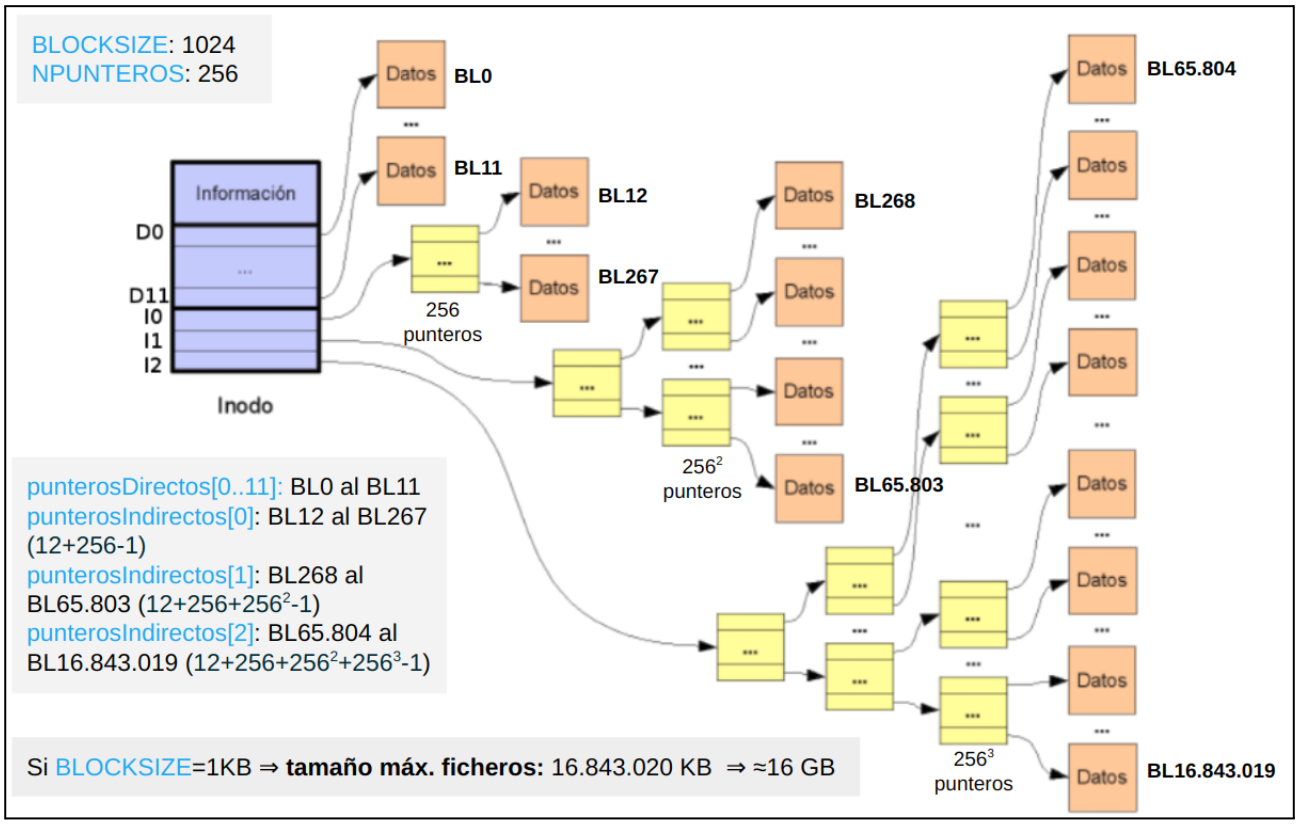
Nivel 4: ficheros_basico.c {traducir_bloque_inodo() y auxiliares: obtener_indice() y obtener_nRangoBL()}

Antes de nada, repasemos cómo los inodos apuntan a los bloques de datos:



En concreto, tenemos inodos con 12 **punteros directos** a bloques físicos y 3 **punteros indirectos**, de diferentes niveles, que apuntan a bloques de índices, y suponemos que el tamaño de bloque es de 1.024 bytes (por lo que en un bloque caben 256 punteros ya que cada puntero se representa con 4 bytes correspondientes al *sizeof(unsigned int)*).

Entonces:

- Los **punteros directos** a bloques físicos de datos, *punterosDirectos[0]* .. *punterosDirectos[11]* [1], permiten encontrar donde se hallan físicamente los primeros 12 bloques lógicos del inodo, es decir, los comprendidos entre el 0 y el 0+12-1: o sea del 0 al 11.
- El **puntero de bloques indirectos 0**, *punterosIndirectos[0]* [2], permite encontrar donde se hallan físicamente los siguientes 256 bloques lógicos del inodo, es decir, los comprendidos entre el 0+12 y el 0+12+256-1: o sea del 12 al 267.

[1] D0..D11 para abreviar
[2] I0 para abreviar

- El **puntero de bloques indirectos 1**, *punterosIndirectos[1]* [3], permite encontrar donde se hallan físicamente los siguientes 256^2 (65.536) bloques lógicos del inodo, es decir, los comprendidos entre el 0+12+256 y el 0+12+256+256^2-1: o sea del 268 al 65.803.
- El **puntero de bloques indirectos 2**, *punterosIndirectos[2]* [4], permite encontrar donde se hallan físicamente los siguientes 256^3 (16.777.216) bloques lógicos del inodo, es decir, los comprendidos entre el 0+12+256+256^2 y el 0+12+256+256^2+256^3-1: o sea del 65.804 al 16.843.019.

A partir de los datos anteriores podemos crear una función auxiliar *obtener_nRangoBL()* que asocie un **rango de bloques lógicos** siendo:

- nRangoBL=0 para bloques lógicos [0 , 11],
- nRangoBL=1 para bloques lógicos [12 , 267],
- nRangoBL=2 para bloques lógicos [268 , 65.803] y
- nRangoBL=3 para bloques lógicos [65.804 , 16.843.019].

También ha de actualizar una variable puntero, *ptr*, pasada por referencia, para que apunte donde lo hace el puntero correspondiente del inodo.

Primeramente podemos definir constantes simbólicas (en **ficheros_basico.h**) que nos ayuden a determinar los rangos de punteros[5]:

```
#define NPUNTEROS (BLOCKSIZE / sizeof(unsigned int))   // 256
#define DIRECTOS 12
#define INDIRECTOS0 (NPUNTEROS + DIRECTOS)    // 268
#define INDIRECTOS1 (NPUNTEROS * NPUNTEROS + INDIRECTOS0)    // 65.804
#define INDIRECTOS2 (NPUNTEROS * NPUNTEROS * NPUNTEROS + INDIRECTOS1) // 16.843.020
```

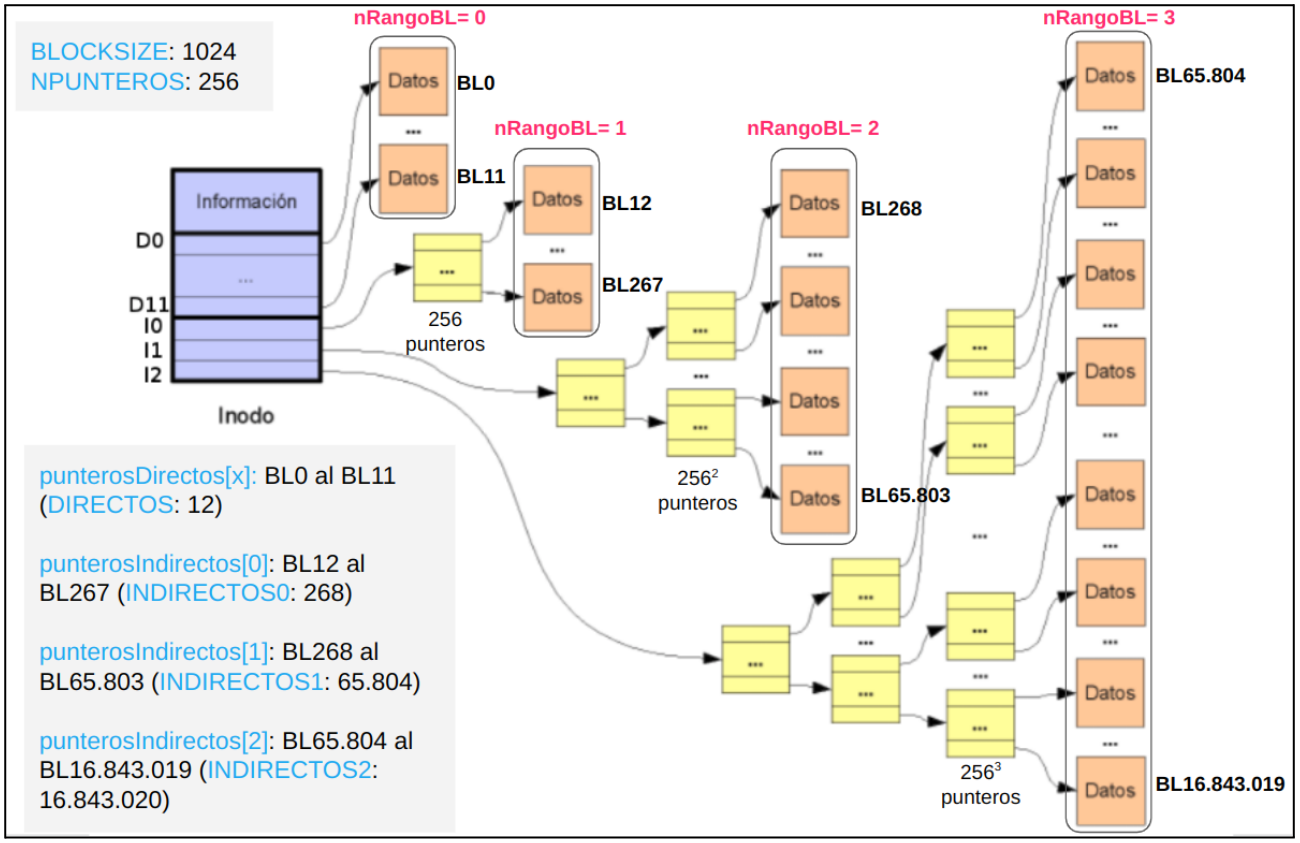
Y luego (en **ficheros_basico.c**) definimos la función *obtener_nRangoBL()* para obtener el **rango de punteros** en el que se sitúa el bloque lógico que buscamos (0:D, 1:I0, 2:I1, 3:I2), y **obtenemos además la dirección almacenada en el puntero correspondiente del inodo**, *ptr*:

```
funcion obtener_nRangoBL (*inodo: struct inodo, nblogico:unsigned ent, *ptr:unsigned ent)
devolver nrangoBL:ent

   si nblogico<DIRECTOS entonces   // <12
      *ptr:=inodo->punterosDirectos[nblogico]
      devolver 0
   si_no si nblogico<INDIRECTOS0 entonces   // <268
      *ptr:=inodo->punterosIndirectos[0]
      devolver 1
```

[3] I1 para abreviar
[4] I2 para abreviar
[5] Muy importante el uso de los paréntesis en estas definiciones!!!

```
  si_no si nblogico<INDIRECTOS1 entonces   // <65.804
     *ptr:=inodo->punterosIndirectos[1]
     devolver 2
  si_no si nblogico<INDIRECTOS2 entonces   // <16.843.020
     *ptr:=inodo->punterosIndirectos[2]
     devolver 3
  si_no
     *ptr:=0
     error("Bloque lógico fuera de rango")
     devolver -1
  fsi
ffuncion
```



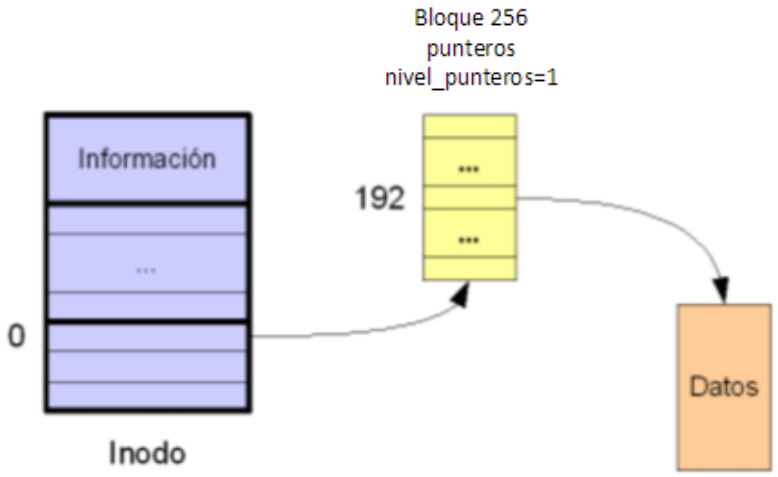
Veamos ahora ejemplos de cómo se obtienen los **índices de los bloques de punteros**:

Si, por ejemplo, queremos traducir el **bloque lógico número 204**:

> Como 12 <= 204 < 268 (rango=1), hemos de recurrir al *punterosIndirectos[0]*.
>
> Restamos los 12 punteros anteriores para obtener un valor índice dentro del rango de los 256 punteros, [0,255], del bloque de índices de nivel 1 que cuelga de *punterosIndirectos[0]*:

204 - 12 = **192** (⊂[0,255])

Por tanto el número de nuestro bloque físico se encuentra en el puntero número **192** del bloque de índices de nivel 1 apuntado por *punterosIndirectos[0]*.



Si, por ejemplo, queremos traducir el **bloque lógico número 30.004**:

Como 268 <= 30.004 < 65.804 (rango=2), hemos de recurrir a *punterosIndirectos[1]*.

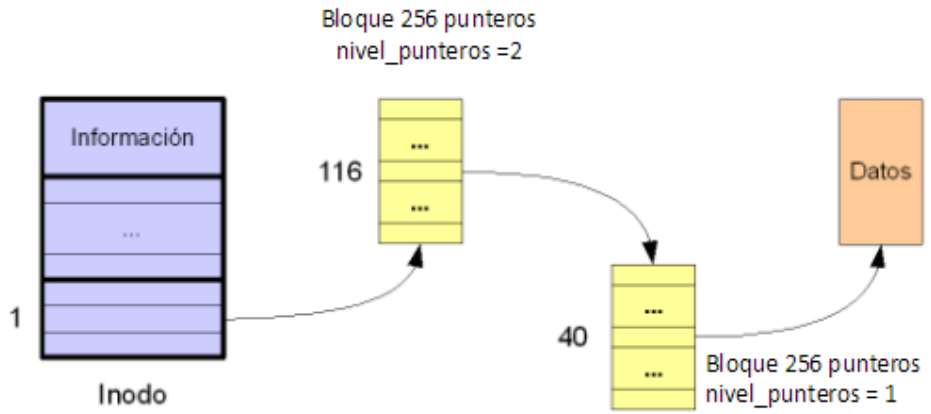
Restamos los 268 punteros anteriores (12+256) para obtener un valor dentro del rango de los 256² (65.536) bloques que puedo direccionar con la estructura global de punteros que cuelga de *punterosIndirectos[1]*:

30.004 - 268 = **29.736** (⊂[0, 65.535])

Dado que ese valor puede estar en uno de los 256 bloques de punteros de nivel 1, que cuelgan del bloque de punteros de nivel 2, tendremos que dividir ese valor entre 256 para obtener de cuál se trata (índice en el bloque de punteros de nivel 2). El resto de la división (módulo) nos dará el índice correspondiente en ese bloque de punteros de nivel 1, apuntado por el que hemos obtenido en el nivel 2:

29.736 / 256 = **116**
29.736 % 256 = **40**

Por tanto, el número de nuestro bloque físico se encuentra en el puntero número **40** del bloque apuntado por el puntero número **116** del bloque apuntado por *punterosIndirectos[1]*.



Si, por ejemplo, queremos traducir el **bloque lógico número 400.004**:

Como 65.804 <= 400.004 < 16.843.020 (rango=3), hemos de recurrir a *punterosIndirectos[2]*.

Restamos los 65.804 punteros anteriores (12+256+65.536) para obtener un valor dentro del rango de los 256^3 (16.777.216) bloques que puedo direccionar con la estructura global de punteros que cuelga de *punterosIndirectos[2]*:

$$400.004 - 65.804 = 334.200 \ (\subset [0, 16.777.215])$$

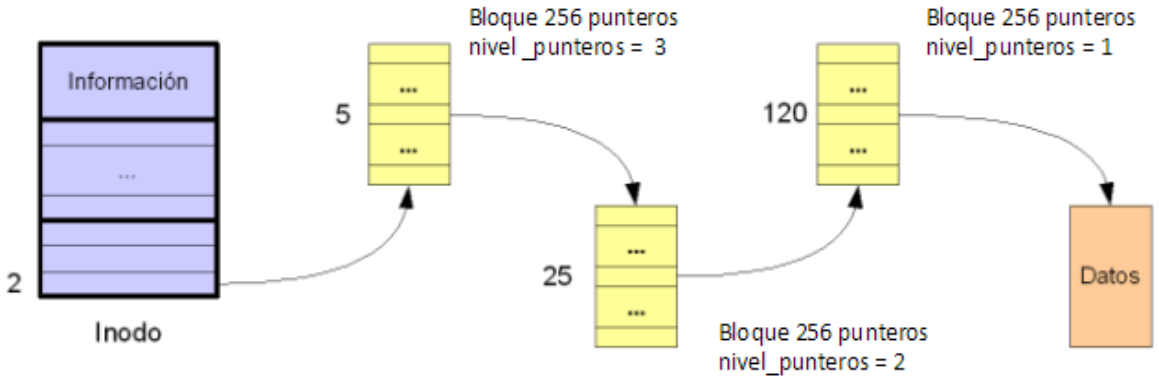
Para saber ese bloque a qué índice de nivel 3 corresponde, tendremos que dividirlo entre los 256^2 que salen de su estructura:

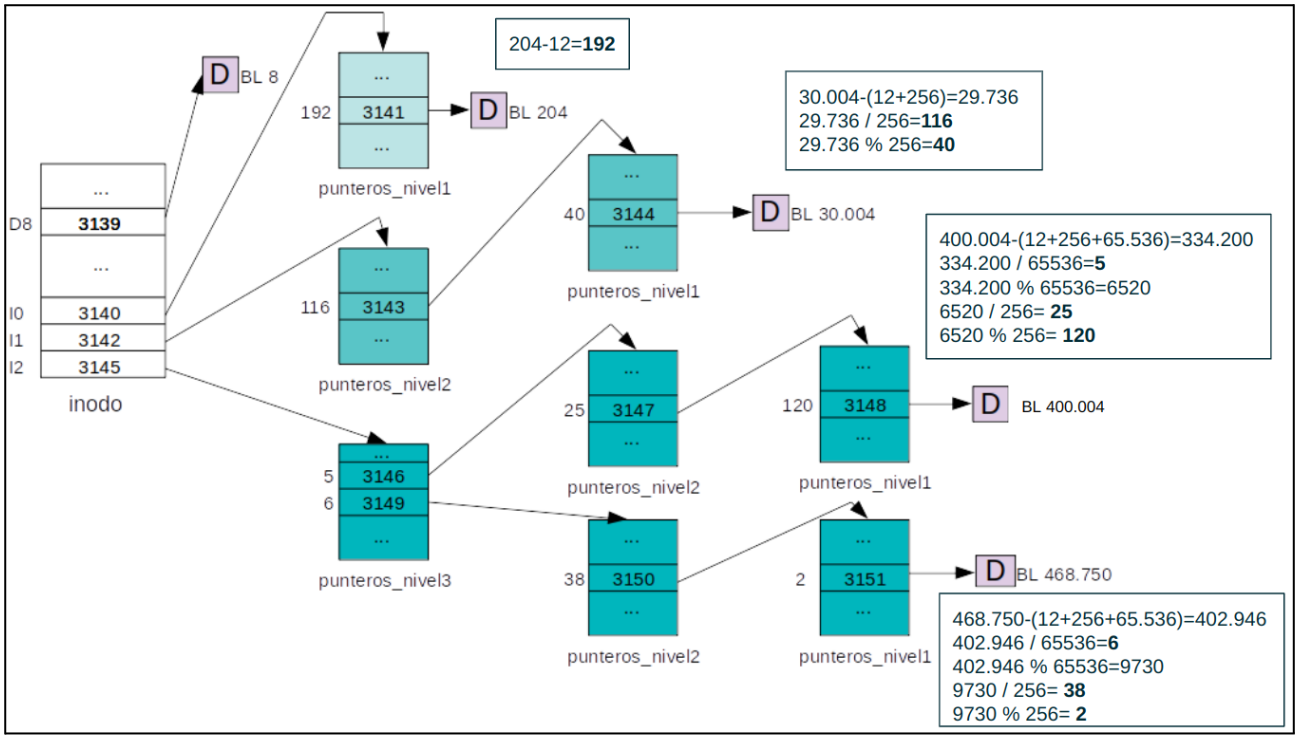
$$334.200 / 65.536 = 5$$

El resto de esa división habrá que dividirlo entre los 256 bloques de punteros que cuelgan para determinar el índice del bloque de punteros de nivel 2, y el módulo nos dará el índice del nivel 1:

$$334.200 \% 65.536 = 6.520$$
$$6.520 / 256 = 25$$
$$6.520 \% 256 = 120$$

Por tanto, el número de nuestro bloque físico se encuentra en el puntero número **120** del bloque apuntado por el puntero número **25** del bloque apuntado por el puntero número **5** del bloque apuntado por indirectos 2.





De los ejemplos anteriores podemos deducir una función, *obtener_indice()* para generalizar la obtención de los índices de los bloques de punteros:

```
funcion obtener_indice (nblogico: unsigned ent, nivel_punteros:ent) devolver ind:ent
si nblogico < DIRECTOS entonces devolver nblogico   //ej. nblogico=8
si_no si nblogico < INDIRECTOS0 entonces devolver nblogico - DIRECTOS   //ej. nblogico=204
si_no si nblogico < INDIRECTOS1 entonces   //ej. nblogico=30.004
 si nivel_punteros = 2 entonces
   devolver (nblogico - INDIRECTOS0) / NPUNTEROS
 si_no si nivel_punteros=1 entonces
   devolver (nblogico - INDIRECTOS0) % NPUNTEROS
 fsi
```

```
si_no si nblogico < INDIRECTOS2 entonces   //ej. nblogico=400.004
 si nivel_punteros = 3 entonces
   devolver (nblogico - INDIRECTOS1) / (NPUNTEROS * NPUNTEROS)
 si_no si nivel_punteros = 2 entonces
   devolver ((nblogico - INDIRECTOS1) % (NPUNTEROS * NPUNTEROS)) / NPUNTEROS
 si_no si nivel_punteros = 1 entonces
   devolver ((nblogico - INDIRECTOS1) % (NPUNTEROS * NPUNTEROS)) % NPUNTEROS
 fsi
fsi
ffuncion
```

Ahora ya podemos desarrollar la función de *traducir_bloque_inodo()*:

1) int traducir_bloque_inodo(struct inodo **ninodo*, unsigned int *nblogico*, unsigned char *reservar*);

Esta función se encarga de obtener el **nº de bloque físico** correspondiente a un **bloque lógico** determinado del inodo indicado. Enmascara la gestión de los diferentes rangos de punteros directos e indirectos del inodo, de manera que funciones externas no tienen que preocuparse de cómo acceder a los bloques físicos apuntados desde el inodo.

Para desarrollarla, se puede optar tanto por una versión recursiva como por una versión iterativa y utilizar las funciones auxiliares para obtener el **nº de rango de punteros**, el **puntero correspondiente del inodo** y los **índices de los bloques de punteros**.

La misma función nos puede servir tanto para sólo consultar (caso del *mi_read_f()*) como para consultar y reservar un bloque libre si ningún bloque físico es apuntado por el número de bloque lógico (caso del *mi_write_f()*).

Podemos indicar este comportamiento con el argumento *reservar*. De manera genérica:

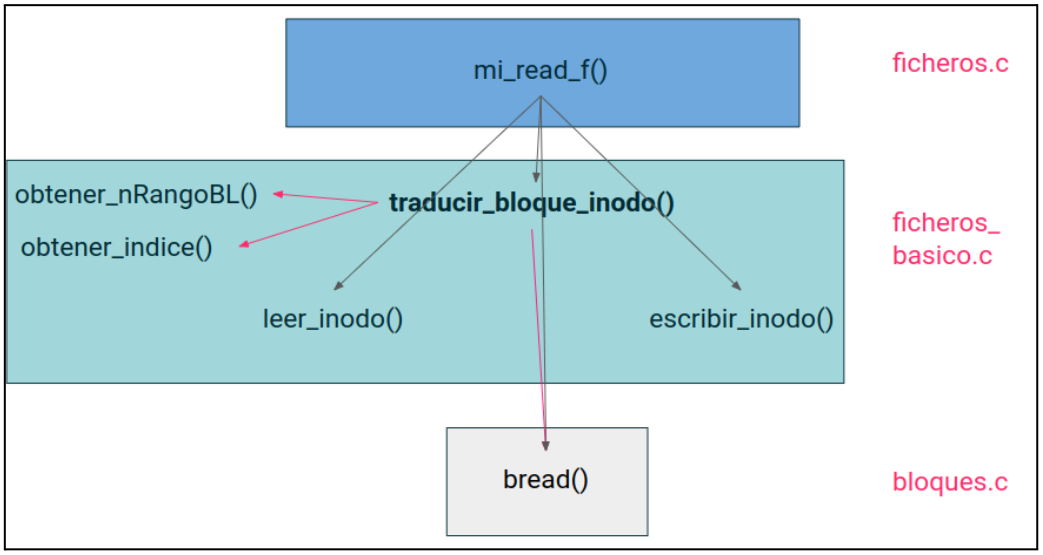
- Si *reservar* vale **0** utilizaremos *traducir_bloque_inodo()* únicamente para **consultar**:
  - Si existe bloque físico de datos, la función devolverá su posición.
  - Si no existe bloque físico de datos, dará error.
- Si *reservar* vale **1** utilizaremos *traducir_bloque_inodo()* para consultar y, si no existe bloque físico, también para **reservar**:
  - Si existe bloque físico de datos, la función devolverá su posición.
  - Si no existe bloque físico de datos, lo reservará y se devolverá su posición.

Hay que tener en cuenta si existen o no los **bloques de punteros intermedios** que precisemos atravesar hasta llegar al **bloque de datos**. En caso de que alguno/s no exista/n:

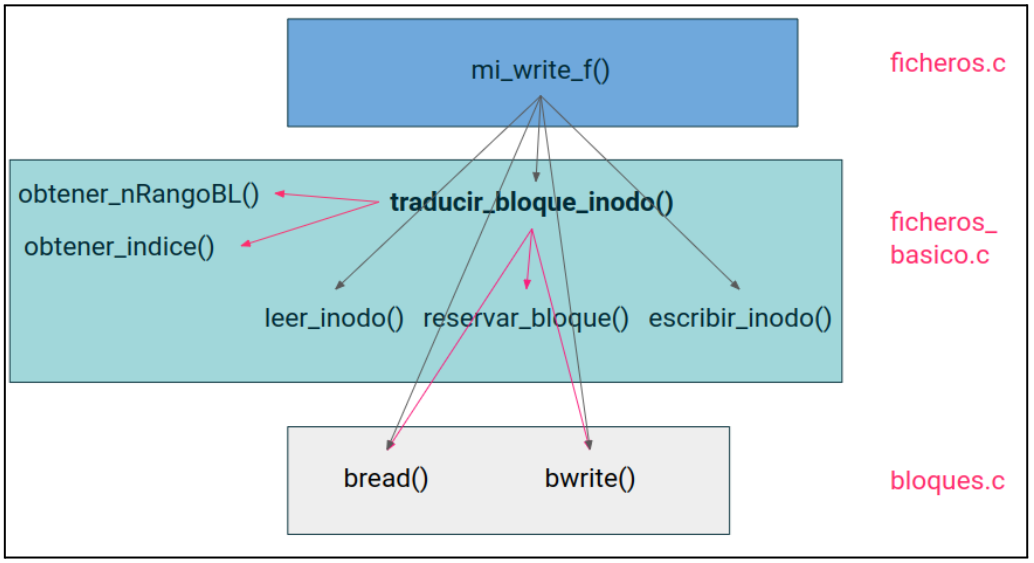
- si estamos en modo de lectura, avisaremos del error,
- y si estamos en modo escritura, habrá que reservarlos (inicializando previamente un buffer con 0s para indicar que no apuntan a nada) y enlazarlos adecuadamente.

Cada vez que reservemos un bloque (sea para datos o para punteros) habrá que incrementar el campo de **número de bloques ocupados** del inodo.

Si se alteran los punteros del inodo y/o la cantidad de bloques ocupados hay que actualizar el *ctime*!



Esquema de llamadas hacia/desde traducir_bloque_inodo() con reservar=0.



Esquema de llamadas hacia/desde traducir_bloque_inodo() con reservar=1.

En el anexo 1 podéis ver el código de la función en C, caso por caso.

Un ejemplo de pseudocódigo para la **función optimizada**[6] sería:

[6] Es la optimización, realizada por Ricardo Galli, del caso por caso que he desarrollado en el Anexo.

```
funcion traducir_bloque_inodo(*inodo:estructura inodo, nblogico:unsigned ent, reservar:
unsigned char) devolver ptr:ent
 var
   ptr, ptr_ant:  unsigned ent
   nRangoBL, nivel_punteros, indice:  ent
   buffer[NPUNTEROS]: unsigned ent
 fvar

 ptr := 0, ptr_ant := 0,
 nRangoBL := obtener_nRangoBL(inodo, nblogico, &ptr); //0:D, 1:I0, 2:I1, 3:I2
 nivel_punteros := nRangoBL //el nivel_punteros +alto es el que cuelga directamente del inodo
 mientras nivel_punteros>0 hacer //iterar para cada nivel de punteros indirectos
   si ptr=0 entonces //no cuelgan bloques de punteros
     si reservar=0 entonces devolver -1 // bloque inexistente -> no imprimir error por pantalla!!!
     si_no //reservar bloques de punteros y crear enlaces desde el  inodo hasta el bloque de datos
       ptr := reservar_bloque() //de punteros
       inodo->numBloquesOcupados++
       inodo->ctime = time(NULL) //fecha actual
       si nivel_punteros = nRangoBL entonces  //el bloque cuelga directamente del inodo
         inodo->punterosIndirectos[nRangoBL-1] := ptr
       si_no   //el bloque cuelga de otro bloque de punteros
         buffer[indice] := ptr
         bwrite(ptr_ant, buffer)  //salvamos en el dispositivo el buffer de punteros modificado
       fsi
       memset(buffer, 0, BLOCKSIZE) //ponemos a 0 todos los punteros del buffer
     fsi
   si_no bread(ptr, buffer) //leemos del dispositivo el bloque de punteros ya existente
   fsi
   indice := obtener_indice(nblogico, nivel_punteros)
   ptr_ant := ptr //guardamos el puntero actual
   ptr := buffer[indice] // y lo desplazamos al siguiente nivel
   nivel_punteros--
 fmientras   //al salir de este bucle ya estamos al nivel de datos

 si ptr=0 //no existe bloque de datos
   si reservar=0 entonces devolver -1  //error lectura ∄ bloque
   si_no
     ptr = reservar_bloque() //de datos
     inodo->numBloquesOcupados++
     inodo->ctime = time(NULL)
     si nRangoBL=0 entonces //si era un puntero Directo
       inodo->punterosDirectos[nblogico] := ptr //asignamos la direción del bl. de datos en el inodo
     si_no
       buffer[indice] := ptr //asignamos la dirección del bloque de datos en el buffer
       bwrite(ptr_ant, buffer) //salvamos en el dispositivo el buffer de punteros modificado
     fsi
   fsi
 fsi
```

```
//mi_write_f() se encargará de salvar los cambios del inodo en disco
 devolver ptr //nº de bloque físico correspondiente al bloque de datos lógico, nblogico
ffuncion
```

Recomendación:

- Haced el seguimiento a mano del algoritmo (con *reservar*=1) para los siguientes bloques lógicos de un mismo inodo: 8, 204, 30.004, 400.004, y 468.750, suponiendo que estamos en la situación inicial de que todos los punteros del inodo apuntan a 0 y no hay nada escrito en la zona de datos del dispositivo virtual. Comprobad los índices obtenidos para cada nivel con los ejemplos proporcionados, y observad cómo se va creando la estructura multinivel de punteros.

Tests de prueba

En este nivel **leer_sf.c** ya no es necesario reservar y liberar un bloque, ni mostrar el mapa de bits, ni el directorio raíz (no borréis las sentencias, tan sólo ponedlas como comentarios o desactivad su impresión usando #if DEBUGNx).

Ahora habrá que añadir a **leer_sf.c** las instrucciones necesarias para comprobar la traducción de bloques de diferentes rangos de punteros.

Para ello necesitaremos previamente reservar artificialmente un inodo con la función *reservar_inodo()*, y llamaremos a la función *traducir_bloque_inodo()* para traducir los bloques lógicos siguientes: 8, 204, 30.004, 400.004 y 468.750

En la función *traducir_bloque_inodo()* de **ficheros_basico.c** hay que poner los *fprintf()* necesarios, justo después de la asignación de un valor a *ptr*, en las 4 situaciones posibles:

- Reservar un **bloque de datos** que cuelgue de punterosDirectos[x].
- Reservar un **bloque de datos** que cuelgue de un **bloque de punteros**.
- Reservar un **bloque de punteros** que cuelgue de punterosIndirectos[x].
- Reservar un **bloque de punteros** que cuelgue de otro **bloque de punteros**.

```
$ ./mi_mkfs disco 100000
$ ./leer_sf disco

DATOS DEL SUPERBLOQUE
posPrimerBloqueMB = 1
posUltimoBloqueMB = 13
posPrimerBloqueAI = 14
posUltimoBloqueAI = 3138
posPrimerBloqueDatos = 3139
posUltimoBloqueDatos = 99999
```

posInodoRaiz = 0
posPrimerInodoLibre = 1
cantBloquesLibres = 96861
cantInodosLibres = 24999
totBloques = 100000
totInodos = 25000

INODO 1. TRADUCCION DE LOS BLOQUES LOGICOS 8, 204, 30.004, 400.004 y 468.750

[traducir_bloque_inodo()→ inodo.punterosDirectos[8] = 3139 (reservado BF 3139 para BL 8)]

[traducir_bloque_inodo()→ inodo.punterosIndirectos[0] = 3140 (reservado BF 3140 para punteros_nivel1)]
[traducir_bloque_inodo()→ punteros_nivel1 [192] = 3141 (reservado BF 3141 para BL 204)]

[traducir_bloque_inodo()→ inodo.punterosIndirectos[1] = 3142 (reservado BF 3142 para punteros_nivel2)]
[traducir_bloque_inodo()→ punteros_nivel2 [116] = 3143 (reservado BF 3143 para punteros_nivel1)]
[traducir_bloque_inodo()→ punteros_nivel1 [40] = 3144 (reservado BF 3144 para BL 30004)]

[traducir_bloque_inodo()→ inodo.punterosIndirectos[2] = 3145 (reservado BF 3145 para punteros_nivel3)]
[traducir_bloque_inodo()→ punteros_nivel3 [5] = 3146 (reservado BF 3146 para punteros_nivel2)]
[traducir_bloque_inodo()→ punteros_nivel2 [25] = 3147 (reservado BF 3147 para punteros_nivel1)]
[traducir_bloque_inodo()→ punteros_nivel1 [120] = 3148 (reservado BF 3148 para BL 400004)]

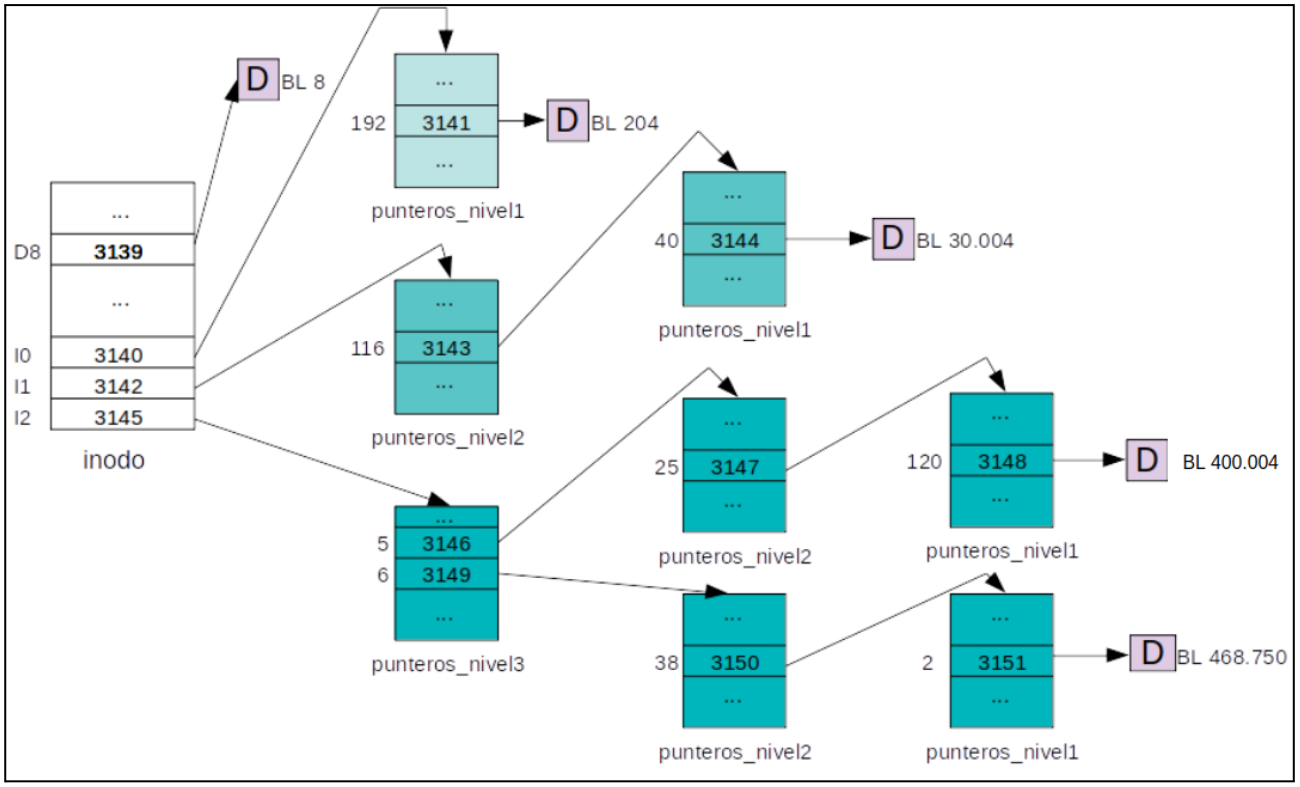
[traducir_bloque_inodo()→ punteros_nivel3 [6] = 3149 (reservado BF 3149 para punteros_nivel2)]
[traducir_bloque_inodo()→ punteros_nivel2 [38] = 3150 (reservado BF 3150 para punteros_nivel1)]
[traducir_bloque_inodo()→ punteros_nivel1 [2] = 3151 (reservado BF 3151 para BL 468750)]

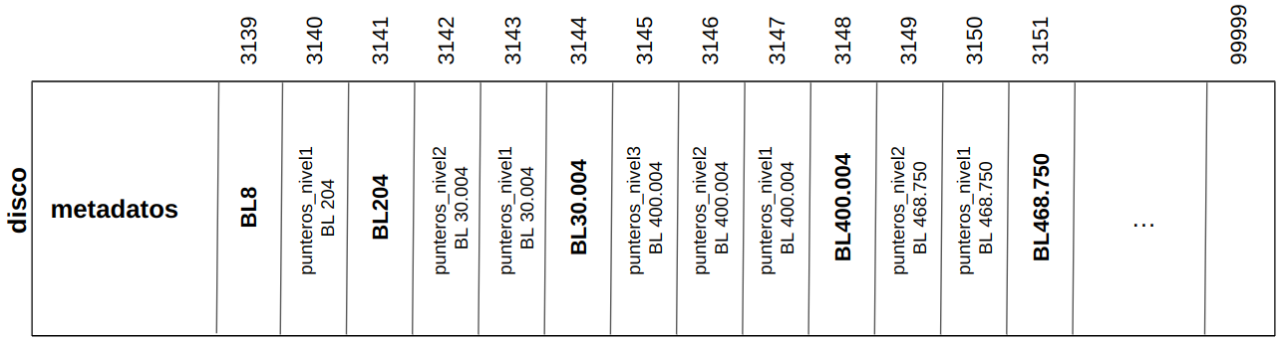
DATOS DEL INODO RESERVADO 1
tipo: f
permisos: 6
atime: Wed 2021-03-10 17:09:42
ctime: Wed 2021-03-10 17:09:42
mtime: Wed 2021-03-10 17:09:42
nlinks: 1
tamEnBytesLog: 0
numBloquesOcupados: 13 [7]

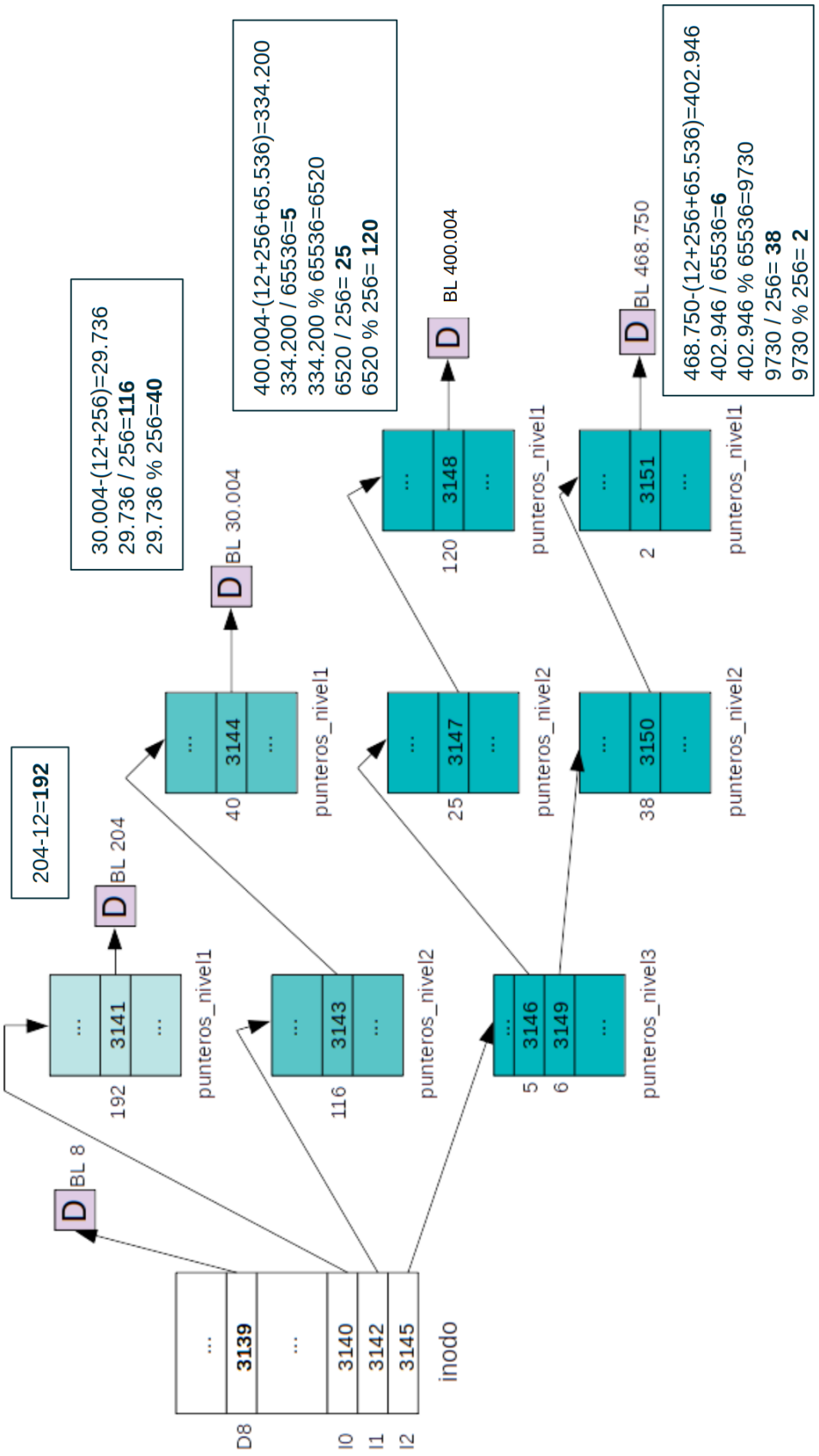
SB.posPrimerInodoLibre = 2

El siguiente gráfico ilustra esta ejecución:

[7] Los ha tenido que crear porque no existían pero no contienen nada. Para hacer esta prueba estamos llamando a *traducir_bloque_inodo()* con *reservar*=1 para ver que cree bien los punteros. Después borraremos en **leer_sf.c** esas llamadas ya que en la práctica sólo tiene sentido llamar a *traducir_bloque_inodo()* con *reservar*=1 cuando vayamos a escribir contenido en los ficheros o a crear entradas en los directorios, ambas acciones desde una capa superior de nuestra biblioteca mediante la función *mi_write_f()*.


D BL 8
192
3141
D BL 204
punteros_nivel1
40
3144
D BL 30.004
punteros_nivel1
D8
3139
I0
3140
I1
3142
I2
3145
inodo
116
3143
punteros_nivel2
5
3146
6
3149
punteros_nivel3
25
3147
punteros_nivel2
120
3148
D BL 400.004
punteros_nivel1
38
3150
punteros_nivel2
2
3151
D BL 468.750
punteros_nivel1


disco
metadatos
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149
3150
3151
99999
BL8
punteros_nivel1 BL 204
BL204
punteros_nivel2 BL 30.004
punteros_nivel1 BL 30.004
BL30.004
punteros_nivel3 BL 400.004
punteros_nivel2 BL 400.004
punteros_nivel1 BL 400.004
BL400.004
punteros_nivel2 BL 468.750
punteros_nivel1 BL 468.750
BL468.750
...


inodo
D8
I0
I1
I2
3139
3140
3142
3145
BL 8
BL 204
204-12=192
192
punteros_nivel1
3141
116
punteros_nivel2
3143
5
6
punteros_nivel3
3146
3149
BL 30.004
30.004-(12+256)=29.736
29.736 / 256=116
29.736 % 256=40
40
3144
BL 400.004
400.004-(12+256+65.536)=334.200
334.200 / 65536=5
334.200 % 65536=6520
6520 / 256= 25
6520 % 256= 120
25
3147
120
3148
BL 468.750
468.750-(12+256+65.536)=402.946
402.946 / 65536=6
402.946 % 65536=9730
9730 / 256= 38
9730 % 256= 2
38
3150
2
3151

Anexo 1: versión caso por caso sin compactar código

```
int traducir_bloque_inodo(*inodo:estructura inodo,  unsigned int nblogico, unsigned char reservar){
  unsigned int ptr = 0;

  if (nblogico < DIRECTOS){ // el bloque logico es uno de los 12 primeros bloques logicos del inodo.
     switch (reservar){
        case 0:// modo consulta
           if (inodo->punterosDirectos[nblogico] == 0)// no tiene bloque físico asignado.
              return -1;
           else{
              ptr=inodo->punterosDirectos[nblogico];
           }
           break;
        case 1:// modo escritura
           if (inodo->punterosDirectos[nblogico] == 0){// si no tiene bloque fisico le asignamos uno.
              inodo->punterosDirectos[nblogico] = reservar_bloque();
              ptr=inodo->punterosDirectos[nblogico];
             // aumentamos en uno el numero de bloques ocupados por el inodo en la zona de datos:
              inodo->numBloquesOcupados++;
              inodo->ctime = time(NULL);
           }
           else{ // tiene bloque fisico asignado y lo devolvemos
              ptr=inodo->punterosDirectos[nblogico];
           }
           break;
     }
     //printf("nblogico= %d, ptr= %d\n", nblogico, ptr);
     return ptr;
  }
  // PUNTERO INDIRECTOS 0
  // El bloque logico lo encontramos en el rango de Indirectos 0, es decir,
  // está comprendido entre   el 0+12 y el 0+12+256-1: entre el 12 y el 267
  else if (nblogico < INDIRECTOS0){
     unsigned int punteros_nivel1[NPUNTEROS];
     switch (reservar){
        case 0:// modo consulta
           if (inodo->punterosIndirectos[0] == 0)// no hay bloque fisico asignado a punteros_nivel1.
              return -1;
           else{// ya existe el bloque de punteros_nivel1 y lo leemos del dispositivo
              bread(inodo->punterosIndirectos[0],punteros_nivel1);
              if (punteros_nivel1[nblogico-DIRECTOS] == 0)
              // no hay bloque físico asignado al bloque lógico de datos.
                 return -1;
              else{
                 ptr=punteros_nivel1[nblogico-DIRECTOS];
              }
           }
           break;
```

```
        case 1:// modo escritura
            if (inodo->punterosIndirectos[0] == 0){ // no hay bloque fisico asignado a punteros_nivel1
                inodo->punterosIndirectos[0] = reservar_bloque(); //para punteros_nivel1
                memset(punteros_nivel1,0,BLOCKSIZE); //iniciamos a 0 los 256 punteros
                punteros_nivel1[nblogico-DIRECTOS] = reservar_bloque(); //para datos
                // aumentamos el numero de bloques ocupados por el inodo en la zona de datos
                inodo->numBloquesOcupados+=2;
                inodo->ctime = time(NULL);
                // salvamos el bloque de punteros_nivel1 en el dispositivo.
                bwrite(inodo->punterosIndirectos[0],punteros_nivel1);
                // devolvemos el bloque fisico de datos
                ptr=punteros_nivel1[nblogico-DIRECTOS];
            }
            else{// existe el bloque de punteros_nivel1 y lo leemos del dispositivo
                bread(inodo->punterosIndirectos[0],punteros_nivel1);
                if (punteros_nivel1[nblogico-DIRECTOS] == 0){
                    // no hay bloque fisico de datos asignado, entonces lo reservamos
                    punteros_nivel1[nblogico-DIRECTOS] = reservar_bloque(); //para datos
                    // salvamos el bloque de punteros_nivel1 en el dispositivo.
                    bwrite(inodo->punterosIndirectos[0],punteros_nivel1);
                    ptr=punteros_nivel1[nblogico-DIRECTOS]; // devolvemos el bloque fisico de datos.
                    // aumentamos el numero de bloques ocupados por el inodo en la zona de datos.
                    inodo->numBloquesOcupados++;
                    inodo->ctime = time(NULL);
                }
                else{ // si existe el bloque fisico de datos lo devolvemos
                    ptr=punteros_nivel1[nblogico-DIRECTOS];
                }
            }
            break;
    }
    //printf("nblogico= %d, ptr= %d\n", nblogico, ptr);
    return ptr;
}
// PUNTERO INDIRECTOS 1
// El bloque logico lo encontramos en el rango de Indirectos 1, es decir,
// los comprendidos entre el 0+12+256 y el 0+12+256+256^2-1: entre el 268 y el 65.803.
else if (nblogico < INDIRECTOS1){
    unsigned int punteros_nivel1[NPUNTEROS];
    unsigned int punteros_nivel2[NPUNTEROS];
    unsigned int indice_nivel1 = obtener_indice(nblogico, 1); // indice para punteros_nivel1
    unsigned int indice_nivel2 = obtener_indice(nblogico, 2); // indice para punteros_nivel2
    switch(reservar){
        case 0:// modo consulta
            if (inodo->punterosIndirectos[1] == 0)// no hay bloque fisico asignado a punteros_nivel2.
                return -1;
            else{// ya existe el bloque de punteros_nivel2 y lo leemos del dispositivo
                bread(inodo->punterosIndirectos[1],punteros_nivel2);
                if (punteros_nivel2[indice_nivel2] == 0)// no hay bloque fisico asignado a punteros_nivel1.
                    return -1;
                else{// ya existe el bloque de punteros_nivel1 y lo leemos del dispositivo
```

```
            bread(punteros_nivel2[indice_nivel2],punteros_nivel1);
            if (punteros_nivel1[indice_nivel1] == 0)
             // no hay bloque físico asignado al bloque lógico de datos.
               return -1;
            else{
               ptr=punteros_nivel1[indice_nivel1]; // devolvemos el bloque fisico solicitado
            }
         }
      }
      break;
   case 1:// modo escritura
      if (inodo->punterosIndirectos[1] == 0){ // no hay bloque fisico asignado a punteros_nivel2
         inodo->punterosIndirectos[1] = reservar_bloque(); //para punteros_nivel2
         memset(punteros_nivel2,0,BLOCKSIZE); //iniciamos a 0 los 256 punteros de nivel2
         punteros_nivel2[indice_nivel2] = reservar_bloque(); //para punteros_nivel1
         memset(punteros_nivel1,0,BLOCKSIZE); //iniciamos a 0 los 256 punteros de nivel1
         punteros_nivel1[indice_nivel1] = reservar_bloque();  //para datos
         // salvamos los buffers de los bloques de punteros en el dispositivo
         bwrite(inodo->punterosIndirectos[1],punteros_nivel2);
         bwrite(punteros_nivel2[indice_nivel2],punteros_nivel1);
         // devolvemos el bloque fisico de datos
         ptr=punteros_nivel1[indice_nivel1];
         // aumentamos el numero de bloques ocupados por el inodo en la zona de datos.
         inodo->numBloquesOcupados+=3;
         inodo->ctime = time(NULL);
      }
      else{// existe el bloque de punteros_nivel2 y lo leemos del dispositivo
         bread(inodo->punterosIndirectos[1],punteros_nivel2);
         if (punteros_nivel2[indice_nivel2] == 0){ // no hay bloque fisico asignado a punteros_nivel1
            punteros_nivel2[indice_nivel2] = reservar_bloque();  //para punteros_nivel1
            memset(punteros_nivel1,0,BLOCKSIZE); //iniciamos a 0 los 256 punteros de nivel1
            punteros_nivel1[indice_nivel1] = reservar_bloque(); //para datos
            // salvamos los buffers de los bloques de punteros en el dispositivo
            bwrite(inodo->punterosIndirectos[1],punteros_nivel2);
            bwrite(punteros_nivel2[indice_nivel2],punteros_nivel1);
            // devolvemos el bloque fisico de datos
            ptr=punteros_nivel1[indice_nivel1];
            // aumentamos el numero de bloques ocupados por el inodo en la zona de datos.
            inodo->numBloquesOcupados+=2;
            inodo->ctime = time(NULL);
         }
         else{// existe el bloque de punteros_nivel1 y lo leemos del dispositivo
            bread(punteros_nivel2[indice_nivel2],punteros_nivel1);
            if (punteros_nivel1[indice_nivel1] == 0){  // no hay bloque físico asignado al bloque de datos
               punteros_nivel1[indice_nivel1] = reservar_bloque(); //para datos
               // salvamos el bloque de punteros_nivel1 en el dispositivo
               bwrite(punteros_nivel2[indice_nivel2],punteros_nivel1);
               // devolvemos el bloque físico asignado a los datos
               ptr = punteros_nivel1[indice_nivel1];
               // aumentamos en uno el numero de bloques ocupados por el inodo en la zona de datos.
               inodo->numBloquesOcupados++;
```

```
                inodo->ctime = time(NULL);
            }
            else{
                ptr = punteros_nivel1[indice_nivel1];
            }
        }
    }
    // escribimos en el dispositivo el inodo actualizado
    escribir_inodo(ninodo,inodo);
    break;
  }
  //printf("nblogico= %d, ptr= %d\n", nblogico, ptr);
  return ptr;
}
// PUNTERO INDIRECTOS 2
// El bloque logico lo encontramos en el rango de Indirectos 2, es decir, los comprendidos entre
// el 0+12+256+256^2 y el 0+12+256+256^2+256^3-1: entre el 65.804 y el 16.843.019.
else if (nblogico < INDIRECTOS2){
  unsigned int punteros_nivel1[NPUNTEROS];
  unsigned int punteros_nivel2[NPUNTEROS];
  unsigned int punteros_nivel3[NPUNTEROS];
  unsigned int indice_nivel1 = obtener_indice(nblogico, 1); //indice para punteros_nivel1
  unsigned int indice_nivel2 = obtener_indice(nblogico, 2); //indice para punteros_nivel2
  unsigned int indice_nivel3 = obtener_indice(nblogico, 3); //indice para punteros_nivel3
  switch(reservar){
    case 0://modo consulta
      if (inodo->punterosIndirectos[2] == 0)// no hay bloque fisico asignado a punteros_nivel3.
        return -1;
      else{// ya existe el bloque de punteros_nivel3 y lo leemos del dispositivo
        bread(inodo->punterosIndirectos[2],punteros_nivel3);
        if (punteros_nivel3[indice_nivel3] == 0)// no hay bloque fisico asignado a punteros_nivel2.
         return -1;
        else{// ya existe el bloque de punteros_nivel2 y lo leemos del dispositivo
          bread(punteros_nivel3[indice_nivel3],punteros_nivel2);
          if (punteros_nivel2[indice_nivel2] == 0)// no hay bloque fisico asignado a punteros_nivel1.
            return -1;
          else{// ya existe el bloque de punteros_nivel1 y lo leemos del dispositivo
            bread(punteros_nivel2[indice_nivel2],punteros_nivel1);
            if (punteros_nivel1[indice_nivel1] == 0)
            // no hay bloque físico asignado al bloque lógico de datos.
              return -1;
            else{
              ptr=punteros_nivel1[indice_nivel1];// devolvemos el bloque fisico solicitado
            }
          }
        }
      }
      break;
    case 1://modo escritura
      if (inodo->punterosIndirectos[2] == 0){ // no hay bloque fisico asignado a punteros_nivel3
        inodo->punterosIndirectos[2] = reservar_bloque();  //para punteros_nivel3
```

```
            memset(punteros_nivel3,0,BLOCKSIZE); //iniciamos a 0 los 256 punteros de nivel3
            punteros_nivel3[indice_nivel3] = reservar_bloque(); //para punteros_nivel2
            memset(punteros_nivel2,0,BLOCKSIZE); //iniciamos a 0 los 256 punteros de nivel2
            punteros_nivel2[indice_nivel2] = reservar_bloque();  //para punteros_nivel1
            memset(punteros_nivel1,0,BLOCKSIZE);  //iniciamos a 0 los 256 punteros de nivel1
            punteros_nivel1[indice_nivel1] = reservar_bloque();  //para datos
            // salvamos los buffers de los bloques de punteros en el dispositivo
            bwrite(inodo->punterosIndirectos[2],punteros_nivel3);
            bwrite(punteros_nivel3[indice_nivel3],punteros_nivel2);
            bwrite(punteros_nivel2[indice_nivel2],punteros_nivel1);
            // devolvemos el bloque fisico de datos
            ptr=punteros_nivel1[indice_nivel1];
            // aumentamos en uno el numero de bloques ocupados por el inodo en la zona de datos.
            inodo->numBloquesOcupados+=4;
            inodo->ctime = time(NULL);
        }
        else{// existe el bloque de punteros_nivel3 y lo leemos del dispositivo
            bread(inodo->punterosIndirectos[2],punteros_nivel3);
            if (punteros_nivel3[indice_nivel3] == 0){ // no hay bloque fisico asignado a punteros_nivel2
                punteros_nivel3[indice_nivel3] = reservar_bloque();  //para punteros_nivel2
                memset(punteros_nivel2,0,BLOCKSIZE);   //iniciamos a 0 los 256 punteros de nivel2
                punteros_nivel2[indice_nivel2] = reservar_bloque(); //para punteros_nivel1
                memset(punteros_nivel1,0,BLOCKSIZE);  //iniciamos a 0 los 256 punteros de nivel1
                punteros_nivel1[indice_nivel1] = reservar_bloque();  //para datos
                // salvamos los buffers de los bloques de punteros en el dispositivo
                bwrite(inodo->punterosIndirectos[2],punteros_nivel3);
                bwrite(punteros_nivel3[indice_nivel3],punteros_nivel2);
                bwrite(punteros_nivel2[indice_nivel2],punteros_nivel1);
                // devolvemos el bloque fisico de datos
                ptr=punteros_nivel1[indice_nivel1];
                // aumentamos el numero de bloques ocupados por el inodo en la zona de datos.
                inodo->numBloquesOcupados+=3;
                inodo->ctime = time(NULL);
            }
            else{// existe el bloque de punteros_nivel2 y lo leemos del dispositivo
                bread(punteros_nivel3[indice_nivel3],punteros_nivel2);
                if (punteros_nivel2[indice_nivel2] == 0){ // no hay bloque fisico asignado a punteros_nivel1
                    punteros_nivel2[indice_nivel2] = reservar_bloque(); //para punteros_nivel1
                    memset(punteros_nivel1,0,BLOCKSIZE); //iniciamos a 0 los 256 punteros de nivel1
                    punteros_nivel1[indice_nivel1] = reservar_bloque();  //para datos
                    // salvamos los buffers de los bloques de punteros en el dispositivo
                    bwrite(punteros_nivel3[indice_nivel3],punteros_nivel2);
                    bwrite(punteros_nivel2[indice_nivel2],punteros_nivel1);
                     // devolvemos el bloque fisico de datos
                    ptr=punteros_nivel1[indice_nivel1];
                    // aumentamos el numero de bloques ocupados por el inodo en la zona de datos.
                    inodo->numBloquesOcupados+=2;
                    inodo->ctime = time(NULL);
                }
                else{// existe el bloque de punteros_nivel1 y lo leemos del dispositivo
                    bread(punteros_nivel2[indice_nivel2],punteros_nivel1);
```

```
                if (punteros_nivel1[indice_nivel1] == 0){//no hay bloque físico asignado al bloque de datos
                    punteros_nivel1[indice_nivel1] = reservar_bloque();  //para datos
                    // salvamos el bloque de punteros_nivel1 en el dispositivo
                    bwrite(punteros_nivel2[indice_nivel2],punteros_nivel1);
                    // devolvemos el bloque físico asignado a los datos
                    ptr=punteros_nivel1[indice_nivel1];
                    //aumentamos el numero de bloques ocupados por el inodo en la zona de datos.
                    inodo->numBloquesOcupados++;
                    inodo->ctime = time(NULL);
                }
                else{
                    ptr=punteros_nivel1[indice_nivel1];
                }
            }
        }
    }

    break;
  }
  //printf("nblogico= %d, ptr= %d\n", nblogico, ptr);
  return ptr;
}
return ptr;
}
```