

IIC2333 - P1

With , by KnowYourselfs

Disco

Es un archivo como
cualquier otro.

Disco



- MBT contiene información de particiones.
- El resto contiene todas las particiones.

Master Boot Table

Master Boot Table



- 128 entradas ($128 \times 8\text{B} = 1024\text{B} = 1\text{KB}$).
- Cada entrada representa una partición.
- 0 particiones mínimo, 128 máximo.

Master Boot Table - Entrada

1b V 0/1	7b ID 0-127	3B Identificador Absoluto	4B Cantidad de Bloques
----------------	-------------------	---------------------------------	------------------------------

- Lo más importante es el bit de validez.
- ID absoluto + Cant. de Bloques es suficiente información para manejar la partición.

Master Boot Table - Crear partición

Queremos crear una partición de
ID 120 y tamaño X KB.

¿Cómo lo hacemos?



Master Boot Table - Crear partición

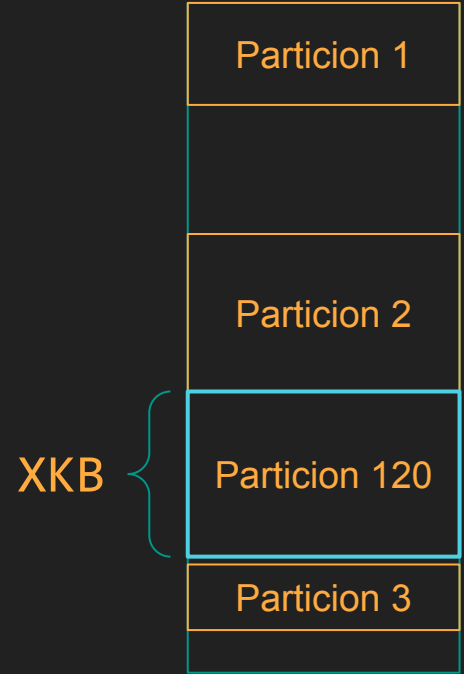
Agarramos el primer espacio disponible.

¿Qué pasaría si fuera más grande?



Master Boot Table - Crear partición

¿Ahora como eliminamos la partición 2?



Master Boot Table - Eliminar partición

Simplemente cambiamos el bit validez a 0.

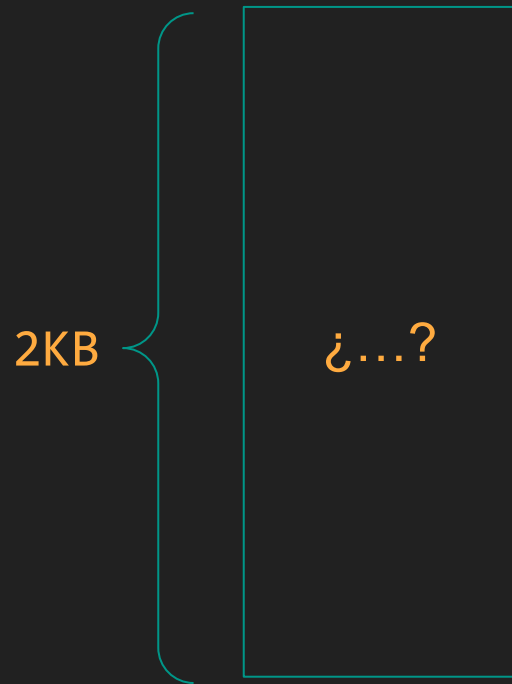


Bloques

Bloques

El disco esta separado en bloques de 2KB
cada uno.

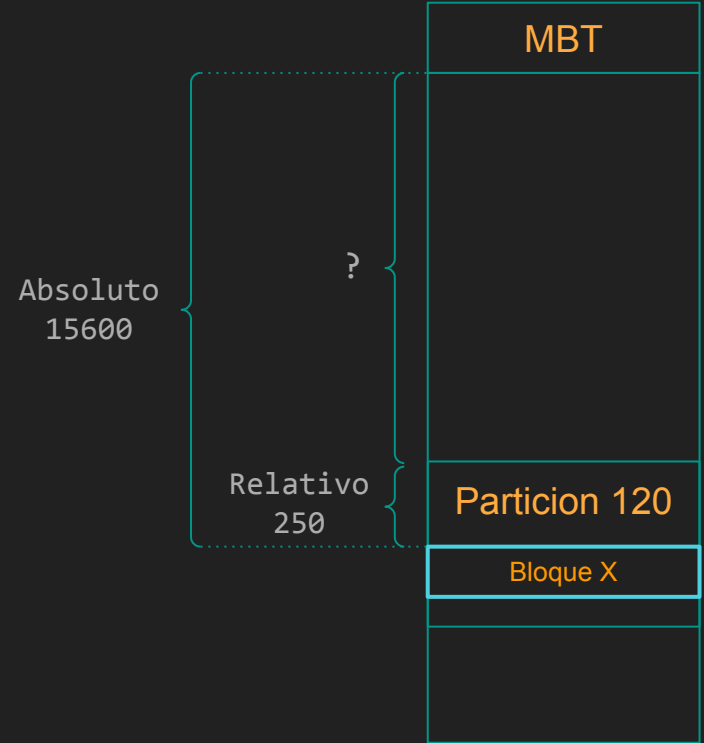
La estructura interna de estos 2KB depende
del tipo de bloque.



Bloques - Identificador

Identificador absoluto al
disco entero.

Identificador relativo a la
partición.



Bloques - Entrada de Directorio

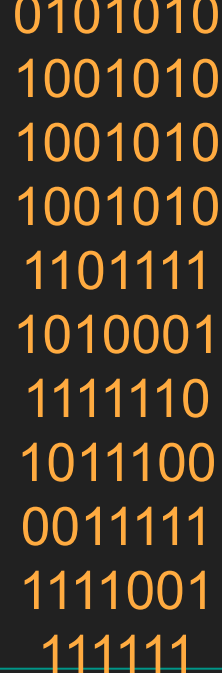
- Byte validez.
- Identificador relativo de bloque índice.
- Nombre.



Bloques - Bitmap

- Cada bit del bitmap representa un bloque.
- Al crear una partición se debe crear la mínima cantidad de bitmaps posible.

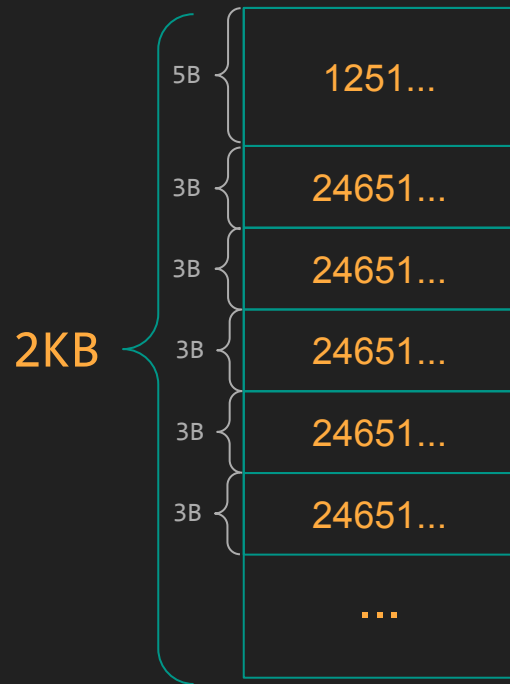
2KB



```
0101010
1001010
1001010
1001010
1101111
1010001
1111110
1011100
0011111
1111001
1111111
```

Bloques - Índice

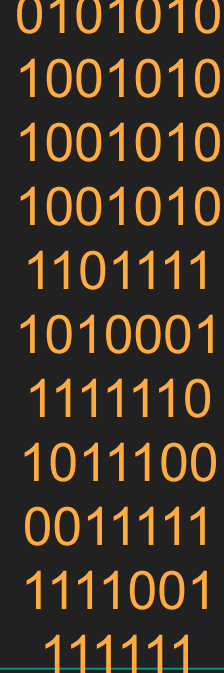
- Tamaño del archivo
- Punteros a bloques de datos



Bloques - Datos

- Son el contenido final del archivo.
- Si tomamos todos los bloques de datos y los pegamos, obtenemos el archivo original.

2KB



```
0101010
1001010
1001010
1001010
1101111
1010001
1111110
1011100
0011111
1111001
1111111
```

¿Cómo mostramos todo esto?

¡Con la API!

¿Dudas?

Manejo de Bytes en C

Manejo de Bytes

- `fopen` en modo `r+b`.
- `fseek` con constantes `SEEK_SET`, `SEEK_END`, `SEEK_CUR`.
- `fwrite` y `fread` para leer y escribir bytes.
- Manejo numérico de bits con shifts.

Manejo de Strings en C

Manejo de Strings

- Recordar que los strings son arrays de caracteres.
- `strcpy` para copiar strings.
- `strcmp` para comparar strings.

Bonus

¿En que consiste?

¿Que necesito saber?

Bonus - enum



```
typedef enum day {  
    MONDAY,  
    TUESDAY,  
    WEDNESDAY,  
    THURSDAY,  
    FRIDAY  
} Day;  
  
Day publication_date = MONDAY;
```

`enum` nos permite crear un nuevo `tipo` enumerado.

Asigna números a palabras y nos permite usarlas.

Mejora legibilidad del código.

Bonus - switch



```
int tipo = 0;

switch (tipo)
{
    case 0:
        // código
        break;
    case 1:
        // código
        break;
    default:
        // código
        break;
}
```

`switch` nos permite comparar una `variable` con un valor `constante` de forma directa y sencilla.

Bonus - switch + enum



```
Day day = MONDAY;

switch (day)
{
    case MONDAY:
        printf("Die.\n")
        break;
    case FRIDAY:
        printf("Celebrate!\n")
        break;
    default:
        printf("Meh.\n")
        break;
}
```

Podemos mezclar ambos para lograr manejo de casos de forma legible.

Bonus - extern



```
extern int errno
```

Explicar esto es una larga historia.

Se debe utilizar en header files para variables que van a estar disponibles en muchos archivos.

Algunas Referencias:

- [Binary Operators](#)
- [Cambiar un bit de un byte](#)
- [¿Que es extern?](#)