**MINOR ASSIGNMENT-03**

**UNIX Network Programming (CSE 4042)**

**1. Write a program to create an IPV4 socket address structure and pack the structure with family**

**AF INET, port=34567 and IP address 127.0.0.0.1 respectively from the command line. Also, dis-**

**play the port and IP address**

**Code-**

int main()

{

char ip[16];

in\_port\_t port;

struct sockaddr\_in sa;

bzero(&sa,sizeof(sa));

printf("Enter port:");

scanf("%hu",&port);

printf("Enter IP in dotted decimal:");

scanf("%s",ip);

sa.sin\_port=htons(port);

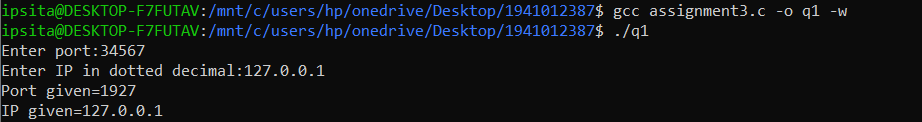
sa.sin\_addr.s\_addr=inet\_addr(ip);

printf("Port given=%hu\n",sa.sin\_port);

printf("IP given=%s\n",inet\_ntoa(sa.sin\_addr));

return 0; }

**Output snippet-**



**2. Write a program to declare two Internet socket address structure, namely servaddr and cliaddr respectively. Read the port and IP address for the structures you have declared from the keyboard and display the port and IP address.**

**Code-**

int main()

{

char ip\_serv[16];

in\_port\_t port\_serv;

printf("Enter port for server:");

scanf("%hu",&port\_serv);

printf("Enter IP in dotted decimal for server:");

scanf("%s",ip\_serv);

char ip\_cli[16];

in\_port\_t port\_cli;

printf("Enter port for client:");

scanf("%hu",&port\_cli);

printf("Enter IP in dotted decimal for client:");

scanf("%s",ip\_cli);

struct sockaddr\_in servaddr;

servaddr.sin\_port=htons(port\_serv);

servaddr.sin\_addr.s\_addr=inet\_addr(ip\_serv);

printf("Port given=%hu\n",ntohs(servaddr.sin\_port));

printf("IP given=%s\n",inet\_ntoa(servaddr.sin\_addr));

struct sockaddr\_in cliaddr;

cliaddr.sin\_port=htons(port\_cli);

cliaddr.sin\_addr.s\_addr=htonl(ip\_cli);

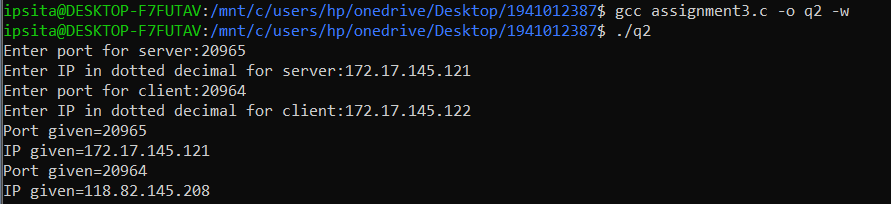
printf("Port given=%hu\n",ntohs(cliaddr.sin\_port));

printf("IP given=%s\n",inet\_ntoa(cliaddr.sin\_addr));

return 0;

}

**Output snippet-**



**3. Create a structure variable of the structure type struct sockaddr\_in defined in the header**

**<netinet/in.h>. Write a program to store the port and IP address in host byte order to net-**

**work byte order from the command-line argument. Display the values of the structure variable, port and IP address, from network byte order to host byte order onto the monitor**

**Code-**

int main()

{

in\_addr\_t ip;

in\_port\_t port;

struct sockaddr\_in sa;

printf("Enter port:");

scanf("%hu",&port);

printf("Enter IP address");

scanf("%u",&ip);

sa.sin\_port=htons(port);

sa.sin\_addr.s\_addr=htonl(ip);

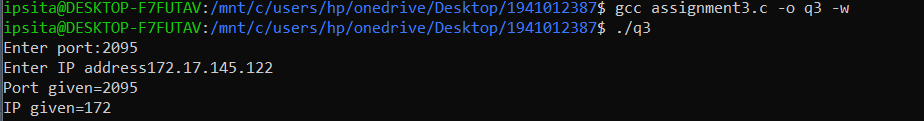
printf("Port given=%hu\n",ntohs(sa.sin\_port));

printf("IP given=%u\n",ntohl(sa.sin\_addr.s\_addr));

return 0;

}

**Output snippet-**

****

**4. Assume that a line of code in your program is to read the IPV4 address in dotted decimal number and is stored in the network byte order as sa.sin\_addr.s\_addr=inet addr(‘‘10.3.4.5’’),where sa is the structure variable of the type struct sockaddr\_in. You are required to rewrite the program to replace inet addr(...) with inet aton(...) to get the IP address and display that IP address using inet ntoa(..).**

**Code-**

int main()

{

char \*ip;

struct sockaddr\_in sa;

inet\_aton("10.3.4.5", &sa.sin\_addr);

ip = inet\_ntoa(sa.sin\_addr);

printf("IP Address=%s\n",ip);

return 0;

}

**Output snippet-**

****

**5. Assume that a line of code in your program is to read the IPV4 address in dotted decimal number and is stored in the network byte order as ca.sin\_addr.s\_addr=inet addr(‘‘10.3.4.5’’), where ca is the structure variable of the type struct sockaddr\_in. You are required to rewrite the program to replace inet addr(...) with inet pton(...) to get the IP address and display that IP address using inet ntop(..).**

**Code-**

int main()

{char ip[16];

struct sockaddr\_in ca;

inet\_pton(AF\_INET, "10.3.4.5", &ca.sin\_addr);

inet\_ntop(AF\_INET, &ca.sin\_addr, ip ,16);

printf("IP Address=%s\n",ip);

return 0;

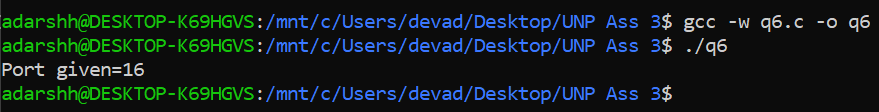
}

**Output snippet-**

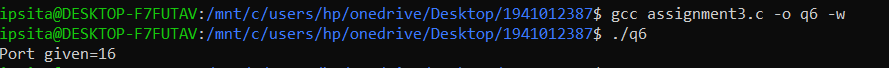


Text, letter

Description automatically generated

**Output snippet-**

Here, simply the servaddr.sin\_port was assigned the value 16, hence the output displays 16 as port given.



**7. Find out the output of the given code snippet and justify the reason of getting such output (Hint: look into Host byte order and Network byte order).**

Graphical user interface, text, application

Description automatically generated

**Output snippet-**

****

**Explanation-**

Here, servaddr.sin\_port was assigned the value 16 over htons function which swaps its byte ordering and converts it into Big-endian notation (i.e., converts it to Network byte ordering from Host byte ordering).

16=00000000 00010000

After Swap=00010000 00000000=4096

Graphical user interface, text

Description automatically generated

**Filled Code-**

int main()

{

**in\_port\_t port; // or unsigned short int port;**

printf("Enter a port address:");

scanf**("%hu"**,&port);

struct sockaddr\_in servaddr;

servaddr.sin\_family=AF\_INET;

servaddr.sin\_port=port;

printf("Port given=%d\n",htons(servaddr.sin\_port));

return 0;

}

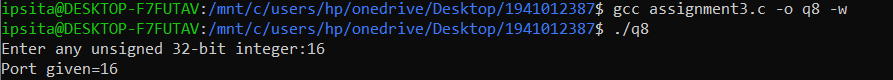
**Output Snippet-**



Graphical user interface, text, application, email

Description automatically generated

**Output snippet-**



**Explanation-**

Here, a 32-bit long integer is taken as input from command line and is passed over a htonl(host to network) function which converts it into network byte order i.e. Big endian. Further it is passed on with a ntohl(network to host) function which again does byte swapping and converts it back to host byte order i.e. Little endian.

16=00000000 00010000

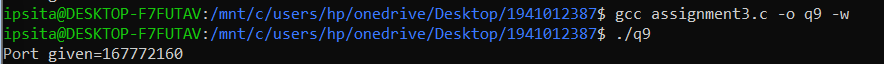
After 1st Swap=00010000 00000000=4096 (htonl)

After 2nd Swap=00000000 00010000=16 (ntohl)

Text, letter

Description automatically generated

**Output snippet-**



**Explanation-**

Here, ip is assigned a value htonl(10) which converts 10 to Big endian notation and displays it.

10=00000000 00000000 00000000 00001010

After Swap=00001010 00000000 00000000 00000000=167772160

10. Find out the output of the code snippet and also trace the reason of getting such output.

Text

Description automatically generated

**Output snippet-**

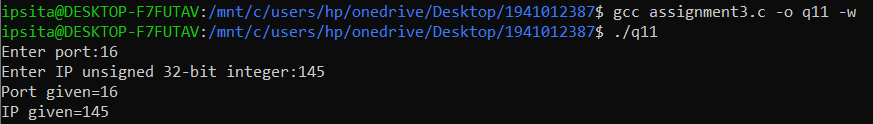


Here, “ip” which is declared as a unsigned 32 bit variable was assigned the value 10, hence the output displays 10 as port given.

Text

Description automatically generated

**Output snippet-**

****

**Explanation-**Here, one 32-bit long integer and a 16 bit unsigned short integer are taken as input from command line and are passed over a htonl(host to network long) and htons(host to network short) function respectively which converts it into network byte order i.e. Big endian. Further they are passed on with a ntohl(network to host long) and ntohs(network to host short) function respectively which again does byte swapping and converts it back to the host byteorder i.e. Little endian.

Id=145=00000000 00000000 00000000 10010001

After 1st Swap=10010001 00000000 00000000 00000000=2432696320 (htons)

After 2nd Swap=00000000 00000000 00000000 10010001=145 (ntohs)

Port=16=00000000 00010000

After 1st Swap=00010000 00000000=4096 (htons)

After 2nd Swap=00000000 00010000=16 (ntohs)

**11. Write a program to read an unsigned 32-bit integer x ( i.e datatype uint32 t). Assign x to y in network byte order. Display the value of y in network byte order as well as in host byte order.**

**Code-**

int main()

{

uint32\_t y;

unsigned int x;

printf("Enter any unsigned 32-bit integer:");

scanf("%u",&x);

y=htonl(x);

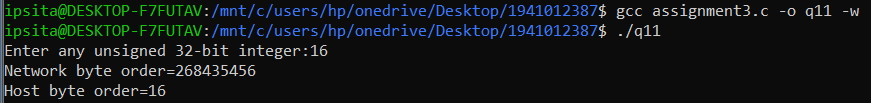
printf("Network byte order=%u\n",y);

printf("Host byte order=%u\n",ntohl(y));

return 0;

}

**Output snippet-**

****

12. Develop a program to determine whether your working machine is in little-endian or in big-endian

byte order.

**Code-**

#include <stdio.h>

int main()

{

int x = 1;

char \*cp = &x;

if(\*cp == 1)

printf("Little endian Machine\n");

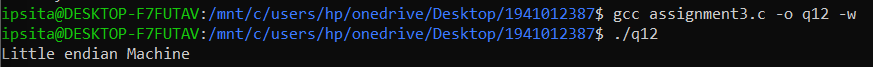
else

printf("Big endian Machine\n");

return 0;

}

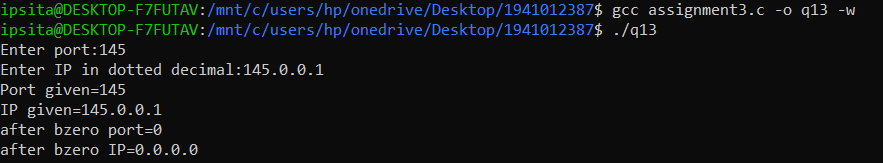
**Output snippet-**



Text

Description automatically generated**Since, I use a x86 intel processor my system is Little endian.**

**Output snippet-**



**Explanation-** Since, we know that bzero sets the specified number of bytes to 0 in the destination and here it is done for the size of the whole structure variable for which all the bytes are set to 0.

Text

Description automatically generatedRun the test cases and justify the program generated output

1. Enter the IP address: 12.13.14.15

(b) Enter the IP address: 12.13.14

(c) Enter the IP address: 12.13

(d) Enter the IP address: 12

(e) Enter the IP address: 255.255.255.255

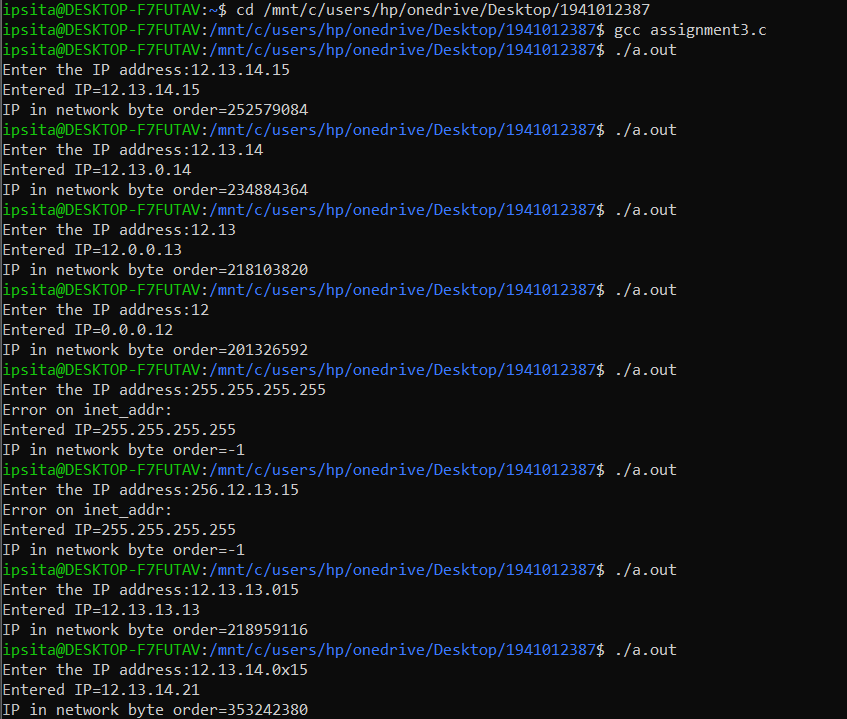
(f) Enter the IP address: 256.12.13.15

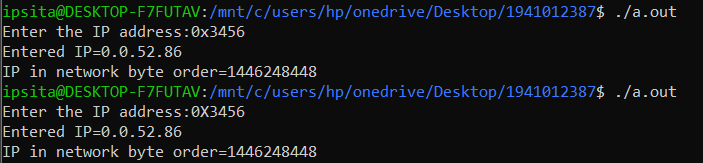
(g) Enter the IP address: 12.13.14.015

(h) Enter the IP address: 12.13.14.0x15

(i) Enter the IP address: 0x3456

(j) Enter the IP address: 0X3456





**Explanation-** Here, the input string is passed over inet\_addr to check if it’s a valid ip address, if not valid i.e., exceeds or equals 255.255.255.255 then it throws -1 else the dotted decimal notation.

That is why (e) and (f) throw an error. In (i) and (j) the inputs are given as hexadecimal values and output show their respective dotted decimal notation.

**15. Rewrite the question-14 code to replace inet addr() function with inet aton() and verify the above test cases in your modified program.**

**Code-**

int main(){

struct sockaddr\_in serv;

char ipaddr[16];

printf("Enter the IP address:");

scanf("%s",ipaddr);

inet\_aton(ipaddr,&serv.sin\_addr.s\_addr);

if(inet\_aton(ipaddr,&serv.sin\_addr.s\_addr)==0)

{

printf("Error on inet\_addr:\n");

}

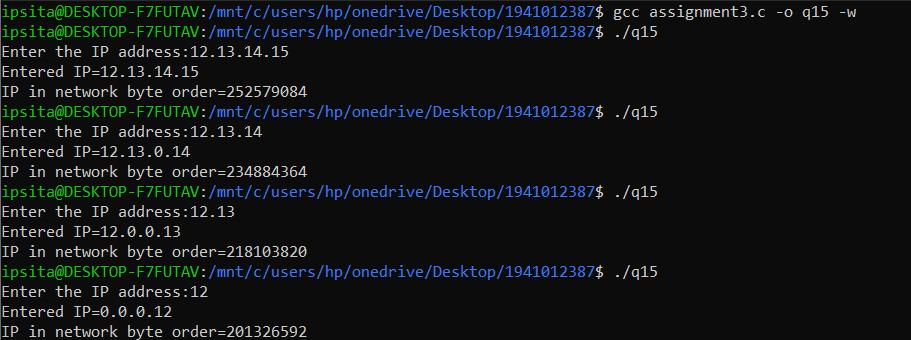
printf("Entered IP=%s\n",inet\_ntoa(serv.sin\_addr));

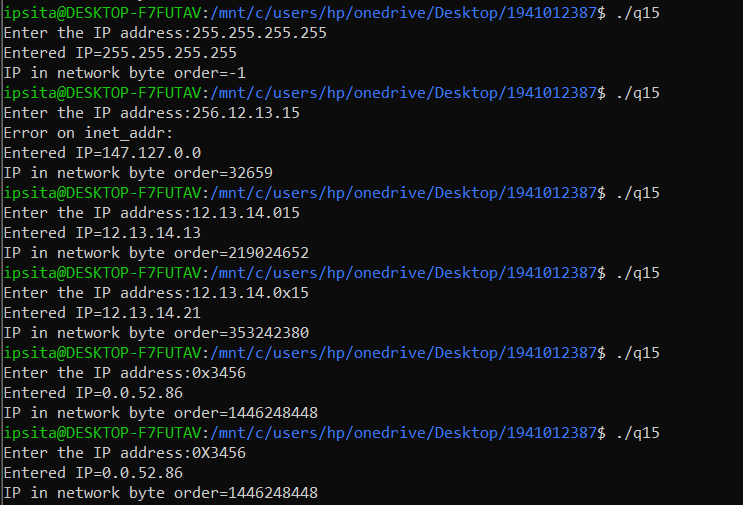
printf("IP in network byte order=%d\n",serv.sin\_addr.s\_addr);

return 0;

}

**Output snippet-**





**16. Rewrite the question-14 code to replace inet addr() function with inet pton() and verify the above test cases in your modified program.**

**Code-**

int main(){

struct sockaddr\_in serv;

char ipaddr[16];

printf("Enter the IP address:");

scanf("%s",ipaddr);

inet\_pton(AF\_INET,ipaddr,&serv.sin\_addr.s\_addr);

if(inet\_pton(AF\_INET,ipaddr,&serv.sin\_addr.s\_addr)==0){

printf("Error on inet\_addr:\n");

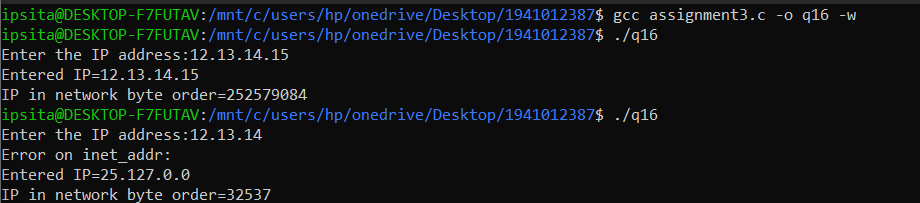
}

printf("Entered IP=%s\n",inet\_ntoa(serv.sin\_addr));

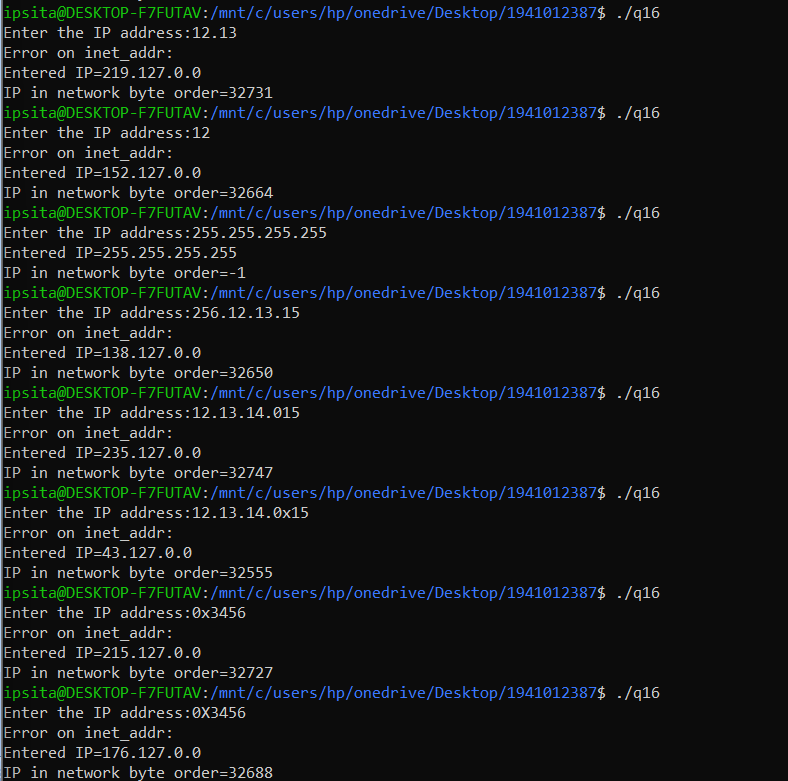
printf("IP in network byte order=%d\n",serv.sin\_addr.s\_addr);

return 0;

}



**Output snippet-**

****

**17. Rewrite the question-14 code to replace inet addr() function with inet pton() and verify the above test cases in your modified program.**

**--Same as 16---**

**18. Rewrite the question-14 code to replace inet addr() function with inet pton() and inet ntoa() to inet ntop(). Also, verify the above test cases in your modified program.**

**Code-**

int main(){

struct sockaddr\_in serv;

char ipaddr[16];

printf("Enter the IP address:");

scanf("%s",ipaddr);

inet\_pton(AF\_INET,ipaddr,&serv.sin\_addr.s\_addr);

if(inet\_pton(AF\_INET,ipaddr,&(serv.sin\_addr.s\_addr))==0){

printf("Error on inet\_addr:\n");

}

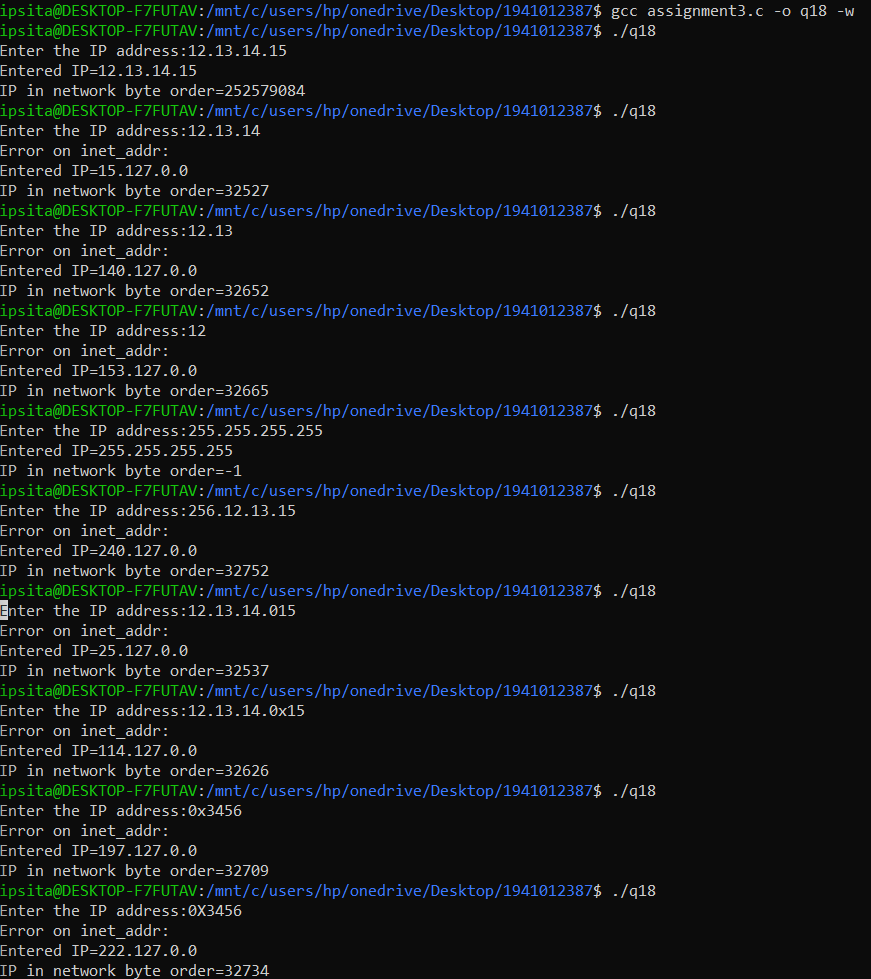
printf("Entered IP=%s\n",inet\_ntop(AF\_INET,&(serv.sin\_addr),ipaddr,16));

printf("IP in network byte order=%d\n",serv.sin\_addr.s\_addr);

return 0;

}

**Output snippet-**

****

**20. Write a program to create a socket (i.e. end-point of a connection ) and display the whether end-point is successfully created or not.**

**Code-**

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

{

int socket\_fd;

struct sockaddr\_in server;

//Socket (i.e. end-point of a connection )created successfully

socket\_fd = socket(AF\_INET , SOCK\_STREAM , IPPROTO\_TCP);//The protocol type mentioned supports the family

if (socket\_fd < 0)

{

printf("Could not create socket\n");

}

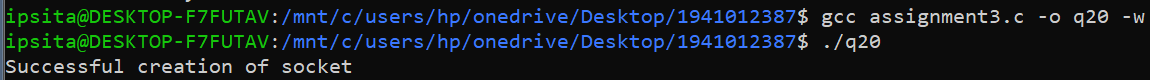
else {

printf("Successful creation of socket\n");

}

}

**Output snippet-**

****