INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

Redes de computadoras

Práctica 1 “Clasificación de direcciones IP”

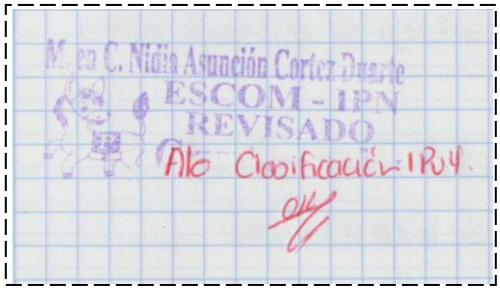
ALUMNO:

ZEPEDA FLORES ALEJANDRO DE JESÚS

PROFESOR:

M. en C. NIDIA ASUNCIÓN CORTEZ DUARTE

SEPTIEMBRE 2018



Índice

Objetivo

Desarrollar un programa en lenguaje C que le permita al usuario introducir una dirección IP. Verificar si es una dirección válida o inválida; si es válida, mostrar en pantalla la clase, el tipo, la máscara, la IP madre, IP broadcast y el rango de host de la dirección IP original.

Material y equipo

- Equipo de cómputo con el sistema operativos Windows
- Entorno de desarrollo (Bloc de Notas)
- Compilador para programas en C (GCC)

Marco teórico

Una dirección IP consiste en 32 bits que normalmente se expresan en forma decimal, en cuatro grupos de tres dígitos separados por puntos. Cada número estará entre cero y 255. Cada número entre los puntos en una dirección IP se compone de 8 dígitos binarios (00000000 a 11111111); los escribimos en la forma decimal para hacerlos más comprensibles, pero hay que tener bien claro que la red entiende sólo direcciones binarias.

Clases de direcciones IP

a) **Redes de clase A:** son aquellas redes que precisan un gran número de direcciones IP, debido al número de host que comprenden. A este tipo de redes se les asigna un rango de direcciones IP identificado por el primer octeto de la IP, de tal forma que disponen de los otros 3 octetos siguientes para asignar direcciones a su host.

b) **Redes de clase B**: son redes que precisan un número de direcciones IP intermedio para conectar todos sus hosts con Internet. A este tipo de redes se les asigna un rango de direcciones IP identificado por los dos primeros octetos de la IP de tal forma que disponen de los otros 2 octetos siguientes para asignar direcciones a sus hosts.

c) **Redes de clase C:** son redes que precisan un número de direcciones IP pequeño para conectar sus hosts con Internet. A este tipo de redes se les asigna un rango de direcciones IP identificado por los tres primeros octetos de la IP, de tal forma que disponen de un sólo octeto para asignar direcciones a sus hosts.

Debido a que cada clase utiliza diferente número de octetos, su tamaño también es diferente. Para conocer el tamaño de cada clase checar la Tabla1.

Clase	Tamaño de porción
A	2,147,483,648 = 2GB
B	1,073,741,824 = 1GB
C	536,870,912 = 512MB
D	268,435,456 = 256MB
E	268,435,456 = 256MB

Tabla 1. Tamaño de porción

Gráfica 1. Tamaño de porción

Clase	Direcciones disponibles		Cantidad de redes	Cantidad de hosts	Aplicación
	Desde	Hasta			
A	0.0.0.0	127.255.255.255	128	16,777,214	Redes grandes
B	128.0.0.0	192.255.255.255	16,384	65,534	Redes medianas
C	192.0.0.0	223.255.255.255	2,097,154	254	Redes pequeñas
D	224.0.0.0	239.255.255.255	No aplica	No aplica	Multicast
E	240.0.0.0	255.255.255.255	No aplica	No aplica	Investigación

Tabla 2. Características de las direcciones IP

Resultados

El programa fue desarrollado de manera exitosa, como primer paso solicitamos al usuario introducir una dirección de IP, en caso de ser válida, procedemos desplegar las características correspondientes de la dirección introducida.

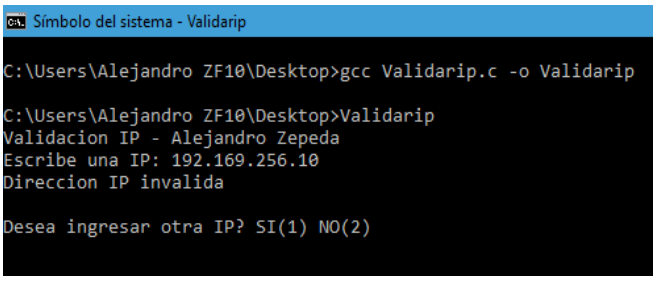


Imagen 1. IP inválida

Nuestro primer filtro funciona de manera correcta, ya que detecta si el usuario introduce una dirección de IP errónea e imprime un mensaje de error, de igual manera, brinda al usuario la oportunidad de introducir otra dirección IP.

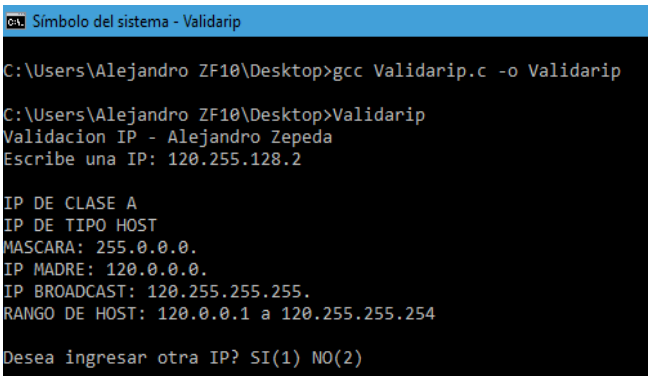


Figura 2. IP válida

Ahora introducimos una dirección de IP válida y vemos como despliega las características correspondientes de forma correcta.

Es bueno mencionar que para la implementación de este programa, se tuvo que hacer uso de variables enteras para la validación.

Mapa de memoria inicial

0	0	0	0	0	0	0	0	Char i = 0;
0	0	0	0	0	0	0	0	Char j = 0
0	0	0	0	0	0	0	0	Char flag = 0;
0	0	0	0	0	0	0	0	Char repeat = 0;
x	x	x	x	x	x	x	x	Char ip[4];
x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x	x	Char masc[4];
x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x	x	Int aux[4];
x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x	x	

Tabla 1. Mapa de memoria inicial

Prueba de escrito (150.0.0.0)

0	0	0	0	0	0	1	1	Char i = 3;
0	0	0	0	0	0	1	1	Char j = 3;
0	0	0	0	0	0	0	1	Char flag = 1;
0	0	0	0	0	0	0	X	Char repeat;
1	0	0	1	0	1	1	0	Char ip[0] = 150;
0	0	0	0	0	0	0	0	Char ip[1] = 0;
0	0	0	0	0	0	0	0	Char ip[2] = 0;
0	0	0	0	0	0	0	0	Char ip[3] = 0;
1	1	1	1	1	1	1	1	Char masc[0] = 255;
1	1	1	1	1	1	1	1	Char masc[1] = 255;
0	0	0	0	0	0	0	0	Char masc[2] = 0;
0	0	0	0	0	0	0	0	Char masc[3] = 0;

Tabla 2. Prueba de escritorio

A diferencia del mapa de memoria, notará que en la prueba de escritorio no aparece el arreglo int aux[4], esto se debe a que es solicitado con malloc y durante el proceso de validación, si es exitosa, liberamos el espacio de memoria ocupado por el arreglo.

Conclusiones

El desarrollo de esta práctica es interesante porque en primera, el uso de entorno de desarrollo esta prohibido y hacerlo en Bloc de nota es algo complejo porque no trae todas las ayudas como autocompletado, línea de código y detección de errores. Ádemas, el uso de variables enteras también esta prohibido para reducir el desperdicio de memoria.

La lección importante adquirida durante el desarrollo de la práctica es que programar a nivel de bits es interesante y más efectivo, ádemas sirvió para enteder de mejor manera la forma en que se utilizan las operaciones de bits.

Código

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

void imprimir_ip(unsigned char ip[]){
    char i = 0;
    printf("%d",ip[i]);
    for(i=1; i<4 ;i++)
        printf(".%d",ip[i]);
}

char tipo_clase(unsigned char ip[]){
    if(ip[0]&128)
        if(ip[0]&64)
            if(ip[0]&32)
                if(ip[0]&16)
                    return 5; //CLASE E
                else
                    return 4; //CLASE D
            else
                return 3; //CLASE C
        else
            return 2; //CLASE B
    else
        return 1; //CLASE A
}

void tipo_ip(unsigned char ip[]){
    switch(tipo_clase(ip)){
        case 1:
             if(ip[1]==0 && ip[2]==0 && ip[3]==0)
                printf("IP DE TIPO RED");
             else if(ip[1]==255 && ip[2]==255 && ip[3]==255)
                printf("IP DE TIPO BROADCAST");
             else
                printf("IP DE TIPO HOST");
        break;
        case 2:
            if(ip[2]==0 && ip[3]==0)
                printf("IP DE TIPO RED");
            else if(ip[2]==255 && ip[3]==255)
                printf("IP DE TIPO BROADCAST");
            else
                printf("IP DE TIPO HOST");
        break;
        case 3:
            if(ip[3]==0)
                printf("IP DE TIPO RED");
            else if(ip[3]==255)
                printf("IP DE TIPO BROADCAST");
            else
                printf("IP DE TIPO HOST");
        break;
        default:
            break;
    }
}


int main(void){

    unsigned char i = 0, j = 0, k = 0, flag= 0, repeat = 0;
    do{
        fflush(stdin);
        unsigned char ip[4] = {};
        unsigned char masc[4] = {};
        unsigned short int aux[4] = {};
        printf("Validacion IP - Alejandro Zepeda\nEscribe una IP: ");
        scanf("%d.%d.%d.%d",&aux[0],&aux[1],&aux[2],&aux[3]);
        for(i = 0; i < 4; i++){
            if(aux[i]<=255){
                ip[i]=aux[i];
            }
            else{
                printf("Direccion IP invalida\n");
                flag = 1;
            }
        }
        if(flag == 0){
            switch(tipo_clase(ip)){
                case 1:
                    printf("\nIP DE CLASE A\n"); tipo_ip(ip);
                    masc[0] = 255; masc[1] = 0; masc[2] = 0; masc[3] = 0;
                    printf("\nMASCARA: ");
                    for(i = 0; i < 4; ++i)
                        printf("%d.",masc[i]); i = 0;
                    printf("\nIP MADRE: ");
                    for(i = 0; i < 4; ++i)
                        printf("%d.",ip[i]&masc[i]);
```

```
                        printf("%d.",ip[i]&masc[i]);
                    for(j = 0; j < 4; ++j)
                        masc[j]=~masc[j];
                    printf("\nIP BROADCAST: ");
                    for(k = 0; k < 4; ++k)
                        printf("%d.", ip[k]|masc[k]);
                    for(j = 0; j < 4; ++j)
                        masc[j]=~masc[j];
                    printf("\nRANGO DE HOST: %u.%u.%u.%u a", ip[0]&masc[0],ip[1]&masc[1],ip[2]&masc[2],(ip[3]&masc[3])+1);
                    for(i = 0; i<4; i++)
                        masc[i]=~masc[i];
                    printf(" %u.%u.%u.%u\n", ip[0]|masc[0],ip[1]|masc[1],ip[2]|masc[2],(ip[3]|masc[3])-1);
                break;
                case 2:
                    printf("\nIP DE CLASE B\n"); tipo_ip(ip);
                    masc[0] = 255; masc[1] = 255; masc[2] = 0; masc[3] = 0;
                    printf("\nMASCARA: ");
                    for(i = 0; i < 4; ++i)
                        printf("%d.",masc[i]);i = 0;
                    printf("\nIP MADRE: ");
                    for(i = 0; i < 4; ++i)
                        printf("%d.",ip[i]&masc[i]);
                    for(j = 0; j < 4; ++j)
                        masc[j]=~masc[j];
                    printf("\nIP BROADCAST: ");
                    for(k = 0; k < 4; ++k)
                        printf("%d.", ip[k]|masc[k]);
                    for(j = 0; j < 4; ++j){masc[j]=~masc[j];}
                    printf("\nRANGO DE HOST: %u.%u.%u.%u a", ip[0]&masc[0],ip[1]&masc[1],ip[2]&masc[2],(ip[3]&masc[3])+1);
                    for(i = 0; i<4; i++)
                        masc[i]=~masc[i];
                    printf(" %u.%u.%u.%u\n", ip[0]|masc[0],ip[1]|masc[1],ip[2]|masc[2],(ip[3]|masc[3])-1);
                break;
                case 3:
                    printf("\nIP DE CLASE C\n"); tipo_ip(ip);
                    masc[0] = 255; masc[1] = 255; masc[2] = 255; masc[3] = 0;
                    printf("\nMASCARA: ");
                    for(i = 0; i < 4; ++i)
                        printf("%d.",masc[i]); i = 0;
                    printf("\nIP MADRE: ");
                    for(i = 0; i < 4; ++i)
                        printf("%d.",ip[i]&masc[i]);
                    for(j = 0; j < 4; ++j)
                        masc[j]=~masc[j];
                    printf("\nIP BROADCAST: ");
                    for(k = 0; k < 4; ++k)
```

```
                  for (j = 0; j < 4; ++j)
                      masc[j]=~masc[j];
                  printf("\nRANGO DE HOST: %u.%u.%u.%u a", ip[0]&masc[0],ip[1]&masc[1],ip[2]&masc[2],(ip[3]&masc[3])+1);
                  for(i = 0; i<4; i++)
                      masc[i]=~masc[i];
                  printf(" %u.%u.%u.%u\n", ip[0]|masc[0],ip[1]|masc[1],ip[2]|masc[2],(ip[3]|masc[3])-1);
              break;
              case 4:
                  printf("\nIP DE CLASE D");
                  printf("\nUSO: MULTICAST\n");
              break;
              case 5:
                  printf("\nIP DE CLASE E");
                  printf("\nUSO: EXPERIMENTAL\n");
              break;
          }
      }
      printf("\nDesea ingresar otra IP? SI(1) NO(2)  ");
      scanf("%d",&repeat);printf("\n");
  }while(repeat == 1);
  return 0;
}
```

Referencias

- Isidoro García. (2010, Mayo 8). Direccionamiento IP en el aula de informática. Temas para la educación, 8, S/N.
- Daniel Morató. (2004). Direccionamiento IP. Septiembre 22, 2018, de Universidad Pública de Navarra Sitio web: https://www.tlm.unavarra.es/~daniel/docencia/lpr/lpr04_05/slides/clase5-DirecIP_1.pdf