Sistemas de numeración y operaciones booleanas

En este documento vamos a centrarnos en los distintos sistemas de numeración que utilizaremos durante el curso, básicos para entender el funcionamiento de ciertos aspectos de la configuración y diseño de las redes. Haremos especial hincapié en el sistema de numeración binario, así como su conversión a otros sistemas, como son el octal, decimal y hexadecimal. Para ampliar algunos aspectos binarios, daremos una pasada a ciertas operaciones booleanas.

LOS SÍMBOLOS

Cada sistema de numeración tiene un número limitado de símbolos, que coincide con la base del sistema:

- Sistema binario, base 2, 2 símbolos: 0 y 1
- Sistema octal, base 8, 8 símbolos: 0, 1, 2, 3, 4, 5, 6 y 7
- Sistema decimal, base 10, 10 símbolos: 0, 1, 2, 3, 4, 5, 6, 7, 8 y 9
- Sistema hexadecimal, base 16, 16 símbolos: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Así, el número 1000 puede estar representado en cualquiera de los cuatro sistemas, mientras que el 100A está claramente representado en hexadecimal, ya que el símbolo A no pertenece a ninguno de los otros tres sistemas de numeración.

En el sistema hexadecimal, utilizamos las letras al quedarnos sin números. Hay que resaltar que las letras del sistema hexadecimal representan mediante un único símbolo dos símbolos decimales. Así, la A representa nuestro 10, la B al 11, la C al 12, la D al 13, la E al 14 y la F al 15.

El porqué se utiliza un sistema u otro depende del ámbito en el que nos encontremos. El sistema de numeración más habitual en el hombre es el decimal, ya que comenzamos a contar con los diez dedos que teníamos más cerca: nuestras manos. Sin embargo, para los sistemas informáticos utilizamos el binario porque son dos los estados que maneja la máquina. En un ordenador, los datos se almacenan en binario (el láser del lector del CD refleja o no refleja la luz, el sensor de la disquetera detecta la polarización positiva o negativa del disquete), internamente las puertas dejan o no dejan pasar la corriente (dos estados), los procesadores leen instrucciones en binario...

El octal y el hexadecimal se utilizan para facilitar el manejo de números muy grandes. Efectivamente, a mayor número de símbolos, menos posiciones se requieren en su representación. Así, el número binario $10000000_{(2}$ se representa $200_{(8}$ en octal y $80_{(16}$ en hexadecimal. Además, estos tres sistemas tienen una base múltiplo de dos:

- Binario, con base $2 = 2^1$
- Octal, con base $8 = 2^3$, y
- Hexadecimal, con base $16 = 2^4$

Esta característica facilita la conversión entre estos sistemas, como veremos posteriormente

LOS PESOS

Los sistemas de numeración no son más que unas normas de juego para representar ciertos valores de distintas formas. Los cuatro sistemas de numeración que vamos a revisar (el binario, octal, decimal y hexadecimal) tienen en común que funcionan por pesos. Cada símbolo del sistema representa un valor o peso en función de la posición que ocupe en el número. Así, en el sistema decimal, el símbolo 2 toma un valor distinto si está en la posición de las unidades (2 euros) o si está en la posición de las centenas (200 euros). A medida que el símbolo se posiciona más a la izquierda, más valor o peso adquiere. Así, los pesos en el sistema decimal son los siguientes:

- En la posición 0 (de las unidades, la más a la derecha) el símbolo toma el peso 10^0=1
- En la posición 1 (de las decenas) el símbolo toma el peso 10^1=10
- En la posición 2 (de las centenas) el símbolo toma el peso 10^2=100

Cada sistema de numeración tiene una tabla de pesos, que en el caso del sistema decimal es el siguiente:

Sistema decimal (10)							
6	5	4	3	2	1	0	Posición
10^6	10^5	10^4	10^3	10^2	10^1	10^0	Potencias
1000000	100000	10000	1000	100	10	1	Pesos

En el sistema binario, los pesos se calculan de igual forma: base del sistema elevado a la posición, o sea:

Sistema binario (2)									
8	7	6	5	4	3	2	1	0	Posición
2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	Potencias
256	128	64	32	16	8	4	2	1	Pesos

Los pesos del sistema octal son:

Sistema octal (8)					
4	3	2	1	0	Posición
8^4	8^3	8^2	8^1	8^0	Potencias
4096	512	64	8	1	Pesos

Y los del hexadecimal:

Sistema hexadecimal (16)					
4	3	2	1	0	Posición
16^4	16^3	16^2	16^1	16^0	Potencias
65536	4096	256	16	1	Pesos

Así, para el símbolo anterior (el 2), su valor cambia según la posición que tome. En el sistema decimal:

Sistema decimal (10)							
1000000	100000	10000	1000	100	10	1	Pesos
0	0	0	0	0	0	2	2
0	0	0	0	0	2	0	20
0	0	0	0	2	0	0	200

- 2, en la posición 0, con el peso 1, $2*1=2_{(10}$
- 20, en la posición 1, con el peso 10, $2*10=20_{(10}$
- 200, en la posición 2, con el peso 100, $2*100=200_{(10}$

Si hablásemos del sistema octal, y cambiásemos de símbolo, por ejemplo, el 3:

Sistema octal (8)					
4096	512	64	8	1	Pesos
0	0	0	0	3	3
0	0	0	3	0	24
0	0	3	0	0	192

- $3_{(8}$, en la posición 0, con el peso 1, $3*1=3_{(10}$
- $30_{(8}$, en la posición 1, con el peso 8, $3*8=24_{(10}$
- $300_{(8}$, en la posición 2, con el peso 64, $3*64=192_{(10}$

Si hablásemos del sistema hexadecimal:

Sistema hexadecimal (16)					
65536	4096	256	16	1	Pesos
0	0	0	0	3	3
0	0	0	3	0	48
0	0	3	0	0	768

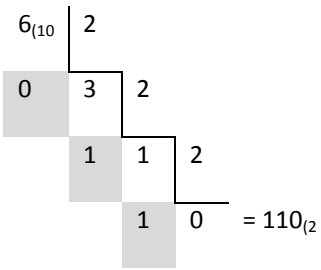
- $3_{(16}$, en la posición 0, con el peso 1, $3*1=3_{(10}$
- $30_{(16}$, en la posición 1, con el peso 16, $3*16=48_{(10}$
- $300_{(16}$, en la posición 2, con el peso 256, $3*256=768_{(10}$

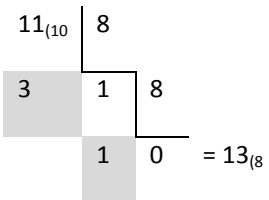
CONVERSIÓN. DEL DECIMAL AL RESTO... POR RESTOS.

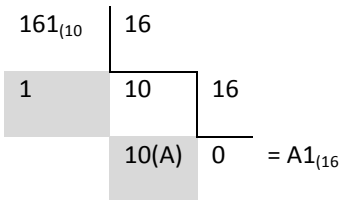
Supongamos que queremos pasar de la representación decimal a cualquiera de los sistemas. Podríamos hacerlo por la "cuenta de la vieja": dividiendo consecutivamente (hasta que no se admitan más divisiones enteras) por la base del sistema al que queremos pasar el valor. De esta forma, los restos serán siempre menores que el divisor (la base) obteniendo así los símbolos de los distintos sistemas:

- Para pasar del decimal al binario, dividimos consecutivamente por 2, obteniendo como restos 0 y 1.
- Para pasar del decimal al octal, dividimos consecutivamente por 8, obteniendo como restos símbolos menores que 8 (0..7)
- Para pasar del decimal al hexadecimal, dividimos consecutivamente por 16, obteniendo como restos números menores que 16 (0..15) Hacer hincapié en que cada resto ha de ser un único símbolo, y así hay que sustituirlo si en el resto nos sale un valor superior a 9 (10=A, 11=B..., 15=F)

Una vez obtenidos todos los restos (símbolos del nuevo sistema de numeración), han de colocarse en el orden correcto, es decir, se toman todos los restos, de abajo a arriba, y se ordenan de izquierda a derecha. De esta forma, el último resto se coloca como el símbolo de mayor peso, y el primer resto tomará la posición más a la derecha, ocupando el menor peso. A continuación se muestran 3 ejemplos para pasar de decimal a otros sistemas mediante divisiones consecutivas y tomando restos…







CONVERSIÓN DEL DECIMAL AL BINARIO: POR PESOS

Como vimos anteriormente, cada posición en el sistema representa un peso. Así, en binario, los pesos de un byte (8 bits) son los siguientes:

128	64	32	16	8	4	2	1

Para representar un valor decimal en binario hay que desglosarlo en estos pesos. Al final, los pesos elegidos han de sumar el número decimal. Para hacerlo, ha de asignarle el mayor peso que "quepa", y a continuación repetir la operación con lo que quede hasta completar la suma.

Veamos algunos ejemplos:

- $128_{(10} = 128 = 10000000_{(2}$
- $129_{(10} = 128 + 1 = 10000001_{(2}$ (al ser 129 un número impar, acaba en 1)
- $130_{(10} = 128 + 2 = 10000010_{(2}$ (al ser 130 un número par, acaba en 0)
- $146_{(10} = 128 + 16 + 2 = 10010010_{(2}$
- $127_{(10} = 64 + 32 + 16 + 8 + 4 + 2 + 1 = 1111111_{(2}$ (es el 128 – 1, el anterior a 128, por lo que se representa con todos los pesos anteriores a 1)
- $192_{(10} = 128 + 64 = 11000000_{(2}$
- $191_{(10} = 128 + 32 + 16 + 8 + 4 + 2 + 1 = 10111111_{(2}$ (el anterior a 192)

CONVERSIÓN ENTRE BINARIO, OCTAL Y HEXADECIMAL: POR GRUPOS

Estos tres sistemas tienen en común que su base (2^0, 2^3, 2^4) es múltiplo de dos. A continuación se muestra la tabla de correspondencia entre los tres sistemas:

Binario	Octal	Hexadecimal
0	0	0
1	1	1
10	2	2
11	3	3
100	4	4
101	5	5
110	6	6
111	7	7
1000	10	8

Binario	Octal	Hexadecimal
1001	11	9
1010	12	A
1011	13	B
1100	14	C
1101	15	D
1110	16	E
1111	17	F

El último símbolo de cada sistema (el 7 y la F) se representa con todos los bits a 1, es decir, con tres o cuatro posiciones al máximo. Es esta característica la que nos facilita la conversión entre los sistemas de base dos: Los bits se agrupan de derecha a izquierda en grupos de 3 ó de 4 y se sustituyen por su correspondiente símbolo en el sistema destino. Veamos unos ejemplos:

$111101001000_{(2} = 111.101.001.000 = 7510_{(8}$

$111101001000_{(2} = 1111.0100.1000 = F48_{(16}$

En caso de querer pasar del hexadecimal u octal al binario, hay que sustituir cada símbolo por su correspondiente valor en el sistema binario, pero siempre en grupo de tres (desde el octal) o de cuatro (desde el hexadecimal). Veamos unos ejemplos:

$10626_{(8} = 1.000.110.010.110 = 1000110010110_{(2}$

$A0F1_{(16} = 1010.0000.1111.0001 = 1010000011110001_{(2}$

PRACTICANDO

Vamos a realizar unas prácticas de conversión

- $367_{(10} = 256 + 64 +32 + 8 + 4 + 2 + 1 = 101101111_{(2}$
- $101101111_{(2} = 1.0110.1111 = 16F_{(16}$
- $101101111_{(2} = 101.101.111 = 557_{(8}$
- $367_{(8} = 011.110.111 = 011110111_{(2}$
- $11110111_{(2} = 1111.0111_{(2} = F7_{(16}$
- $11110111_{(2} = 255 - 8 = 247_{(10}$
- $367_{(16} = 0011.0110.0111 = 001101100111_{(2}$
- $1101100111_{(2} = 1.101.100.111 = 1547_{(8}$
- $1101100111_{(2} = 1023 - (8 + 16 + 128) = 1023-152 = 871_{(10}$

OPERACIONES BOOLEANAS

El Álgebra de Bool es una parte de las Matemáticas dedicada a las operaciones lógicas. Nos centraremos sólo en aquellas operaciones que nos serán necesarias para el desarrollo de las direcciones IP, y son las siguientes:

- Operación AND ó Y lógico,
- Operación OR u O lógico,
- Operación XOR u O exclusivo,
- Operación NOT ó negación lógica

X	Y	AND(X, Y)	OR(X, Y)	XOR(X, Y)	NOT(X)
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

En resumen, estos son los resultados:

- Para conseguir el resultado 1 (verdad) con el operador AND, todos los operadores han de ser 1 (verdad). Han de ser verdad uno Y otro. En cualquier otro caso, el resultado es 0 (falso).
- Para conseguir el resultado 1 (verdad) con el operador OR, ha de aparecer un 1 (verdad) en alguno de los operadores. Han de ser verdad uno u(O) otro. Sólo si todos los operadores son 0 (falso) el resultado es 0 (falso).
- Para conseguir el resultado 1 (verdad) con el operador XOR, ha de aparecer sólo un 1 (verdad) en alguno de los operadores. Han de ser verdad uno u(O) otro de forma EXCLUSIVA. En cualquier otro caso, el resultado es 0 (falso).
- El operador NOT niega (cambia) el valor del operando.

Nótense ciertos aspectos de estos operadores:

- La operación AND transforma un bit a 0 aplicando la operación entre el bit y el 0, y mantiene el valor de un bit si la operación se realiza entre el bit y el 1. Esta operación se usa para pasarle la máscara (número binario de 32 bits con la parte izquierda a 1 y la parte derecha a 0) a una dirección IP (número binario de 32 bits)
- La operación OR transforma un bit a 1 aplicando la operación entre el bit y el 1, y mantiene el valor de un bit si la operación se realiza entre el bit y el 0.
- El resultado de la operación XOR es 0 si ambos operandos coinciden. Se utiliza para comparar dos valores IP.

Veamos algunos ejemplos:

	0	1	0	1	0	1	0	1	0	1	0
AND	1	1	1	1	1	0	0	0	0	0	0
	0	1	0	1	0	0	0	0	0	0	0
	0	1	0	1	0	1	0	1	0	1	0
OR	1	1	1	1	1	0	0	0	0	0	0
	1	1	1	1	1	1	0	1	0	1	0
	0	1	0	1	0	1	0	1	0	1	0
XOR	0	1	0	1	0	0	0	0	1	1	1
	0	0	0	0	0	1	0	1	1	0	1

TAREAS

Ahora practica tú. Traduce al resto de sistemas de numeración sin utilizar la calculadora. El día del examen no dispondrás de ella:

- $193_{(10}=$, $128_{(10}=$, $127_{(10}=$, $169_{(10}=$, $255_{(10}=$, $254_{(10}=$, $172_{(10}=$, $1030_{(10}=$, $990_{(10}=$, $873_{(10}=$,
- $772_{(8}=$, $654_{(8}=$, $637_{(8}=$, $100_{(8}=$
- $A3D_{(16}=$, $7B0_{(16}=$, $ABC_{(16}=$, $11F_{(16}=$
- $11100001_{(2}=$, $1010111011_{(2}=$, $11110111_{(2}=$, $11001000_{(2}=$
- AND($195_{(10}$, $240_{(10)}$)=, AND($174_{(10}$, $224_{(10)}$)=, AND($168_{(10}$, $248_{(10)}$)=, AND($120_{(10}$, $128_{(10)}$)=
- OR($196_{(10}$, $241_{(10)}$)=, OR($172_{(10}$, $220_{(10)}$)=, OR ($160_{(10}$, $241_{(10)}$)=, OR ($126_{(10}$, $126_{(10)}$)=
- XOR($196_{(10}$, $241_{(10)}$)=, XOR($172_{(10}$, $220_{(10)}$)=, XOR ($160_{(10}$, $241_{(10)}$)=, XOR ($126_{(10}$, $126_{(10)}$)=

Pero además, responde a las siguientes preguntas:

- En el sistema binario, ¿cuántas combinaciones son posibles utilizando 7 cifras? ¿Y con 8 cifras?
- En el sistema hexadecimal, ¿cuántas combinaciones son posibles utilizando 2 cifras? ¿Y con 3 cifras?