INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

Redes de computadoras

Práctica 2 “Subnetting”

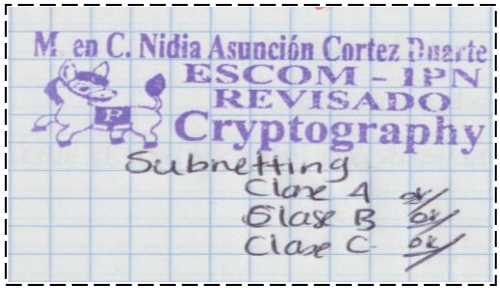
ALUMNO:

ZEPEDA FLORES ALEJANDRO DE JESÚS

PROFESOR:

M. en C. NIDIA ASUNCIÓN CORTEZ DUARTE

OCTUBRE 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 un nuevo menú donde tenga la opción de como realizar el subnetting, ya sea por número de subredes, cantidad de host o máscara de subred.

Material y equipo

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

Marco teórico

Una dirección IP es un direccionamiento usado para identificar únicamente un dispositivo en una red del IP. El direccionamiento se compone de 32 bits binarios, que pueden ser divisibles en una porción de la red y recibir la porción con la ayuda de una máscara de subred. Los 32 bits binarios se dividen en cuatro octetos (1 octeto = 8 bits). Cada octeto se convierte a decimal y se separa con un punto. Por esta razón, se dice que una dirección IP se expresa en formato decimal con puntos. El valor en cada octeto posee un rango decimal de 0 a 255 o binario de 00000000 a 11111111.

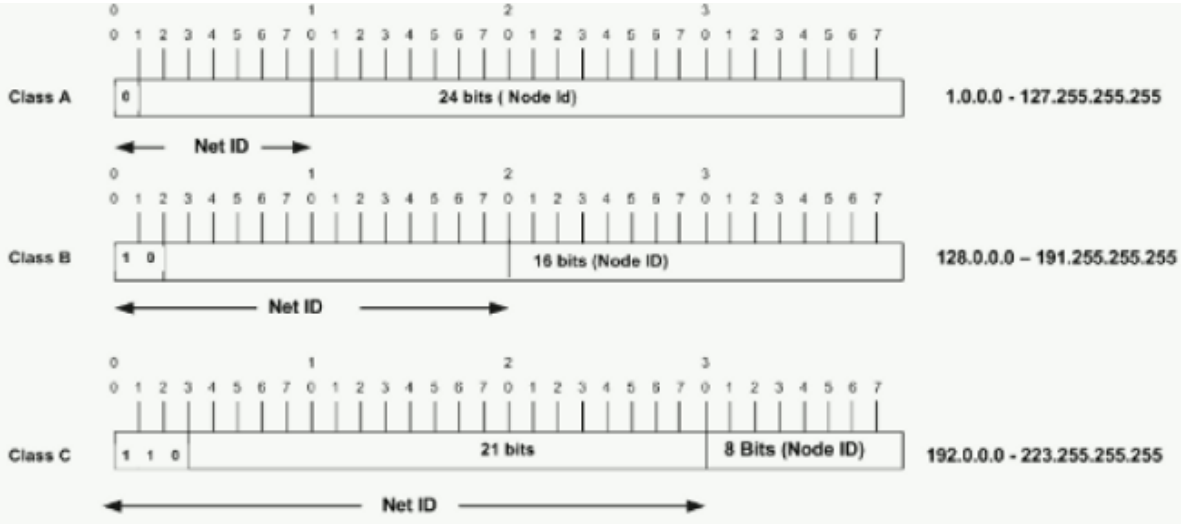


Imagen 1. Porción de red de una dirección IP

Máscaras de red

Una máscara de red ayuda a saber qué parte de la dirección identifica la red y qué parte de la dirección identifica el nodo. Las redes de la clase A, B, y C tienen máscaras predeterminadas, también conocidas como máscaras naturales, como se muestra aquí:

```
Class A: 255.0.0.0
Class B: 255.255.0.0
Class C: 255.255.255.0
```

- **Porción de Red:**
  En el caso que la máscara sea por defecto, una dirección con Clase, la cantidad de bits “1” en la porción de red, indican la dirección de red, es decir, la parte de la dirección IP que va a ser común a todos los hosts de esa red.

  En el caso que sea una máscara adaptada, la parte de la máscara de red cuyos octetos sean todos bits “1” indican la dirección de red y va a ser la parte de la dirección IP que va a ser común a todos los hosts de esa red, los bits “1” restantes son los que en la dirección IP se van a modificar para generar las diferentes subredes y van a ser común solo a los hosts que pertenecen a esa subred.

- **Porción de Host:**
  La cantidad de bits "0" en la porción de host de la máscara, indican que parte de la dirección de red se usa para asignar direcciones de host, es decir, la parte de la dirección IP que va a variar según se vayan asignando direcciones a los hosts.

Subnetting

La conexión en subredes permite crear múltiples redes lógicas que existen dentro de una red única Clase A, B o C. Si no crea una subred, solamente podrá utilizar una red de la red de Clase A, B o C, lo que es poco realista.

Cada link de datos de una red debe tener una identificación de red única, siendo cada nodo de ese link miembro de la misma red. Si divide una red principal (clase A, B, o C) en subredes menores, podrá crear una red de subredes interconectadas. Cada link de datos de esta red tendrá entonces una identificación única de red/subred.

Para la subred una red, amplía a la máscara natural con algunos de los bits de la porción del ID del host del direccionamiento para crear una identificación de la red secundario, por ejemplo, dada una red clase C de 204.17.5.0 que tenga una máscara natural de 255.255.255.0, usted puede crear las subredes de este modo:

```
204.17.5.0 -      11001100.00010001.00000101.00000000
255.255.255.224 - 11111111.11111111.11111111.11100000
                  ------------------------------|sub|-----
```

Extendiendo la máscara para que sea 255.255.255.224, ha tomado tres bits de la parte original del host de la dirección y los ha utilizado para crear subredes. Con estos tres bits, es posible crear ocho subredes. Con los cinco bits de ID de host restantes, cada subred puede tener hasta 32 direcciones de host, 30 de las cuales pueden asignarse realmente a un dispositivo ya que las ID del host con todos ceros o todos unos no están permitidas (es muy importante recordar esto).

Resultados

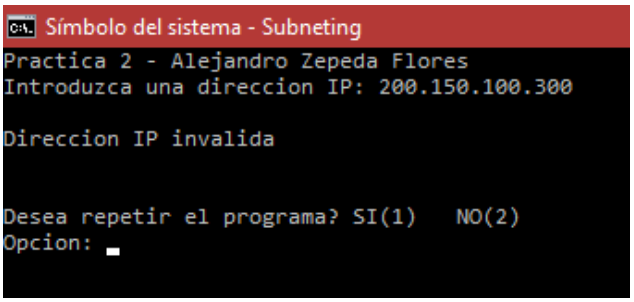


Figura 1. Validación de IP

Reutilizamos el filtro de la práctica anterior para la validación de direcciones IP, por lo que no representa mayor problema.

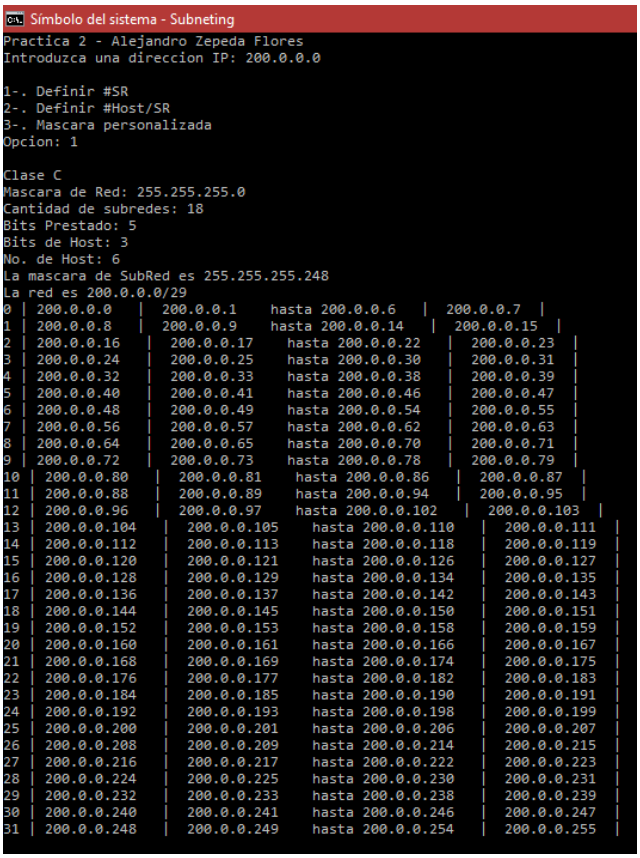


Figura 2. Subnetting de dirección de clase C

Esta implementación de la figura 2, muestra la validación de direcciones IP de clase C; donde utilizamos la definición de subredes y los números de subred para establecer el subnettig correspondiente. La división es correcta, además de la validación de la dirección IP.

En el ejemplo de la figura 2, utilizamos una división por cantidades de subredes, aunque muestra 32 subredes (porque el siguiente múltiplo de 2 a 18 es 32) hace la división correcta para 18 subredes.

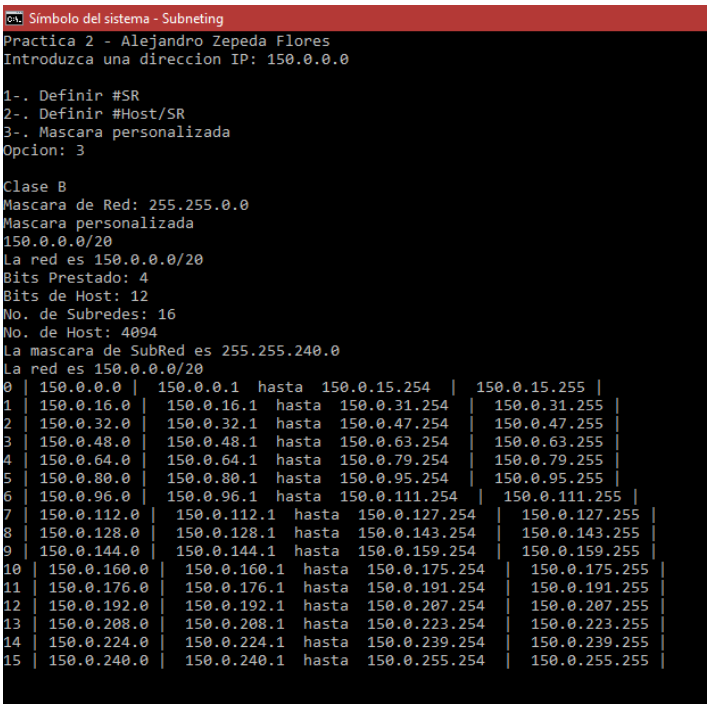


Figura 3. Subnetting de dirección de clase B

Como se muestra en la figura 3, se aplica subnetting a una dirección IP de clase B donde se divide por máscara personalizada. Podemos ver que en el ejemplo utilizamos una máscara de /20, por lo que al utilizar una dirección de clase B, toma 4 bits prestados, pudiendo generar hasta 16 subredes.

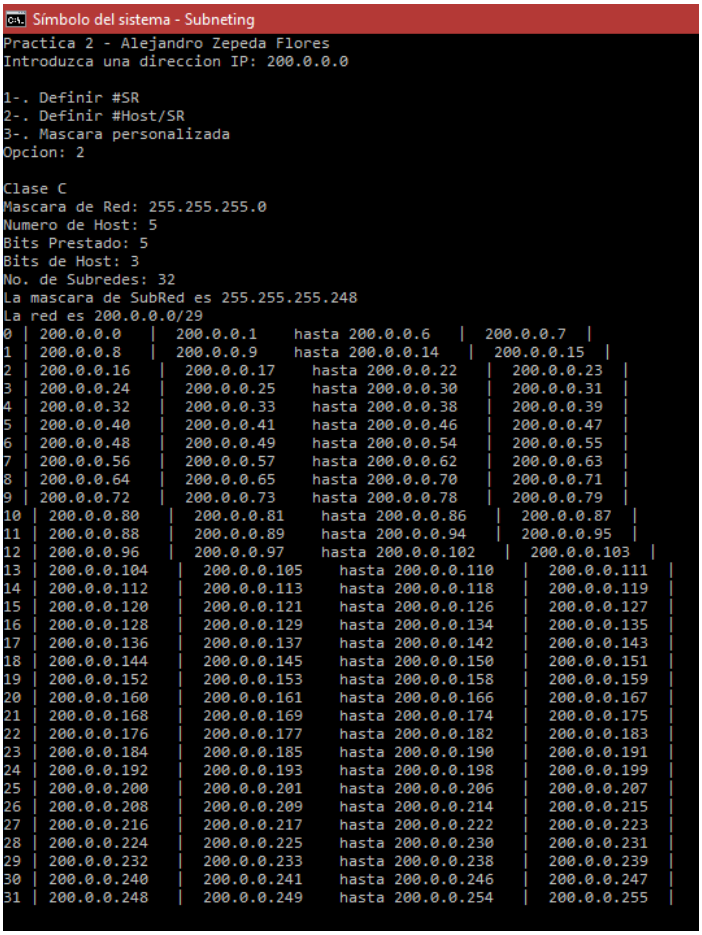


Figura 4. Subnetting de dirección de clase C

Como se muestra en la figura 4, se aplica subnetting a una dirección IP de clase C donde se divide por número de host. Podemos ver que en el ejemplo utilizamos una división por 5 hosts (considerando las dos IP para red y broadcast), por lo que el subnetting está bien aplicado.

MAPA DE MEMORIA INICIAL

X	X	X	X	X	X	X	X	Int * ip;
X	X	X	X	X	X	X	X	
X	X	X	X	X	X	X	X	
X	X	X	X	X	X	X	X	
0	0	0	0	0	0	0	0	Unsigned char ark = 0;
0	0	0	0	0	0	0	0	Unsigned char i = 0;
0	0	0	0	0	0	0	0	Unsigned char val = 0;
0	0	0	0	0	0	0	0	Unsigned char option = 0;
0	0	0	0	0	0	0	0	Unsigned char repeat = 0;
0	0	0	0	0	0	0	0	Unsigned char bitP = 0;
0	0	0	0	0	0	0	0	Unsigned char bitH = 0;
0	0	0	0	0	0	0	0	Unsigned char noHost = 0;
0	0	0	0	0	0	0	0	Unsigned char MaskP = 0;
X	X	X	X	X	X	X	X	Unsigned 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	Unsigned char MR[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	Unsigned char MRSR[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	

Tabla 1. Mapa de memoria inicial

PRUEBA DE ESCRITORIO

1	1	1	1	1	1	1	1	Unsigned char ark = 255;
0	0	0	0	0	0	0	1	Unsigned char i = 1;
0	0	0	0	0	0	0	0	Unsigned char val = 0;
0	0	0	0	0	0	0	1	Unsigned char option = 1;
0	0	0	0	0	0	0	1	Unsigned char repeat = 1;
0	0	0	0	0	0	0	1	Unsigned char bitP = 1;
0	0	0	0	0	0	0	1	Unsigned char bitH = 1;
0	0	1	0	0	0	0	0	Unsigned char noHost = 32;
0	0	0	0	1	0	1	0	Unsigned char MaskP = 10;
1	1	0	0	0	1	0	0	Unsigned char IP[0] = 100;
0	0	0	0	0	0	0	0	Unsigned char IP[1] = 000;
0	0	0	0	0	0	0	0	Unsigned char IP[2] = 000;
0	0	0	0	0	0	0	0	Unsigned char IP[3] = 000;
1	1	1	1	1	1	1	1	Unsigned char MR[0] = 255;
0	0	0	0	0	0	0	0	Unsigned char MR[1] = 000;
0	0	0	0	0	0	0	0	Unsigned char MR[2] = 000;
0	0	0	0	0	0	0	0	Unsigned char MR[3] = 000;
1	1	1	1	1	1	1	1	Unsigned char MRSR[0] = 255;
1	1	0	0	0	0	0	0	Unsigned char MRSR[1] = 192;
0	0	0	0	0	0	0	0	Unsigned char MRSR[2] = 000;
0	0	0	0	0	0	0	0	Unsigned char MRSR[3] = 000;

Tabla 2. Prueba de escritorio

Conclusiones

La división de subredes por subnetting mejora nuestra eficiencia en la asignación de direcciones permitiendo no utilizar una nueva dirección Clase C o Clase B cada vez que necesitemos agregar una nueva red física. Además, gracias a este método, el desperdicio de direcciones IP es menor o se reduce porque el objetivo principal es optimizar las subredes para asignar direcciones IP.

Código

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

void imprime(unsigned char IP[], unsigned char MRSR[]);
void imprimeB(unsigned char IP[]);
void imprimeA(unsigned char IP[]);
int tipo_clase(unsigned char ip[]);
int validar_ip(unsigned int ip[]);

unsigned int ark; char val;
unsigned char IP[4], MR[4], MRSR[4], MaskP;
int i, option, repeat, noRedes, bitP, bitH, noHost, * ip;

int main(void){
    do{
        system("cls");
        ip = (int *)malloc(sizeof(int)*4);
        printf("Practica 2 - Alejandro Zepeda Flores\n");
        printf("Introduzca una direccion IP: "),
        scanf("%d.%d.%d.%d", &ip[0], &ip[1], &ip[2], &ip[3]);
        if(validar_ip(ip)){
            free(ip);
            printf("\n1-. Definir #SR\n2-. Definir #Host/SR\n3-. Mascara personalizada\n");
            printf("Opcion: "); scanf("%d", &option);
            switch(tipo_clase(IP)){
                case 1:
                    printf("\nClase A\n");
                    printf("Mascara de Red: 255.0.0.0\n");
                    MR[1]=MR[2]=MR[3]=MRSR[1]=MRSR[2]=MRSR[3]=0;
                    MR[0]=MRSR[0]=255;
                    switch(option){
                        case 1:
                            i = 1;
                            while(i!=0){
                                printf("Cantidad de Subredes:");
                                scanf("%d", &noRedes);
                                if(noRedes<2 || noRedes>4194304)
                                    printf("Numero de redes invalido\n" );
                                else
                                  i = 0;
                            }
                            bitP = 1;
                            while(pow(2,bitP)<noRedes)
                                bitP++;
                            printf("Bits Prestado: %d\n", bitP );
                            printf("Bits de Host: %d\n", 24-bitP );
                            noHost = pow(2,24-bitP);
                            printf("No. de Host: %d\n", noHost-2 );
                            imprimeA(IP);
                        break;
                        case 2:
                            i = 1;
                            while(i!=0){
                                printf("Numero de Host");
                                scanf("%d", &noHost);
                                if(noHost<2 || noHost>8388608)
                                        printf("Numero de Host invalid\n" );
                                else
                                    i=0;
                            }
                            bitH = 1;
                            while(pow(2,bitH)<(noHost+2))
                                bitH++;
                            bitP = 24-bitH;
                            noHost = pow(2,24-bitP);
                            printf("Bits Prestado: %d\n", bitP );
                            printf("Bits de Host: %d\n", bitH );
                            noRedes = pow(2,bitP);
                            printf("No. de Subredes: %d\n", noRedes );
                            imprimeA(IP);
                        break;
                        case 3:
                            i = 1;
                            while(i!=0){
                                printf("Mascara personalizada\n");
                                printf("%d.%d.%d.%d/", IP[0], IP[1], IP[2], IP[3]);
                                scanf("%d", &MaskP);
                                if(MaskP<8 || MaskP>30)
                                    printf("Mascara invalida\n" );
                                else
                                    i = 0;
                            }
                            printf("La red es %d.%d.%d.%d/%d \n", IP[0], IP[1], IP[2], IP[3], MaskP);
                            bitH = 32-MaskP;
                            bitP = 24-bitH;
```

```
                            bitP = 24-bitH;
                            printf("Bits Prestado: %d\n", bitP );
                            printf("Bits de Host: %d\n", bitH );
                            noRedes = pow(2,bitP);
                            printf("No. de Subredes: %d\n", noRedes );
                            noHost=pow(2,24-bitP);
                            printf("No. de Host: %d\n", noHost-2 );
                            imprimeA(IP);
                        break;
                    }
                break;
                case 2:
                    printf("\nClase B\n");// INICIO CLASE B
                    printf("Mascara de Red: 255.255.0.0\n");
                    MR[2]=MR[3]=MRSR[2]=MRSR[3]=0;
                    MR[0]=MR[1]=MRSR[0]=MRSR[1]=255;
                    switch(option){
                        case 1:
                            i = 1;
                            while(i!=0){
                                printf("Cantidad de subredes: ");
                                scanf("%d", &noRedes);
                                if(noRedes<2 || noRedes>16384)
                                    printf("Numero de redes invalido\n" );
                                else
                                    i = 0;
                            }
                            bitP = 1;
                            while(pow(2,bitP)<noRedes)
                                bitP++;
                            printf("Bits Prestado: %d\n", bitP );
                            printf("Bits de Host: %d\n", 16-bitP );
                            noHost = pow(2,16-bitP);
                            printf("No. de Host: %d\n", noHost-2 );
                            imprimeB(IP);
                        break;
                        case 2:
                            i = 1;
                            while(i!=0){
                                printf("Numero de Host: ");
                                scanf("%d", &noHost);
                                if(noHost<2 || noHost>32768)
                                    printf("Numero de Host invalido\n" );
                                else
                                    i = 0;
                            }
                            bitH = 1;
                            while(pow(2,bitH)<(noHost+2))
                                bitH++;
                            bitP = 16-bitH;
                            noHost = pow(2,16-bitP);
                            printf("Bits Prestado: %d\n", bitP );
                            printf("Bits de Host: %d\n", bitH );
                            noRedes = pow(2,bitP);
                            printf("No. de Subredes: %d\n", noRedes );
                            imprimeB(IP);
                        break;
                        case 3:
                            i = 1;
                            while(i!=0){
                                printf("Mascara personalizada\n");
                                printf("%d.%d.%d.%d/", IP[0], IP[1], IP[2], IP[3]);
                                scanf("%d", &MaskP);
                                if(MaskP<16 || MaskP>30)
                                    printf("Mascara invalida\n" );
                                else
                                    i = 0;
                            }
                            printf("La red es %d.%d.%d.%d/%d \n", IP[0], IP[1], IP[2], IP[3], MaskP);
                            bitH = 32-MaskP;
                            bitP = 16-bitH;
                            printf("Bits Prestado: %d\n", bitP );
                            printf("Bits de Host: %d\n", bitH );
                            noRedes = pow(2,bitP);
                            printf("No. de Subredes: %d\n", noRedes );
                            noHost = pow(2,16-bitP);
                            printf("No. de Host: %d\n", noHost-2 );
                            imprimeB(IP);
                        break;
                    }
                break;
                case 3:
                    printf("\nClase C\n");// INICIO CLASE C
                    printf("Mascara de Red: 255.255.255.0\n");
                    MR[0]=MR[1]=MR[2]=MRSR[0]=MRSR[1]=MRSR[2]=255; MR[3]=0;
                    switch(option){
                        case 1:
                            i = 1;
                            while(i != 0){
                                printf("Cantidad de subredes: "); scanf("%d", &noRedes);
```

```
                                if(noRedes<2 || noRedes>64)
                                    printf("Numero de subredes invalido\n");
                                else
                                    i = 0;
                            }
                            bitP = 1;
                            while(pow(2,bitP)<noRedes)
                                bitP++;
                            printf("Bits Prestado: %d\n", bitP);
                            printf("Bits de Host: %d\n", 8-bitP);
                            noHost = pow(2,8-bitP);
                            printf("No. de Host: %d\n", noHost-2);
                            MRSR[3] = (255-noHost)+1;
                            imprime(IP,MRSR);
                        break;
                        case 2:
                            i = 1;
                            while(i!=0){
                                printf("Numero de Host: ");
                                scanf("%d", &noHost);
                                if(noHost<2 || noHost>124)
                                    printf("Numero de Host invalido\n");
                                else
                                    i = 0;
                            }
                            bitH = 1;
                            while(pow(2,bitH)<(noHost+2))
                                bitH++;
                            bitP = 8-bitH;
                            noHost = pow(2,8-bitP);
                            printf("Bits Prestado: %d\n", bitP);
                            printf("Bits de Host: %d\n", bitH);
                            noRedes = pow(2,bitP);
                            printf("No. de Subredes: %d\n", noRedes);
                            MRSR[3] = (255-noHost)+1;
                            imprime(IP,MRSR);
                        break;
                        case 3:
                            i=1;
                            while(i!=0){
                                printf("Mascara personalizada\n");
                                printf("%d.%d.%d.%d/", IP[0], IP[1], IP[2], IP[3]);
                                scanf("%d", &MaskP);
                                if(MaskP<24 || MaskP>30)
                                    printf("Mascara invalida\n" );
                                else
                                    i = 0;
                            }
                            printf("La red es %d.%d.%d.%d/%d \n", IP[0], IP[1], IP[2], IP[3], MaskP);
                            bitH = 32-MaskP;
                            bitP = 8-bitH;
                            bitP = 8-bitH;
                            printf("Bits Prestado: %d\n", bitP );
                            printf("Bits de Host: %d\n", bitH );
                            noRedes = pow(2,bitP);
                            printf("No. de Subredes: %d\n", noRedes );
                            noHost = pow(2,8-bitP);
                            MRSR[3] = (255-noHost)+1;
                            printf("No. de Host: %d\n", noHost-2 );
                            imprime(IP,MRSR);
                        break;
                    }
                break;
                case 4:
                    printf("\nIP de clase D\n");
                break;
                case 5:
                    printf("\nIP de clase E\n");
                break;
            }
        }
        else
            printf("\nDireccion IP invalida\n");
        printf("\n\nDesea repetir el programa? SI(1)   NO(2)\n");
        printf("Opcion: "); scanf("%d",&repeat);
        system("cls");
    }while(repeat == 1);
    return 0;
}

void imprime(unsigned char IP[], unsigned char MRSR[]){
    printf("La mascara de SubRed es %d.%d.%d.%d \n", MRSR[0], MRSR[1], MRSR[2], MRSR [3]);
    printf("La red es %d.%d.%d.%d/%d \n", IP[0], IP[1], IP[2], IP[3], 24+bitP);
    for(i=0;i<pow(2,bitP);i++){
        printf("%d | %d.%d.%d.%d   | ", i, IP[0], IP[1], IP[2], IP[3]+(noHost*i) );
        printf(" %d.%d.%d.%d    ", IP[0], IP[1], IP[2], (IP[3]+(noHost*i))+1 );
        printf("hasta %d.%d.%d.%d   | ", IP[0], IP[1], IP[2], (IP[3]+(noHost*i) + (noHost-1))-1 );
        printf(" %d.%d.%d.%d  | ", IP[0], IP[1], IP[2], IP[3]+(noHost*i) + (noHost-1));
        printf("\n");
    }
}
```

```
void imprimeB(unsigned char IP[]){
    ark=((pow(2,16-bitP))-1);
    MRSR[2] = ( (~ark)>>8) &255;
    MRSR[3] =  (~ark) & 255;
    printf("La mascara de SubRed es %d.%d.%d.%d \n", MRSR[0], MRSR[1], MRSR[2], MRSR [3]);
    printf("La red es %d.%d.%d.%d/%d \n", IP[0], IP[1], IP[2], IP[3], 16+bitP);
    ark=0;
    for(i=0;i<pow(2,bitP);i++){
        printf("%d | %d.%d.%d.%d | ", i, IP[0], IP[1], IP[2], IP[3]);
        printf(" %d.%d.%d.%d  ", IP[0], IP[1], IP[2], IP[3] +1 );
        ark= (noHost*i)+(noHost-1);
        IP[3]=ark & 255;
        IP[2]=(ark>>8) & 255;
        printf("hasta  %d.%d.%d.%d  | ", IP[0], IP[1], IP[2], (IP[3])-1 );
        printf(" %d.%d.%d.%d | ", IP[0], IP[1], IP[2] , IP[3] );
        printf("\n");
        ark= (noHost*i)+(noHost-1)+1;
        IP[3]=(ark & 255);
        IP[2]=(ark>>8) & 255;
    }
}

void imprimeA(unsigned char IP[]){
    ark=((pow(2,24-bitP))-1);
    MRSR[1] = ( (~ark)>>16) &255;
    MRSR[2] = ( (~ark)>>8) &255;
    MRSR[3] =  (~ark) & 255;
    printf("La mascara de SubRed es %d.%d.%d.%d \n", MRSR[0], MRSR[1], MRSR[2], MRSR [3]);
    printf("La red es %d.%d.%d.%d/%d \n", IP[0], IP[1], IP[2], IP[3], 8+bitP);
    ark=0;
    for(i=0;i<pow(2,bitP);i++){
        printf("%d | %d.%d.%d.%d | ", i, IP[0], IP[1], IP[2], IP[3]);
        printf(" %d.%d.%d.%d  ", IP[0], IP[1], IP[2], IP[3] +1 );
        ark= (noHost*i)+(noHost-1);
        IP[3]=ark & 255;
        IP[2]=(ark>>8) & 255;
        IP[1]=(ark>>16) & 255;
        printf("hasta  %d.%d.%d.%d  | ", IP[0], IP[1], IP[2], (IP[3])-1 );
        printf(" %d.%d.%d.%d | ", IP[0], IP[1], IP[2] , IP[3] );
        printf("\n");
        ark = (noHost*i)+(noHost-1)+1;
        IP[3]=(ark & 255);
        IP[2]=(ark>>8) & 255;
        IP[1]=(ark>>16) & 255;
    }
}

int 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
}

int validar_ip(unsigned int ip[]){
    int i = 0;

    for(i = 0; i < 4; i++){
        if(ip[i]<=255)
            IP[i] = ip[i];
        else
            return 0;
    }
    return 1;
}
```

Referencias

- Cisco Systems. (2018). Direccionamiento de IP y conexión en subredes para los usuarios nuevos. Septiembre 30, 2018, de Cisco Systems Sitio web: https://www.cisco.com/c/es_mx/support/docs/ip/routing-information-protocol-rip/13788-3.pdf
- Martínez, R. (2012). Subneteo. Septiembre 30, 2018, de S/N Sitio web: http://www.unico.com.ec/subneteo.pdf