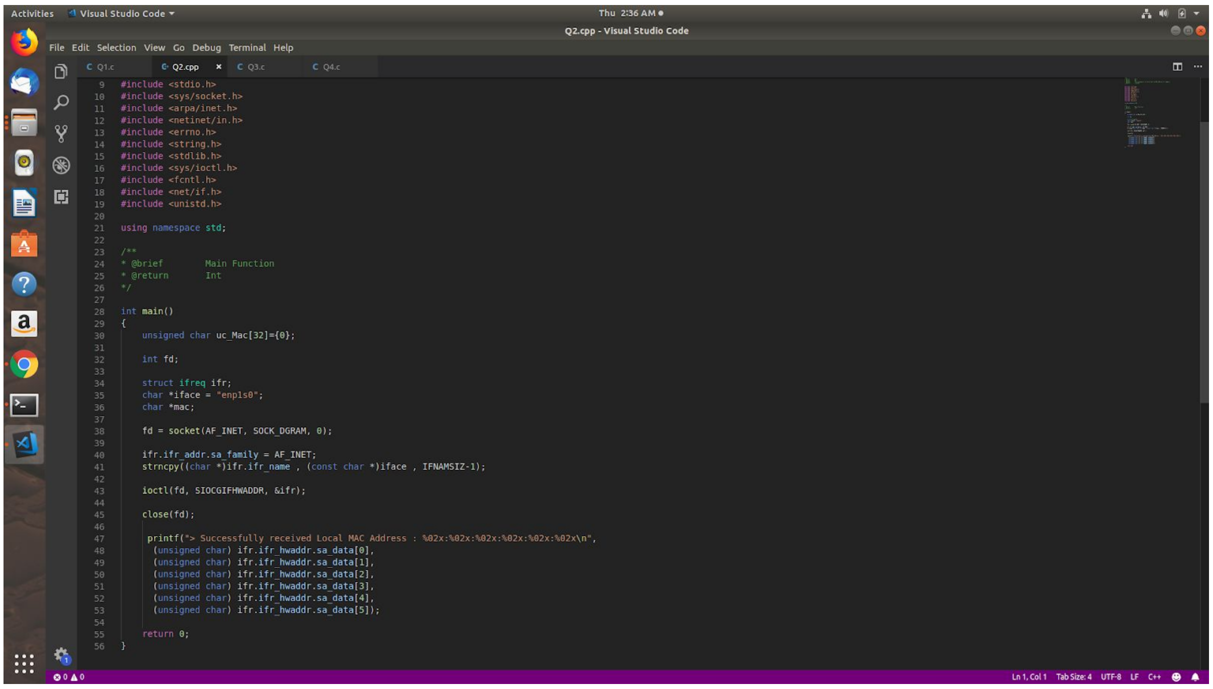
- Using fork we have created the childs and granchilds of the main process and have used the function getpid() to get the pid of parent.
- Libraries are used and no noticable data structure is used.



```c
/**
* @file        Q1.c
* @brief       Q1
* @detail      This program is for printing PIDs of various childs and grandchilds
* @author      Prateek
*/

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

int main()

{

  int pid = -1, pid1 = -1, pid2 = -1, pid11 = -1, pid12 = -1, pid21 = -1, pid22 = -1;
  int ppppid = getpid();
  pid1 = fork(); // child1
  pid2 = fork(); //child2
  if(pid1>0 && pid2>0){ //parent
  printf("PID of First Child is %d.\n", pid1);
  printf("PID of Second Child is %d.\n", pid2);
  }
  else if(pid1==0 && pid2>0)
  {
    pid11 = pid2;   // child1
    pid12 = fork();
    if(pid12 != 0){ // child1
      printf("PID of First Grandchild is %d.\n", pid11);
      printf("PID of Second Grandchild is %d.\n", pid12);
    }
  }

  else if(pid2==0 && pid1!=0){
    pid21 = fork();
    if(pid21 != 0){ //child2
    pid22 = fork();
    if(pid22 != 0){ //child2
    printf("PID of third Grandchild is %d.\n", pid21);
    printf("PID of fourth Grandchild is %d.\n", pid22);
    printf("PID of parent is %d.\n", ppppid);
    }

    }
  }
  }
```

Q2

- Struct is the data structure which is used here.
- Interface id is defined.
- Socket is created.
- Mac Address is fetched in blocks but represented as a whole in the end.



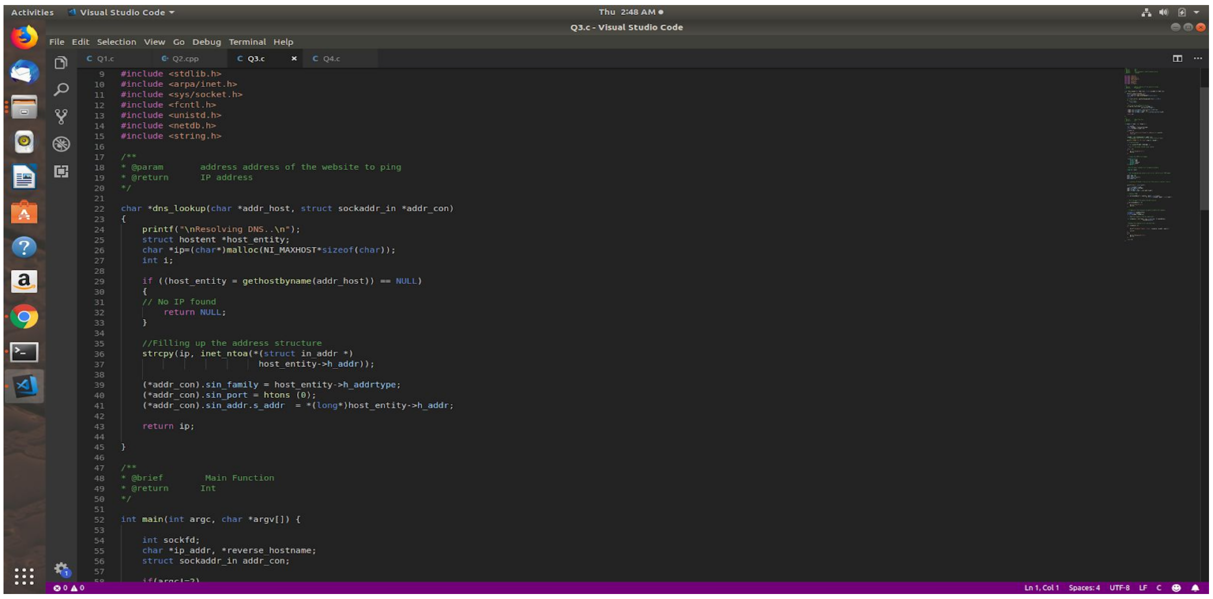
```cpp
#include <stdio.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <errno.h>
#include <string.h>
#include <stdlib.h>
#include <sys/ioctl.h>
#include <fcntl.h>
#include <net/if.h>
#include <unistd.h>

using namespace std;

/**
* @brief        Main Function
* @return       Int
*/

int main()
{
    unsigned char uc_Mac[32]={0};

    int fd;

    struct ifreq ifr;
    char *iface = "enp1s0";
    char *mac;

    fd = socket(AF_INET, SOCK_DGRAM, 0);

    ifr.ifr_addr.sa_family = AF_INET;
    strncpy((char *)ifr.ifr_name , (const char *)iface , 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;
}
```

<u>Q3</u>

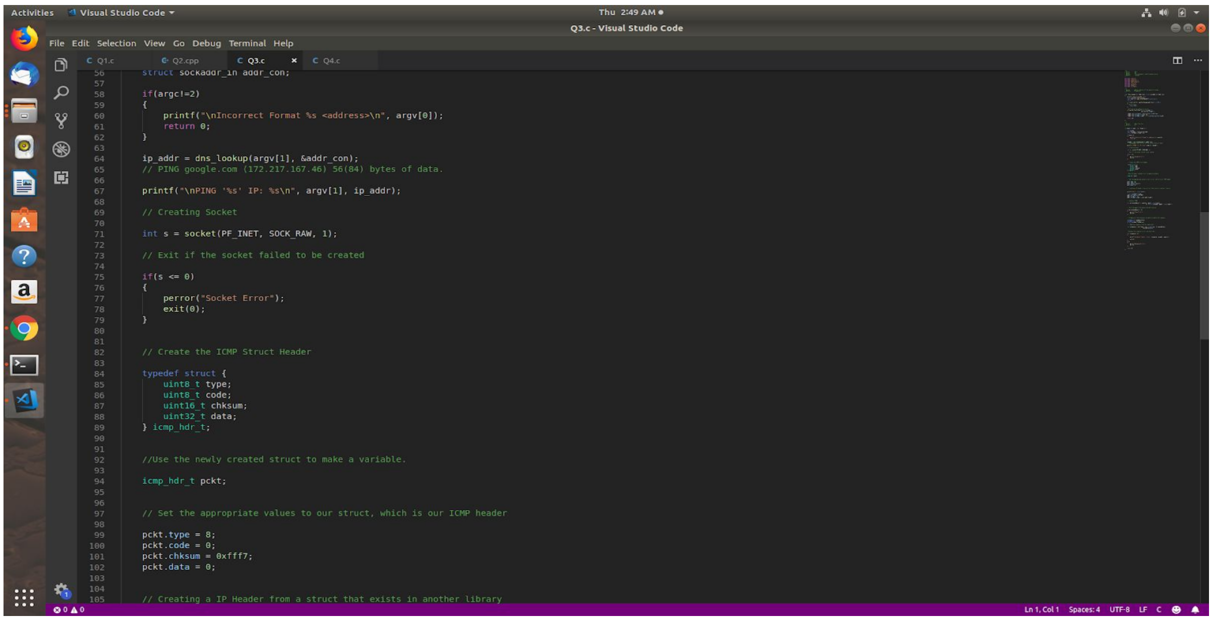
- Run dsn_lookup() by providing it the host address and retrieve the IP.
- Create socket.
- Exit if it fails to create socket.
- Create the ICMP struct header.
- Send Ping using sendto function.
- Reading and displaying of the response.

- **Struct is used as the main data structure here.**

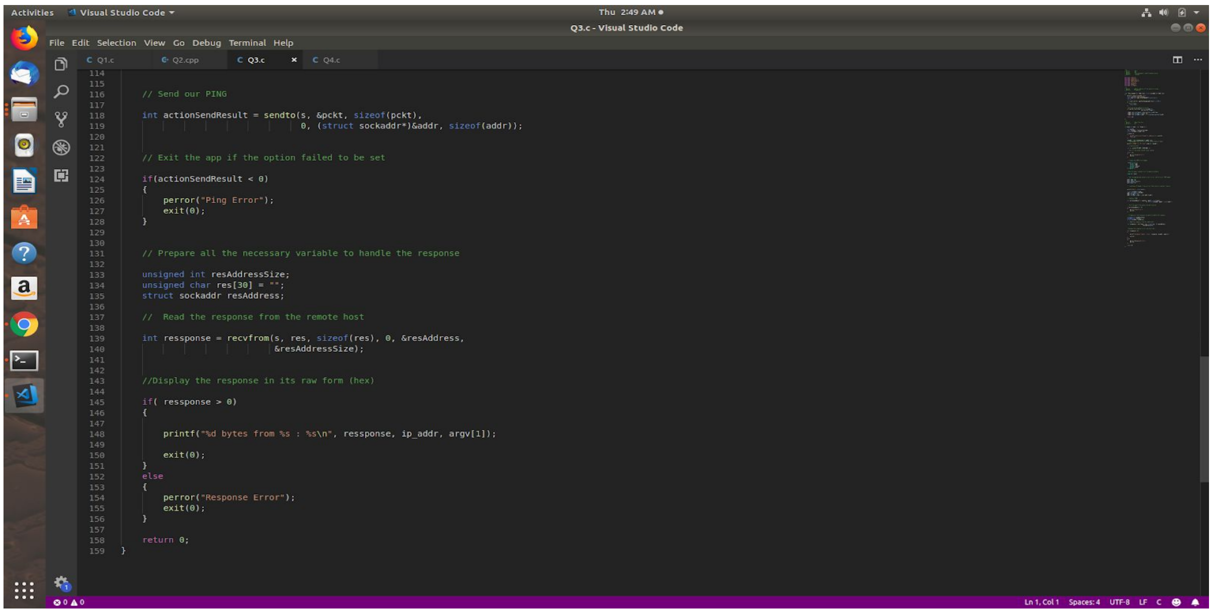
<u>Code Snippets</u>



```c
#include <stdlib.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <fcntl.h>
#include <unistd.h>
#include <netdb.h>
#include <string.h>

/**
* @param       address address of the website to ping
* @return      IP address
*/

char *dns_lookup(char *addr_host, struct sockaddr_in *addr_con)
{
    printf("\nResolving DNS..\n");
    struct hostent *host_entity;
    char *ip=(char*)malloc(NI_MAXHOST*sizeof(char));
    int i;

    if ((host_entity = gethostbyname(addr_host)) == NULL)
    {
    // No IP found
        return NULL;
    }

    //Filling up the address structure
    strcpy(ip, inet_ntoa(*(struct in_addr *)
                        host_entity->h_addr));

    (*addr_con).sin_family = host_entity->h_addrtype;
    (*addr_con).sin_port = htons (0);
    (*addr_con).sin_addr.s_addr  = *(long*)host_entity->h_addr;

    return ip;

}

/**
* @brief        Main Function
* @return      Int
*/

int main(int argc, char *argv[]) {

    int sockfd;
    char *ip_addr, *reverse_hostname;
    struct sockaddr_in addr_con;
```



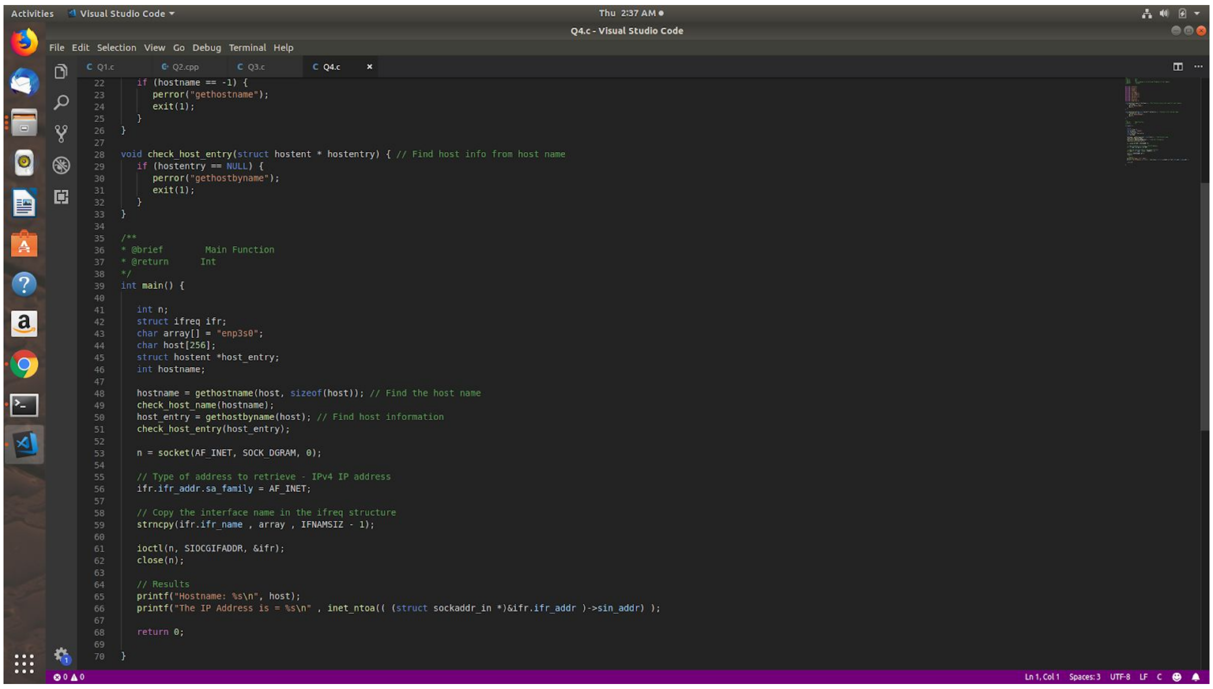
```c
    if(argc!=2)
    {
        printf("\nIncorrect Format %s <address>\n", argv[0]);
        return 0;
    }

    ip_addr = dns_lookup(argv[1], &addr_con);
    // PING google.com (172.217.167.46) 56(84) bytes of data.

    printf("\nPING '%s' IP: %s\n", argv[1], ip_addr);

    // Creating Socket

    int s = socket(PF_INET, SOCK_RAW, 1);

    // Exit 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
```



```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;
}
```

Q4

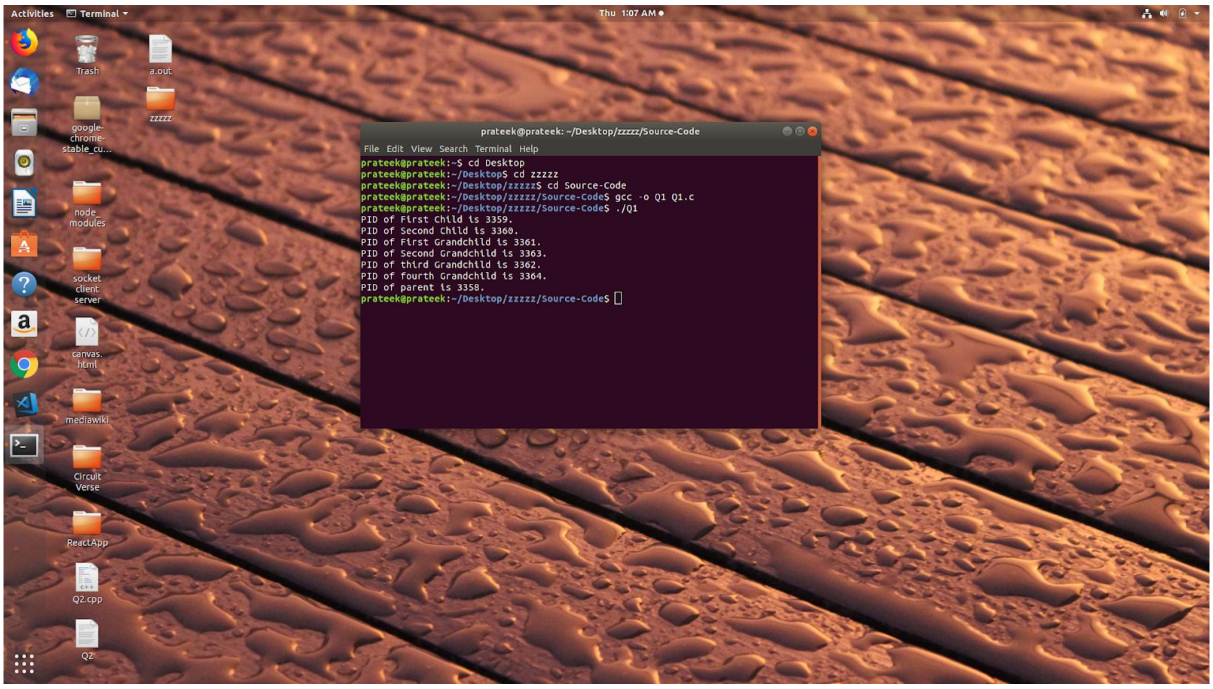
- Struct is the data structure which is used here.
- Interface id is defined.
- Socket is created for the connection.
- One struct is used for storing the host information and the other one is used for storing the types of addresses to retrieve.


Activities
Visual Studio Code
Thu 2:37 AM
Q4.c - Visual Studio Code
File Edit Selection View Go Debug Terminal Help
Q1.c
Q2.cpp
Q3.c
Q4.c
Ln 1, Col 1 Spaces: 3 UTF-8 LF C

```c
    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);
    }
}

/**
* @brief        Main Function
* @return      Int
*/
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);

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

    return 0;

}
```

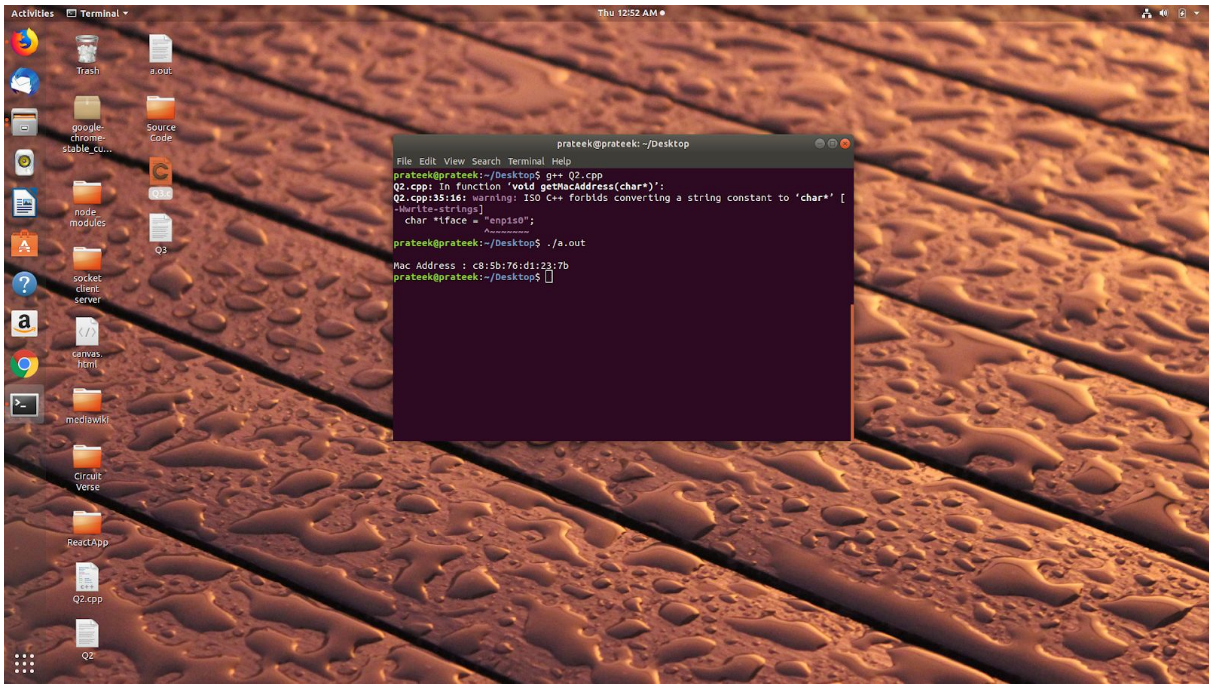
Snapshots of the running codes

Q1



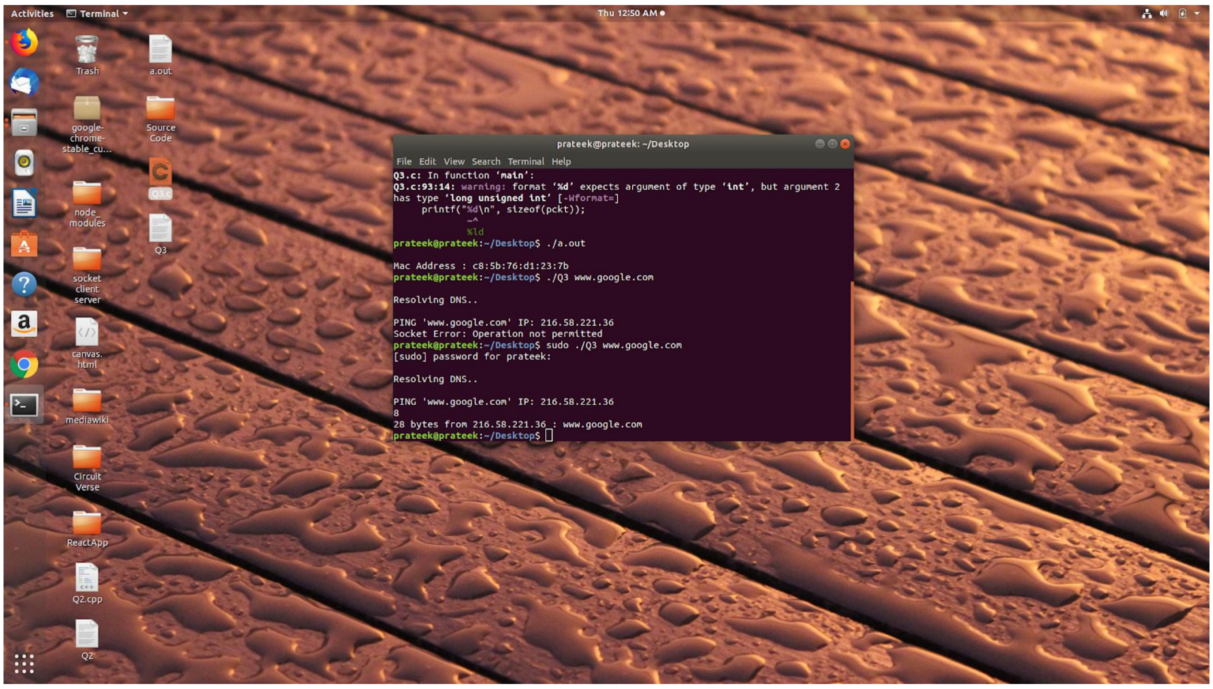
```
prateek@prateek:~$ cd Desktop
prateek@prateek:~/Desktop$ cd zzzzz
prateek@prateek:~/Desktop/zzzzz$ cd Source-Code
prateek@prateek:~/Desktop/zzzzz/Source-Code$ gcc -o Q1 Q1.c
prateek@prateek:~/Desktop/zzzzz/Source-Code$ ./Q1
PID of First Child is 3359.
PID of Second Child is 3360.
PID of First Grandchild is 3361.
PID of Second Grandchild is 3363.
PID of third Grandchild is 3362.
PID of fourth Grandchild is 3364.
PID of parent is 3358.
prateek@prateek:~/Desktop/zzzzz/Source-Code$
```

Q2



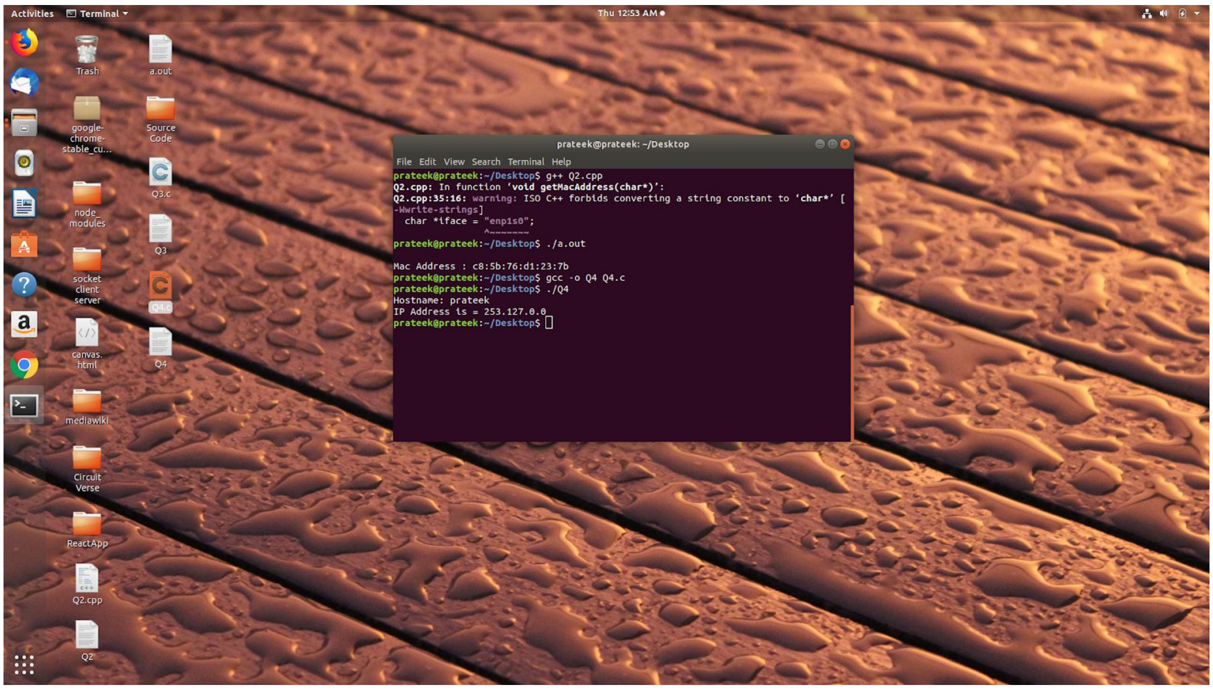
```
prateek@prateek:~/Desktop$ g++ Q2.cpp
Q2.cpp: In function ‘void getMacAddress(char*)’:
Q2.cpp:35:16: warning: ISO C++ forbids converting a string constant to ‘char*’ [
-Wwrite-strings]
  char *iface = "enp1s0";
                ^~~~~~~~
prateek@prateek:~/Desktop$ ./a.out

Mac Address : c8:5b:76:d1:23:7b
prateek@prateek:~/Desktop$
```

Q3



```
Q3.c: In function 'main':
Q3.c:93:14: warning: format '%d' expects argument of type 'int', but argument 2
has type 'long unsigned int' [-Wformat=]
     printf("%d\n", sizeof(pckt));
             ~^
             %ld
prateek@prateek:~/Desktop$ ./a.out

Mac Address : c8:5b:76:d1:23:7b
prateek@prateek:~/Desktop$ ./Q3 www.google.com

Resolving DNS..

PING 'www.google.com' IP: 216.58.221.36
Socket Error: Operation not permitted
prateek@prateek:~/Desktop$ sudo ./Q3 www.google.com
[sudo] password for prateek:

Resolving DNS..

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

Q4



```
prateek@prateek:~/Desktop$ g++ Q2.cpp
Q2.cpp: In function 'void getMacAddress(char*)':
Q2.cpp:35:16: warning: ISO C++ forbids converting a string constant to 'char*' [
-Wwrite-strings]
  char *iface = "enp1s0";
                ^~~~~~~~
prateek@prateek:~/Desktop$ ./a.out

Mac Address : c8:5b:76:d1:23:7b
prateek@prateek:~/Desktop$ gcc -o Q4 Q4.c
prateek@prateek:~/Desktop$ ./Q4
Hostname: prateek
IP Address is = 253.127.0.0
prateek@prateek:~/Desktop$
```

-