Objetivo: el alumno programará una calculadora IP versión 1 en C. Haciendo uso de un block de notas y compilando en gcc

Dada una direccion IP (inicializada en hexadécimal)

Output: Clase, Dirección IP de Red y dirección de Broadcast.

Las IP de prueba son las siguientes, favor de llenar la tabla

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Dirección IP | Dir en hexadecimal | Clase | Dir. IP Red | Broadcast |
| **Z** | 10.250.1.1 | 0x0A.0xFA.0x01.0x01 | A | 10.0.0.0 | 10.255.255.255 |
| **Y** | 150.10.15.0 | 0X96 0X0A,0X0F 0X00 | B | 150.10.0.0 | 150.10.255.255 |
| **X** | 192.14.2.0 | 0XC0,0X0E,0X02,0X00 | C | 192.14.2.0 | 192.14.2.255 |
| **W** | 148.17.9.1 | 0X94 0X11 0X09 0X01 | B | 148.17.0.0 | 148.17.255.255 |
| **V** | 193.0.0.255 | 0XC1 0X00 0X00 0XFF | C | 193.0.0.0 | 193.255.255.255 |
| **U** | 220.200.23.1 | 0xDC,0xC8,0x17,0x01 | C | 220.200.23.0 | 220.200.23.255 |
| **T** | 177.100.18.4 | 0x4D,0x64,0x12,0x04 | B | 177.100.0.0 | 177.100.255.255 |
| **S** | 95.0.21.0 | 0x5E,0x00,0x15,0x00 | A | 95.0.0.0 | 95.255.255.255 |
| **R** | 111.111.111.111 | 0x6F,0x6F,0x6F,0x6F | A | 111.0.0.0 | 111.255.255.255 |
| **Q** | 219.26.51.0 | 0xDB,0x1A,0x33,0x00 | C | 219.26.51.0 | 219.26.51.255 |
| **P** | 167.0.0.0 | 0xA7,0x00,0x00,0x00 | B | 167.0.0.0 | 167.0.255.255 |
| **O** | 240.1.1.1 | 0xF0,0x01,0x01,0x01 | E | - | - |
| **N** | 10.1.1.1 | 0x0A,0x01,0x01,0x01 | A | 10.0.0.0 | 10.255.255.255 |
| **M** | 225.1.1.1 | 0xE1,0x01,0x01,0x01 | D | - | - |

Instrucciones.

-Abrir un block de notas y escribir el código correspondiente en C.

-La dirección IP se inicializará en hexadecimal en un arreglo de caracteres sin signo

- La mascará de red se inicializará en décimal en un arreglo de caracteres sin signo

-Compilar en consola haciendo uso de gcc (en caso de no tenerlo, instalarlo).

-Una vez terminado deberán probar con las 14 direcciones IP de la tabla. Tomar captura de pantalla de la salida para cada IP y pegarlas al final de este documento.

Se entrega este documento, incluir:

#include <stdio.h>

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

**{**

unsigned char ip**[**4**]** **=** **{**0xE1**,**0x01**,**0x01**,**0x01**};**

// {0x0A,0xFA,0x01,0x01}; //Z {10,250,1,1}

// {0x96,0x0A,0x0F,0x00}; //Y {150,10,15,0}

// {0xE1,0x01,0x01,0x01}; //M {225,1,1,1}

// {0xC0,0x0E,0x02,0x00}; // {192,14,2,0}

// {0x94,0x11,0x09,0x01}; //W {148,17,9,1}

// {0xC1,0x00,0x00,0xFF}; //V {193,0,0,255}

// {0xDC,0xC8,0x17,0x01}; //U {220,200,23,1}

// {0xB,0x64,0x12,0x04}; //T {177,100,18,4}

// {0x5F,0x00,0x15,0x00}; //S {95,0,21,0}

// {0x6F,0x6F,0x6F,0x6F}; //R {111,111,111,111}

// {0xDB,0x1A,0x33,0x00}; //Q {219,26,51,0}

// {0xA7,0x00,0x00,0x00}; //P {167,0,0,0}

// {0xF0,0x01,0x01,0x01}; //O {240,1,1,1}

// {0x0A,0x01,0x01,0x01}; //N {10,1,1,1}

unsigned char m**[**4**]={**0**,**0**,**0**,**0**};**

**if** **(**ip**[**0**]** **&** 128**)**

**{**

**if** **(**ip**[**0**]** **&** 64**)**

**{**

**if** **(**ip**[**0**]** **&** 32**)**

**{**

**if** **(**ip**[**0**]** **&** 16**)**

**{**

printf**(**"La IP ingresada es de tipo E\n"**);**

**}**

**else**

**{**

printf**(**"La IP ingresada es de tipo D \n"**);**

**}**

**}**

**else**

**{**

printf**(**"La IP ingresada es de tipo C\n"**);**

m**[**0**]** **=** 255**;**

m**[**1**]** **=** 255**;**

m**[**2**]** **=** 255**;**

printf**(**"La direccion de tipo red es: %u.%u.%u.%u \n"**,** **(**ip**[**0**]** **&** m**[**0**]),** **(**ip**[**1**]** **&** m**[**1**]),** **(**ip**[**2**]** **&** m**[**2**]),** **(**ip**[**3**]** **&** m**[**3**]));**

printf**(**"La direccion de tipo Broadcast es: %u.%u.%u.%u"**,** **(**unsigned char**)(**ip**[**0**]** **|** **(~**m**[**0**])),** **(**unsigned char**)(**ip**[**1**]** **|** **(~**m**[**1**])),** **(**unsigned char**)(**ip**[**2**]** **|** **(~**m**[**2**])),** **(**unsigned char**)(**ip**[**3**]** **|** **(~**m**[**3**])));**

**}**

**}**

**else**

**{**

printf**(**"La IP ingresada es de tipo A: \n"**);**

m**[**0**]** **=** 255**;**

printf**(**"La direccion de tipo red es: %d.%d.%d.%d \n"**,** **(**ip**[**0**]** **&** m**[**0**]),** **(**ip**[**1**]** **&** m**[**1**]),** **(**ip**[**2**]** **&** m**[**2**]),** **(**ip**[**3**]** **&** m**[**3**]));**

printf**(**"La direccion de tipo Broadcast es: %d.%d.%d.%d"**,** **(**unsigned char**)(**ip**[**0**]** **|** **(~**m**[**0**])),** **(**unsigned char**)(**ip**[**1**]** **|** **(~**m**[**1**])),** **(**unsigned char**)(**ip**[**2**]** **|** **(~**m**[**2**])),** **(**unsigned char**)(**ip**[**3**]** **|** **(~**m**[**3**])));**

**}**

**return** 0**;**

**}**

Deberan crear y mostrar **el mapa de memoria** utilizado en su programa (considerar registros de 8 bits)

*mapa de memoria*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Unsigned char IP[0] | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
| Unsigned char MR[0] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

Finalmenteincluir tus **Conclusiones**

\* sobre el uso de

If ()

Else como falso y verdadero

Me parece que saber implementarlo proporciona muchas ventajas al momento de programar en este lenguaje, pues nos ahorra memoria dentro de programas mas extensos, pues nos ahorramos tener que declarar como tal una variable que ocupara espacio, como muchas veces observamos que simplemente la variable verdadera le asignan el numero 1 (pues el if lo considera como verdadero), pero utilizando la lógica de los bits y sabiendo emplear estas técnicas nos ahorramos esa memoria, que probablemente en programas pequeños no vemos tan grave, pero a medida que el programa crece serán mayores los tiempos de ejecución así como de recursos consumidos, por lo que sería muy importante tener en cuenta que existen estas formas de poder aplicar los if-else

Comparar el costo computacional entre.

If(IP[0]&128)

Else

“clase A”

Contra

If (IP[0]>0 && IP[0]<127)

“clase A”

El costo computacional se ve reflejado en la cantidad de operaciones que deben ser ejecutadas para determinar el tipo de Red en cuestión, pues como primer ejemplo observamos que simplemente se hace uso de un operador binario que devolverá 1 en caso de que el numero en cuestión IP[0], también tenga encendido el bit mas significativo, en caso contrario no entrará a la condicional y continuará el programa o la siguiente comprobación. Por el toro lado, observamos que en el caso numero 2 se esta realizando la comparativa de:

1. El numero en cuestión es mayor a 0
2. El numero es menor a 127
3. Mas aparte la sentencia lógica de && la cual devuelve un tipo entero al momento de realizarse (lo cual estamos obteniendo también mediante el uso de la operación binaria)

Cabe destacar que en el último punto se menciono que && devuelve un valor de tipo int (4 bytes) mientras que el operador binario nos devuelve un tipo de dato de la misma procedencia de aquellos que fueron ingresados (en este caso unsigned char el cual solo ocupa 1 byte)

¿Habías usado los operadores binarios?

Tenía conocimiento acerca de ellos y los había utilizado en otras materias, aunque realmente nunca los había implementado en un programa directamente o los había usado para programar, me parece que es bastante efectivo y sabiéndolos utilizar puede reducir las líneas de código empleadas, así como costos en procesamiento, ejecución y por último optimiza la cantidad de memoria que podríamos estar consumiendo en comparación de otras implementaciones a las cuales estamos mas acostumbrados

¿Qué opinas del uso de operadores binarios?

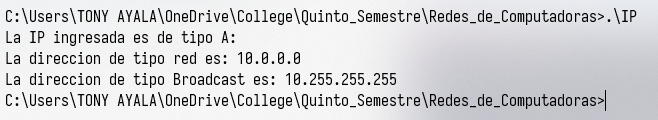
Me parece que debería ser fundamental su aprendizaje, pues como ya he mencionado con anterioridad ofrecen una gran ventaja y pueden proporcionar mayor eficacia con una implementación correcta, así como la disminución de costos y consumo de memoria, así como la optimización en velocidad.

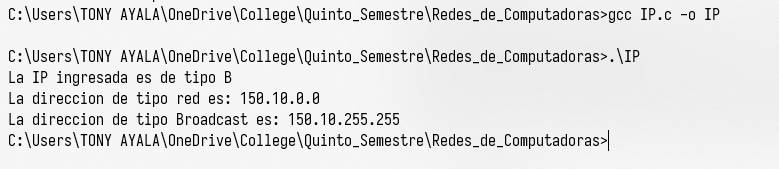
Considero que deberían haber mas implementaciones de los mismos con ejemplos dentro de los libros, pues no es un tema que sea muy retomado por los textos, a menos que estos sean un poco mas especializados, pero en los básicos casi no, cuando estos proporcionan una gran ventaja en cuestión de memoria

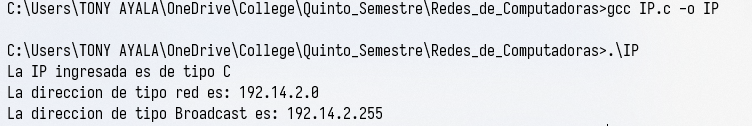
|  |  |  |
| --- | --- | --- |
| Criterio | Valor | Tu evaluación |
| El programa se escribe en un block de notas y se compila con gcc | **1** | 1 |
| Se incluye todo el código y se hace uso de operadores binarios | **1** | 1 |
| El mapa de memoria refleja todas las variables utilizadas en el programa y han sido seleccionadas de forma consciente. | **1** | 1 |
| Las imágenes son claras | **1** | 1 |
| Se incluyen las conclusiones solicitadas |  | 1 |
| TOTAL | **5** | 5 |

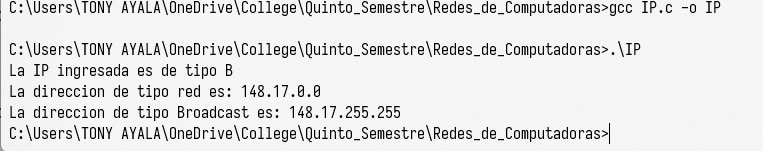
## Imágenes de la compilación del Código

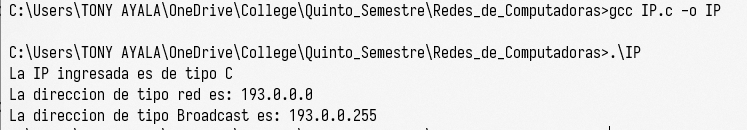
**Z 10.250.1.1**

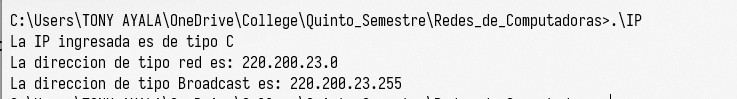


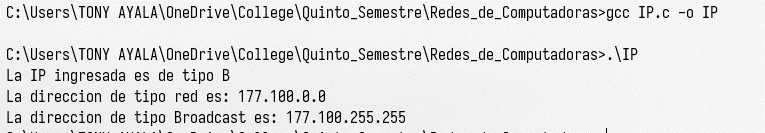
**Y 150.10.15.0**

**X 192.14.2.0**

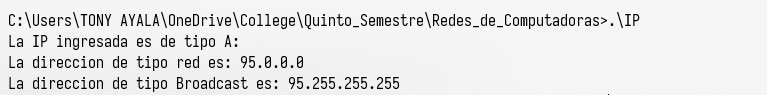
**W 148.17.9.1**

**V 193.0.0.255**

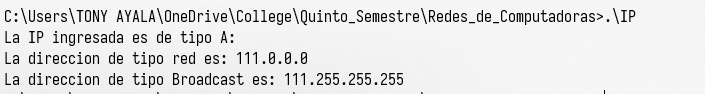
**U 220.200.23.1**

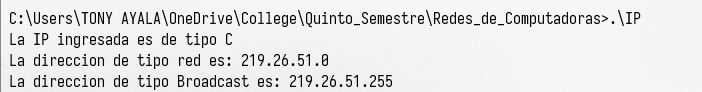
**T 177.100.18.4**

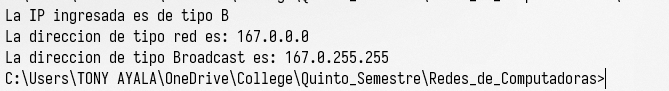
**S 95.0.21.0**

****

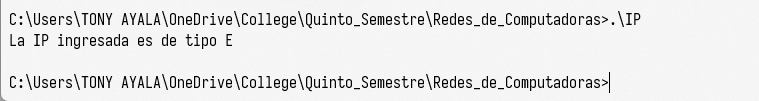
**R 111.111.111.111**

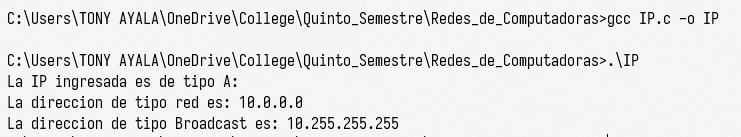
****

**Q 219.26.51.0**

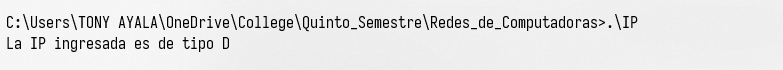
**P 167.0.0.0**

**O 240.1.1.1**

****

**N 10.1.1.1**

**M 225.1.1.1**

****