INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

Computers Networks

"IP Calculator"

Abstract

The purpose of this report is not to only document and explain the code, if not trying to give a deep understanding of why the code work and to analyze the given results. Also, it'll guide you in the execution of the program, as well how to read the output and what it all means.

By:
Jose David Portilla Martinez

Professor:
MSc. NIDIA ASUNCIÓN CORTEZ DUARTE

September 24

To validate this report it is necessary to include the corresponding seal

Index

Content

Introduction:

This is an easy to follow application, what might be a little confusing or what you might ask is, what do you have to give a certain input and why does it give that output, well, if brief everything will be explained and also for those who have never run a program through a command console, there will be a little tutorial.

Literature Review:

IP Address Basics

For systems to locate each other in a distributed environment, nodes are given explicit addresses that uniquely identify a particular network the system is on and uniquely identify the system to that particular network. When these two identifiers are combined, the result is a globally-unique address.

This address, known as "IP address", as "IP number", or merely as "IP" is a code made up of numbers separated by three dots that identifies a computer on the Internet. These addresses are 32-bit binary numbers, consisting of the two sub-addresses (identifiers) mentioned above which, respectively, identify the network and the host to the network, with an imaginary boundary separating the two. An IP address is, as such, generally shown as 4 octets of numbers from 0-255 represented in decimal form instead of binary form. For example:

the address 168.212.226.204 represents the 32-bit binary number
10101000.11010100.11100010.11001100.

The binary number is important because that will determine which class of network the IP address belongs to.

There are 5 IP Classes:

- **Class A**: these addresses always have the first bit of their IP addresses set to "0". Since Class A networks have an 8-bit network mask, the use of a leading zero leaves only 7 bits for the network portion of the address, allowing for a maximum of 128 possible network numbers, ranging from 0.0.0.0 – 127.0.0.0. *Number 127.x.x.x is reserved for loopback, used for internal testing on the local machine.*
- **Class B**: these addresses always have the first bit set to "1" and their second bit set to "0". Since Class B addresses have a 16-bit network mask, the use of a leading "10" bit-pattern leaves 14 bits for the network portion of the address, allowing for a maximum of 16,384 networks, ranging from 128.0.0.0 – 191.255.0.0.
- **Class C**: these addresses have their first two bits set to "1" and their third bit set to "0". Since Class C addresses have a 24-bit network mask, this leaves 21 bits for the network portion of the address, allowing for a maximum of 2,097,152 network addresses, ranging from 192.0.0.0 – 223.255.255.0.

- **Class D**: these addresses are used for multicasting applications. Class D addresses have their first three bits set to "1" and their fourth bit set to "0". Class D addresses are 32-bit network addresses, meaning that all the values within the range of 224.0.0.0 – 239.255.255.255 are used to uniquely identify multicast groups. There are no host addresses within the Class D address space, since all the hosts within a group share the group's IP address for receiver purposes.
- **Class E**: these addresses are defined as experimental and are reserved for future testing purposes. *They have never been documented or utilized in a standard way.*

Network Part

This part specifies the unique number assigned to your network. It also identifies the class of network assigned. The network part takes two bytes of the IPv4 address.

Host Part

This is the part of the IPv4 address that you can assign to each host. It uniquely identifies this machine on your network. For all hosts on your network, the network part of the IP address will be the same and host part will be changing.

Procedure:

Step-by-Step Procedure

All following instructions will be displayed in the *figure 1.0*

- It will let you enter an IP address, it'll check if it's within the ranges of a valid IP.
- Behind the scenes, in the process or run of the program that isn't showing, first it will check and then show what class the given IP has, if it has an *A, B* or *C class,* it will set the netmask.
- If the IP class turned out to be D or E, it'll show you only that, nothing more.
- Otherwise, it'll show you the *netmask*, which was initialized when the class was determined, what *kind of IP is, that'd be Network, Broadcast or Host*, and, depending of the case:
  - If the kind of IP is either *Network or Broadcast:* it'll show only the kind and the other IP
  - Else, if it's a Host IP, it will give you the Network AND the Broadcast IP.
- Finally, it will also show you the range of the IP and then ask you if you want to repeat the program, if you want you just have to press [1] or [0] if you don't.

Code’s Flowchart

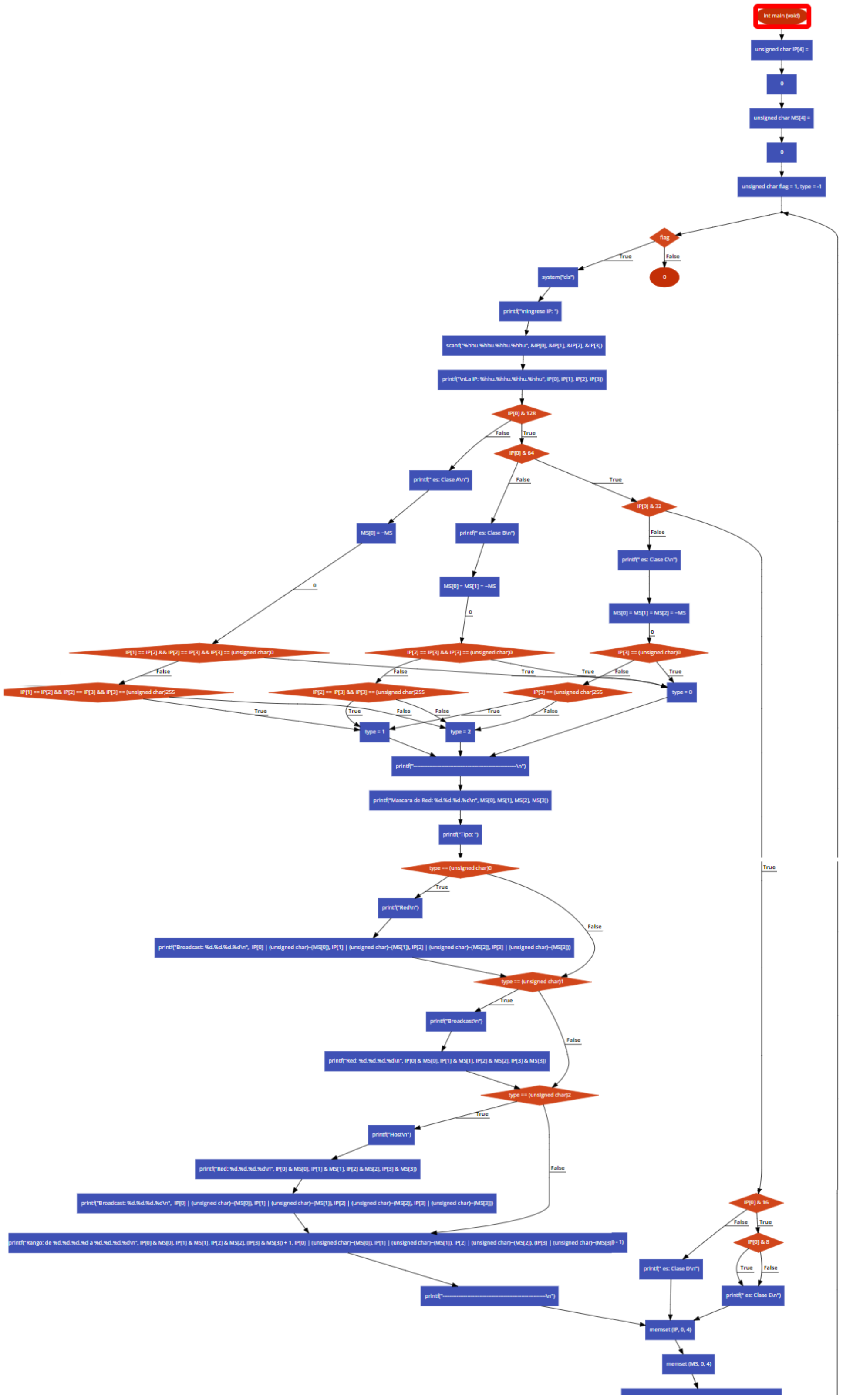
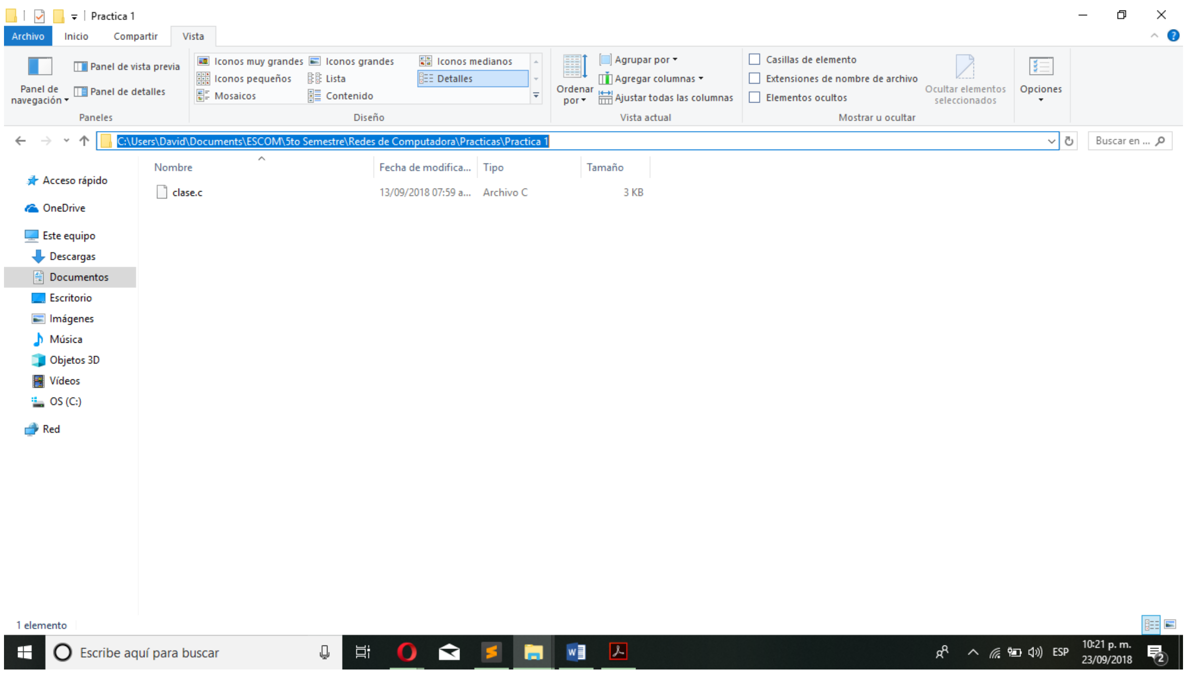

int main (void)
unsigned char IP[4] =
0
unsigned char MS[4] =
0
unsigned char flag = 1, type = -1
flag
True
False
0
system("cls")
printf("\nIngrese IP: ")
scanf("%hhu.%hhu.%hhu.%hhu", &IP[0], &IP[1], &IP[2], &IP[3])
printf("\nLa IP: %hhu.%hhu.%hhu.%hhu", IP[0], IP[1], IP[2], IP[3])
IP[0] & 128
False
True
IP[0] & 64
printf(" es: Clase A\n")
False
True
printf(" es: Clase B\n")
IP[0] & 32
False
printf(" es: Clase C\n")
MS[0] = ~MS
MS[0] = MS[1] = ~MS
MS[0] = MS[1] = MS[2] = ~MS
0
0
0
IP[1] == IP[2] && IP[2] == IP[3] && IP[3] == (unsigned char)0
IP[2] == IP[3] && IP[3] == (unsigned char)0
IP[3] == (unsigned char)0
False
True
False
True
False
True
IP[1] == IP[2] && IP[2] == IP[3] && IP[3] == (unsigned char)255
IP[2] == IP[3] && IP[3] == (unsigned char)255
IP[3] == (unsigned char)255
type = 0
True
True
False
False
True
False
type = 1
type = 2
printf("--\n")
printf("Mascara de Red: %d.%d.%d.%d\n", MS[0], MS[1], MS[2], MS[3])
printf("Tipo: ")
type == (unsigned char)0
True
printf("Red\n")
False
printf("Broadcast: %d.%d.%d.%d\n", IP[0] | (unsigned char)~(MS[0]), IP[1] | (unsigned char)~(MS[1]), IP[2] | (unsigned char)~(MS[2]), IP[3] | (unsigned char)~(MS[3]))
type == (unsigned char)1
True
printf("Broadcast\n")
False
printf("Red: %d.%d.%d.%d\n", IP[0] & MS[0], IP[1] & MS[1], IP[2] & MS[2], IP[3] & MS[3])
type == (unsigned char)2
True
printf("Host\n")
False
printf("Red: %d.%d.%d.%d\n", IP[0] & MS[0], IP[1] & MS[1], IP[2] & MS[2], IP[3] & MS[3])
printf("Broadcast: %d.%d.%d.%d\n", IP[0] | (unsigned char)~(MS[0]), IP[1] | (unsigned char)~(MS[1]), IP[2] | (unsigned char)~(MS[2]), IP[3] | (unsigned char)~(MS[3]))
printf("Rango: de %d.%d.%d.%d a %d.%d.%d.%d\n", IP[0] & MS[0], IP[1] & MS[1], IP[2] & MS[2], (IP[3] & MS[3]) + 1, IP[0] | (unsigned char)~(MS[0]), IP[1] | (unsigned char)~(MS[1]), IP[2] | (unsigned char)~(MS[2]), (IP[3] | (unsigned char)~(MS[3])) - 1)
printf("--\n")
True
IP[0] & 16
False
True
IP[0] & 8
True
False
printf(" es: Clase D\n")
printf(" es: Clase E\n")
memset (IP, 0, 4)
memset (MS, 0, 4)

Figure 1. Flowchart of how the instructions are followed

How to Compile

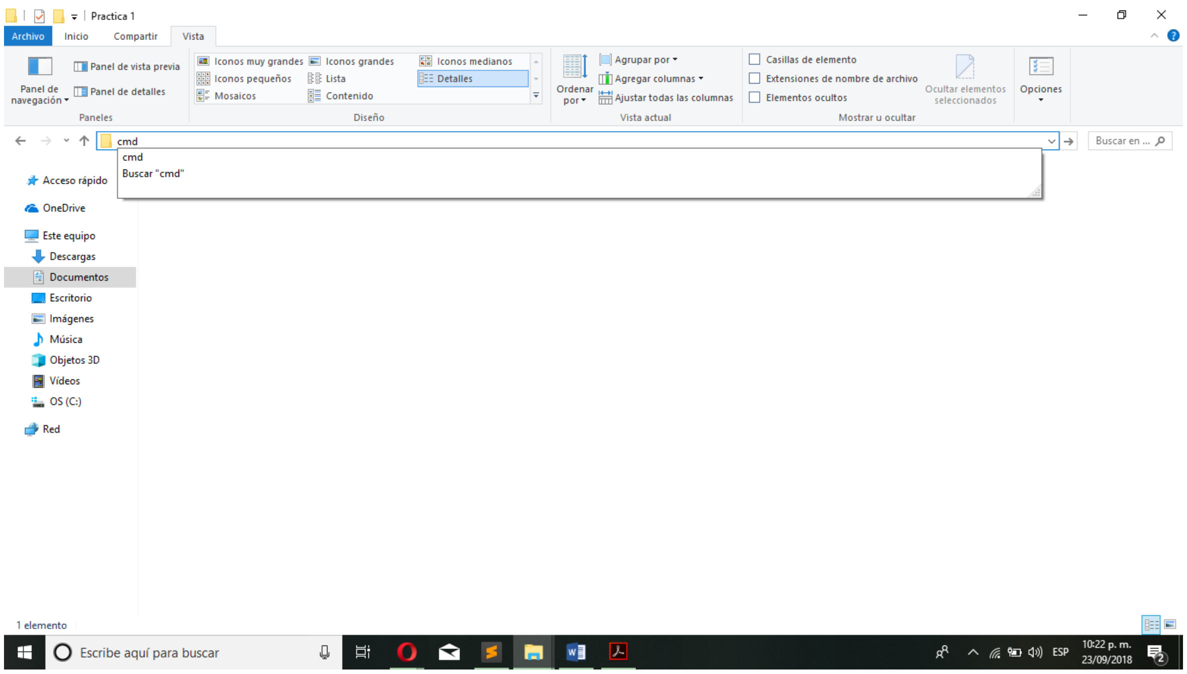


C:\Users\David\Documents\ESCOM\5to Semestre\Redes de Computadora\Practicas\Practica 1

Nombre	Fecha de modifica...	Tipo	Tamaño
clase.c	13/09/2018 07:59 a...	Archivo C	3 KB

10:21 p. m.
23/09/2018

Figure 2.1 Go to the folder where the source code is and go the address bar

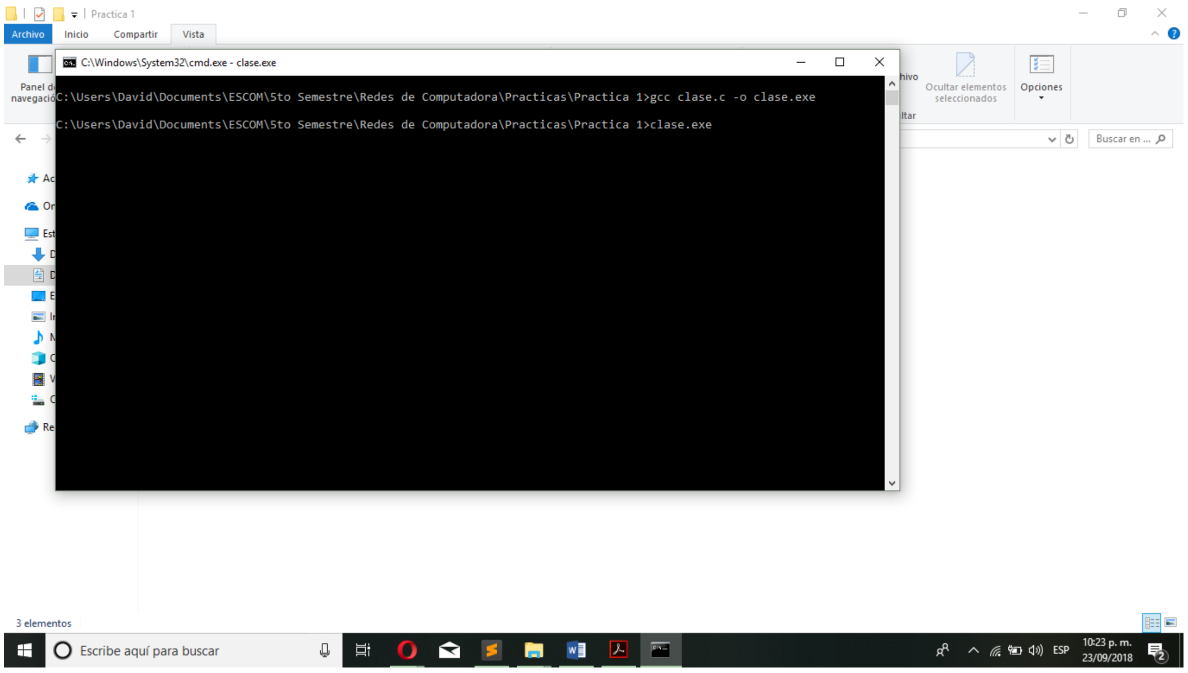


cmd

cmd
Buscar "cmd"

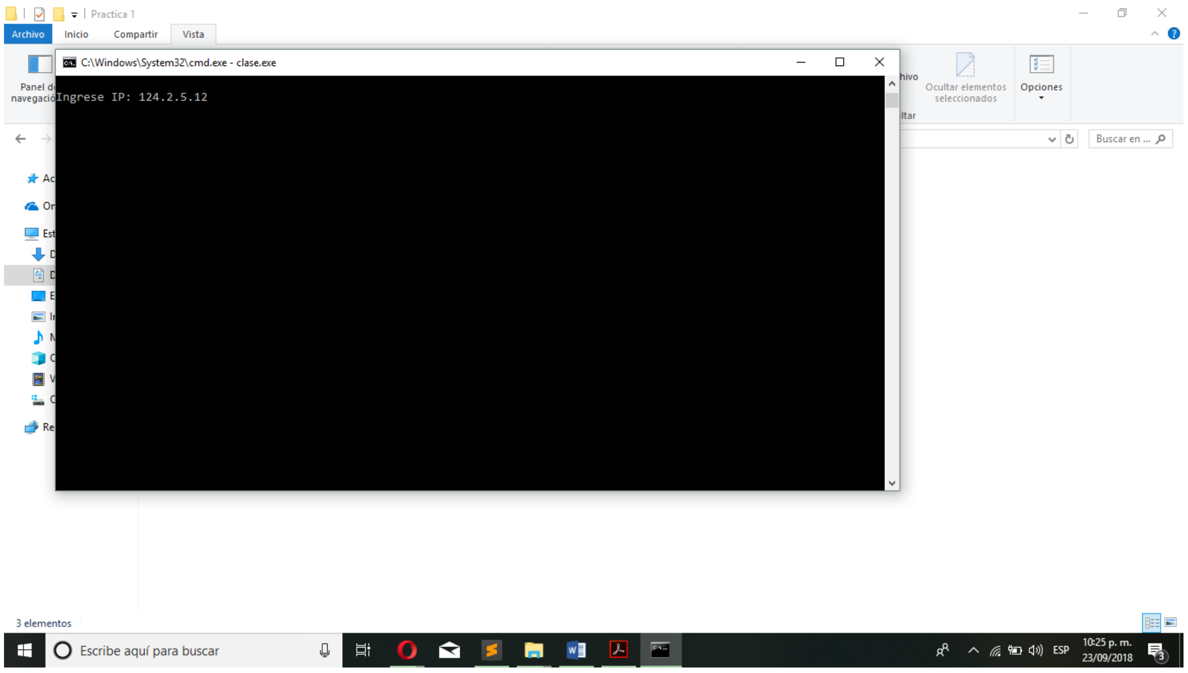
10:22 p. m.
23/09/2018

Figure 2.2 Type cmd and click enter



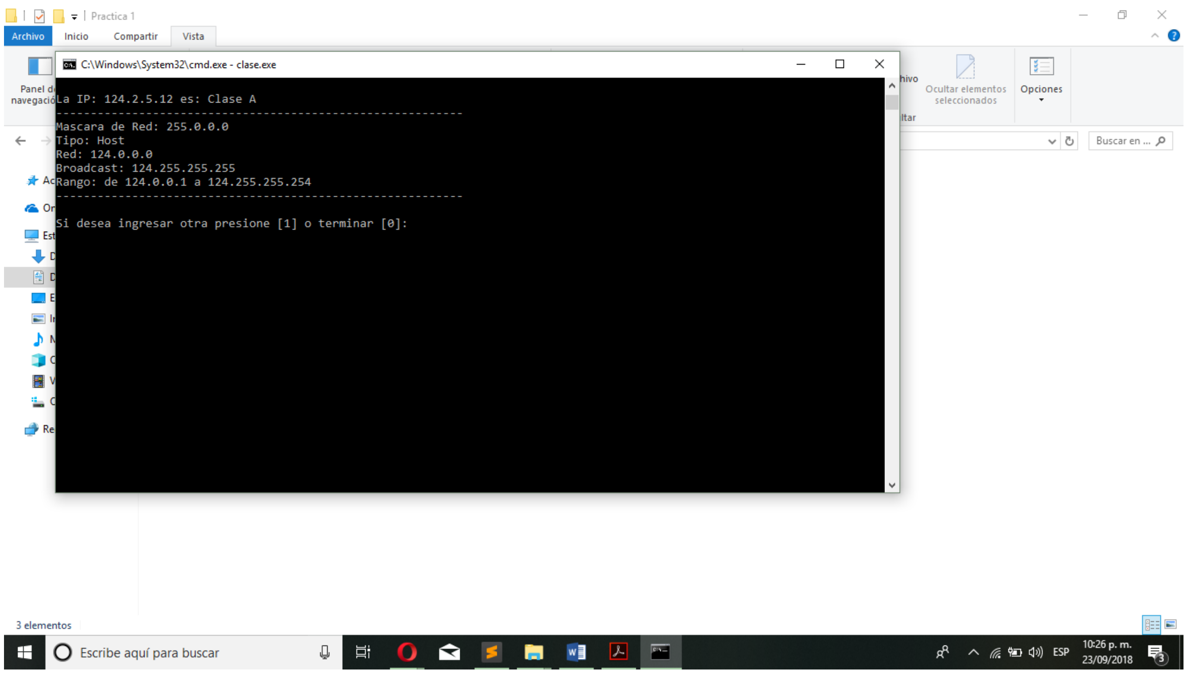
```
C:\Users\David\Documents\ESCOM\5to Semestre\Redes de Computadora\Practicas\Practica 1>gcc clase.c -o clase.exe

C:\Users\David\Documents\ESCOM\5to Semestre\Redes de Computadora\Practicas\Practica 1>clase.exe
```

Figure 2.3 In the Command Line type what's shown



```
Ingrese IP: 124.2.5.12
```

Figure 2.4 Type an IP and click enter

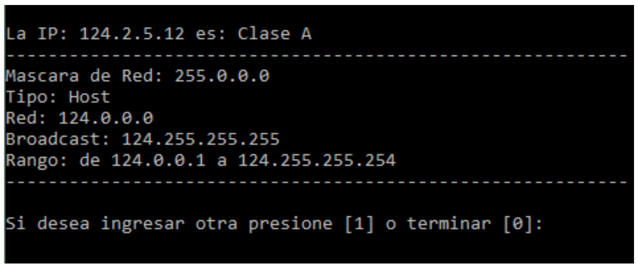


```
La IP: 124.2.5.12 es: Clase A
---------------------------------------------------------------
Mascara de Red: 255.0.0.0
Tipo: Host
Red: 124.0.0.0
Broadcast: 124.255.255.255
Rango: de 124.0.0.1 a 124.255.255.254
---------------------------------------------------------------

Si desea ingresar otra presione [1] o terminar [0]:
```

Figure 2.5 It will show the information mentioned before

Results:

What the screen printed



```
La IP: 124.2.5.12 es: Clase A
---------------------------------------------------------------
Mascara de Red: 255.0.0.0
Tipo: Host
Red: 124.0.0.0
Broadcast: 124.255.255.255
Rango: de 124.0.0.1 a 124.255.255.254
---------------------------------------------------------------

Si desea ingresar otra presione [1] o terminar [0]:
```

Figure 3.1 End of the Execution

Memory Map

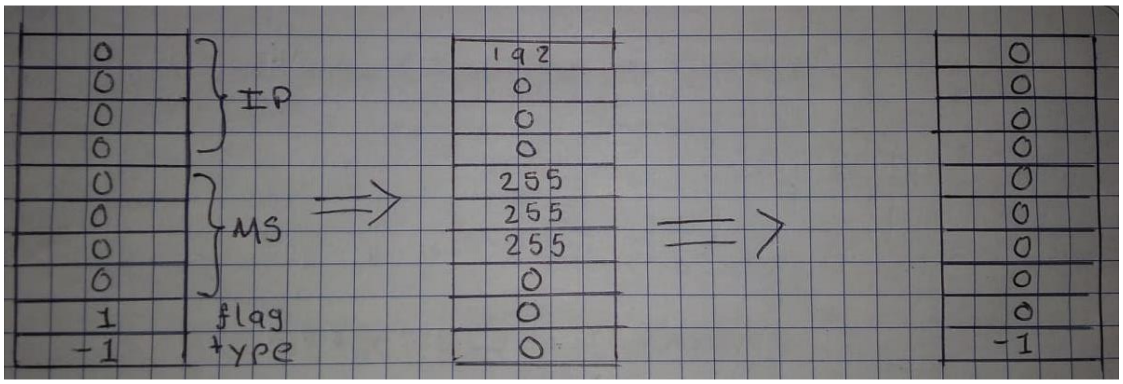


Figure 4.2 Memory use during the run of the program

Discussion:

I found that is not necessary to use the normal logical operators to compare values, actually if we see number in its binary representation, it’s easier to make comparisons between values and the analyzing the information same as bits, and not as decimal values. Maybe the program won’t run exactly as it did here in other platforms, the logic it’s the same and it will help us built other applications easier.

Conclusions:

Although I wasn’t familiarized with bitwise operators, well, I kind of was, but I’ve never got my head around them, until now. I think they’re pretty useful and help us make our programs more efficient and thus, to execute faster. The recurrent problem I had was to only use these bitwise operators, since there where logical operations that I couldn’t understand how to put a use of them, so I had to give up and use normal, slower operator. Other than that, it wasn’t that hard being that I had experience using the programming language C and that I’d learnt the themes I needed.

References:

- **Interserver Tips.** (2018). *Types, Features and Classes of IP Address - Interserver Tips*. [online] Available at: https://www.interserver.net/tips/kb/types-features-classes-ip-address/ [Accessed 22 Sep. 2018].
- **Es.paessler.com.** (2018). *Overview on IP addresses and IP classes*. [online] Available at: https://www.es.paessler.com/info/ip_address_basics_ii [Accessed 22 Sep. 2018].

Code

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main (void)
{
        unsigned char IP[4] = {0}; //IP
        unsigned char MS[4] = {0}; //Máscara de Red
        unsigned char flag = 1, type = -1;

        while (flag)
        {
                system("clear");
                printf("\nIngrese IP: ");
                scanf("%hhu.%hhu.%hhu.%hhu", &IP[0], &IP[1], &IP[2], &IP[3]); //Se lee la IP en
el formato n.n.n.n

                system("clear");
                printf("\nLa IP: %hhu.%hhu.%hhu.%hhu", IP[0], IP[1], IP[2], IP[3]);

                if (IP[0] & 128)
                {
                        if (IP[0] & 64)
                        {
                                if (IP[0] & 32)
                                {
                                        if (IP[0] & 16)
                                        {
                                                if (IP[0] & 8)
                                                {
                                                        printf(" es: Clase E\n");
                                                        goto classE;
                                                }
                                                else
                                                {
                                                        printf(" es: Clase E\n");
                                                        goto classE;
                                                }
                                        }
                                        else
                                        {
                                                printf(" es: Clase D\n");
                                                goto classD;
                                        }
                                }
                                else
                                {
                                        printf(" es: Clase C\n");
                                        MS[0] = MS[1] = MS[2] = ~MS[0];
                                        if (IP[3] == (unsigned char)0)
                                                type = 0;
                                        else if (IP[3] == (unsigned char)255)
                                                type = 1;
                                        else
                                                type = 2;
                                }
                        }
                        else
                        {
                                printf(" es: Clase B\n");
                                MS[0] = MS[1] = ~MS[0]; //Se inicializa la máscara de red
                                if (IP[2] == IP[3] && IP[3] == (unsigned char)0) //Revisa que tipo
de IP es
                                        type = 0;
                                else if (IP[2] == IP[3] && IP[3] == (unsigned char)255)
                                        type = 1;
                                else
                                        type = 2;
                        }
                }
                else
                {
                        printf(" es: Clase A\n");
                        MS[0] = ~MS[0]; //Se inicializa la máscara de red
                        if (IP[1] == IP[2] && IP[2] == IP[3] && IP[3] == (unsigned char)0)
//Revisa que tipo de IP es
                                type = 0;
                        else if (IP[1] == IP[2] && IP[2] == IP[3] && IP[3] == (unsigned char)255)
                                type = 1;
                        else
                                type = 2;



                }

                printf("------------------------------------------------------------\n");
                printf("Mascara de Red: %d.%d.%d.%d\n", MS[0], MS[1], MS[2], MS[3]);
                printf("Tipo: ");
                if (type == (unsigned char)0) //Si es de Red
                {
                        printf("Red\n");
                        printf("Broadcast: %d.%d.%d.%d\n",  IP[0] | (unsigned char)~(MS[0]), IP[1]
| (unsigned char)~(MS[1]), IP[2] | (unsigned char)~(MS[2]), IP[3] | (unsigned char)~(MS[3]));
                }
                if (type == (unsigned char)1)//Si es de Broadcast
```

```
		{
			printf("Broadcast\n");
			printf("Red: %d.%d.%d.%d\n", IP[0] & MS[0], IP[1] & MS[1], IP[2] & MS[2],
IP[3] & MS[3]);
		}
		if (type == (unsigned char)2) //Si es de Host
		{
			printf("Host\n");
			printf("Red: %d.%d.%d.%d\n", IP[0] & MS[0], IP[1] & MS[1], IP[2] & MS[2],
IP[3] & MS[3]);
			printf("Broadcast: %d.%d.%d.%d\n",  IP[0] | (unsigned char)~(MS[0]), IP[1]
| (unsigned char)~(MS[1]), IP[2] | (unsigned char)~(MS[2]), IP[3] | (unsigned char)~(MS[3]));
		}
		printf("Rango: de %d.%d.%d.%d a %d.%d.%d.%d\n", IP[0] & MS[0], IP[1] & MS[1],
IP[2] & MS[2], (IP[3] & MS[3]) + 1, IP[0] | (unsigned char)~(MS[0]), IP[1] | (unsigned
char)~(MS[1]), IP[2] | (unsigned char)~(MS[2]), (IP[3] | (unsigned char)~(MS[3])) - 1);

		printf("-------------------------------------------------------------\n");


		classD:
		classE:
		memset (IP, 0, 4); //Limpia la variable de la IP
		memset (MS, 0, 4); //Limpia la variable de la mascara
		printf("\nSi desea ingresar otra presione [1] o terminar [0]: ");
		scanf("%hhu", &flag); //Lee la respuesta de repeticion
	}

	return 0;
}
```