Ejercicios

Sistemas de ficheros

Grupo ARCOS

Diseño de Sistemas Operativos

Grado en Ingeniería Informática

Universidad Carlos III de Madrid

cc BY NC SA

Ejercicio
enunciado (1/3)

Disponemos de una maquina monoprocesador y queremos implementar un sistema de ficheros para un sistema operativo UNIX con un *kernel* monolítico no expulsivo con la siguiente funcionalidad:

- Reservar un bloque inicial para una posible tabla de particiones futura, que por ahora el bloque está relleno con ceros.
- El superbloque ocupará un bloque de disco.
- Para la gestión de espacio libre se usará un mapa de bytes, usando un byte con valor 0 para indicar libre y 1 para indicar ocupado.
- La suma de ficheros y directorios es como máximo de numInodo. El i-nodo asociado a cada entrada ocupará un bloque.

Ejercicio
enunciado (2/3)

- El nombre de una entrada tiene como máximo 200 caracteres.
- Cada fichero solo tendrá un bloque de datos asociado.
- El número máximo de bloques de datos será de numBloquesDatos.
- Solo hay un directorio raíz, no hay subdirectorios, no obstante se diseñará el sistema en disco para tener 200 entradas en un directorio como máximo, que se guardarán en el i-nodo asociado al directorio.
- Cada fichero tiene su puntero de lectura y escritura (no compartido), y no se podrá desmontar el sistema de ficheros si hay alguno abierto.

Ejercicio
enunciado (3/3)

Se pide:

a) Diseñar las estructuras en disco que permitan satisfacer con los requisitos pedidos de forma simple y fácil de entender.

b) Diseñar las estructuras en memoria que permitan satisfacer la funcionalidad pedida.

c) Diseñar las funciones de tratamiento de bloques (alloc, free, bmap) e i-nodos (ialloc, ifree y namei).

d) Diseñar las funciones de interfaz de sistema mount, umount, open, close, creat, unlink, read, write así como la utilidad mkfs.

Ejercicio
solución

1. Planteamiento inicial
    1. Estado inicial del sistema
    2. Estudio de qué hay que modificar
2. Responder a las preguntas
3. Revisar las respuestas

Ejercicio
solución

1. Planteamiento inicial
    1. Estado inicial del sistema
    2. Estudio de qué hay que modificar
2. Responder a las preguntas
3. Revisar las respuestas

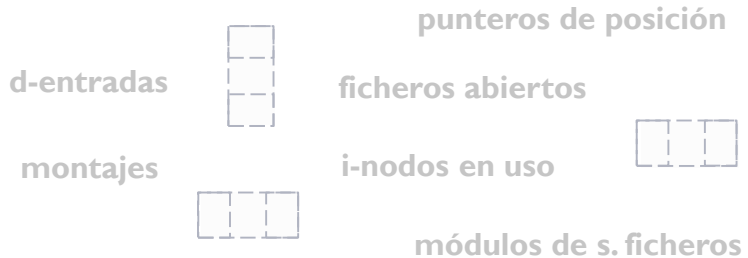
Planteamiento general

Llamadas al sistema de archivos

Descriptor	Uso de *namei*	Asig. i-n.	Atributos	E/S.	Sist. Arch.	Vista
open pipe creat close dup	open chown unlink creat chmod mknod chdir stat mount chroot link umount	creat mknod link unlink	chown chmod stat	read write lseek	mount umount	chdir chroot

Algoritmos de bajo nivel del sistema de archivos

namei		ialloc	alloc	
iget	iput	ifree	free	bmap



Algoritmos de gestión de bloques/caché

getblk	brelse	bread	breada	bwrite

Planteamiento
apartado a)

Llamadas al sistema de archivos

Descriptor	Uso de *namei*	Asig. i-n.	Atributos	E/S.	Sist. Arch.	Vista
open pipe creat close dup	open chown unlink creat chmod mknod chdir stat mount chroot link umount	creat mknod link unlink	chown chmod stat	read write lseek	mount umount	chdir chroot

Algoritmos de bajo nivel del sistema de archivos

namei		ialloc	alloc	
iget	iput	ifree	free	bmap

punteros de posición

d-entradas

ficheros abiertos

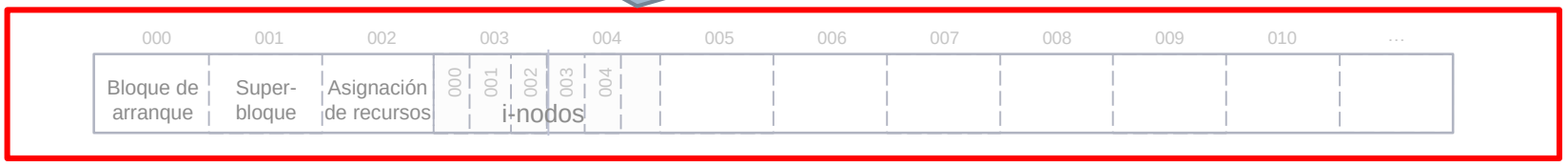
montajes

i-nodos en uso

Algoritmos de gestión de

getblk brelse

1. Diseñar la organización en disco



Planteamiento
apartado b)

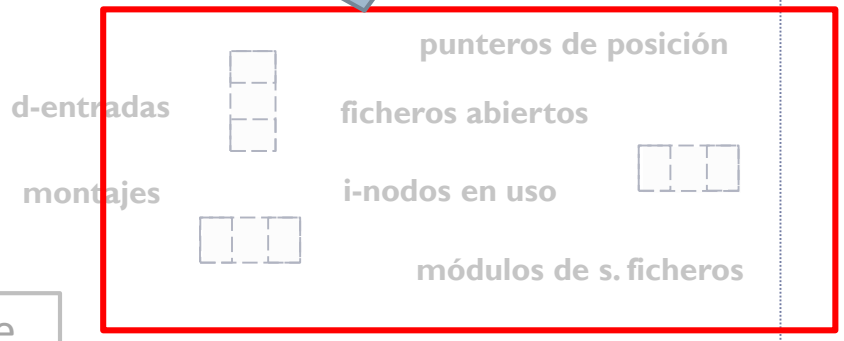
Llamadas al sistema de archivos

Descriptor	Uso de *namei*	Asig. i-n.	Atributos	E/S.	Sist. Arch.	Vista
op cre dup	chdir stat mount chroot link umount	link unlink	st			

2. Crear variables para las estructuras en memoria

Algoritmos de bajo nivel del sistema de archivos

namei		ialloc	alloc		bmap
iget	iput	ifree	free		



Algoritmos de gestión de bloques/caché

getblk	brelse	bread	breada	bwrite

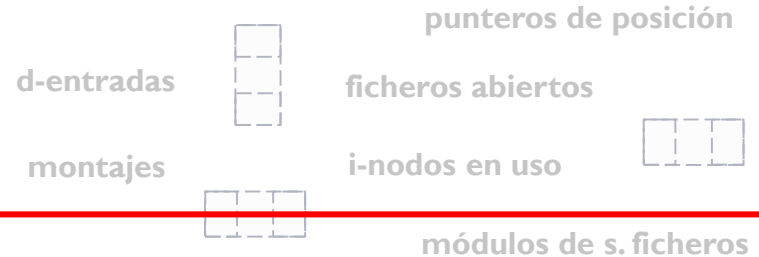
Planteamiento
apartado c)

Llamadas al sistema de archivos

Descriptor	Uso de *namei*	Asig. i-n.	Atributos	E/S.	Sist. Arch.	Vista
open pipe creat close dup	open chown unlink creat chmod mknod chdir stat mount chroot link umount	creat mknod link unlink	chown chmod stat	read write lseek	mount umount	chdir chroot

Algoritmos de bajo nivel del sistema de archivos

namei		ialloc	alloc	
iget	iput	ifree	free	bmap



Algoritmos de gestión de bloq[illegible]hé

4. Diseñar las rutinas de gestión
   - Lectura y escritura en disco de estructuras en memoria

Planteamiento
apartado d)

Llamadas al sistema de archivos

Descriptor	Uso de *namei*	Asig. i-n.	Atributos	E/S.	Sist. Arch.	Vista
open pipe creat close dup	open chown unlink creat chmod mknod chdir stat mount chroot link umount	creat mknod link unlink	chown chmod stat	read write lseek	mount umount	chdir chroot

Algoritmos de bajo nivel del sistema de archivos

namei	ialloc	alloc
iget iput	ifree	bmap

Ejercicio
solución

1. Planteamiento inicial
    1. Estado inicial del sistema
    2. Estudio de qué hay que modificar
2. Responder a las preguntas
3. Revisar las respuestas

Ejercicio
solución a)

Llamadas al sistema de archivos

Descriptor	Uso de *namei*	Asig. i-n.	Atributos	E/S.	Sist. Arch.	Vista
open pipe creat close dup	open chown unlink creat chmod mknod chdir stat mount chroot link umount	creat mknod link unlink	chown chmod stat	read write lseek	mount umount	chdir chroot

Algoritmos de bajo nivel del sistema de archivos

namei		ialloc	alloc	
iget	iput	ifree	free	bmap

punteros de posición

d-entradas

ficheros abiertos

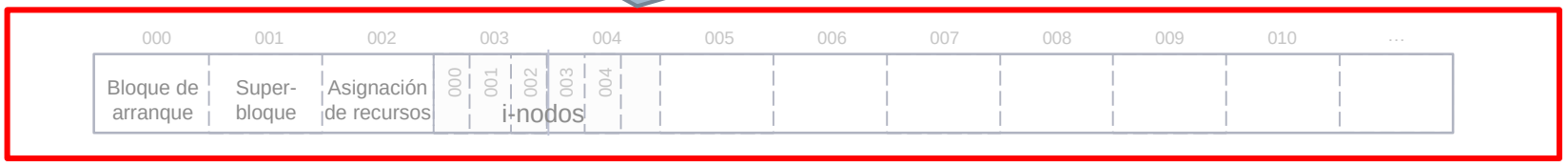
montajes

i-nodos en uso

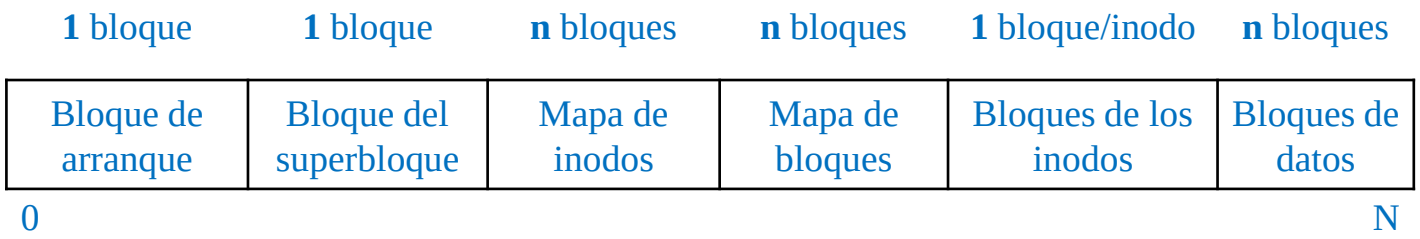
Algoritmos de gestión de

getblk brelse

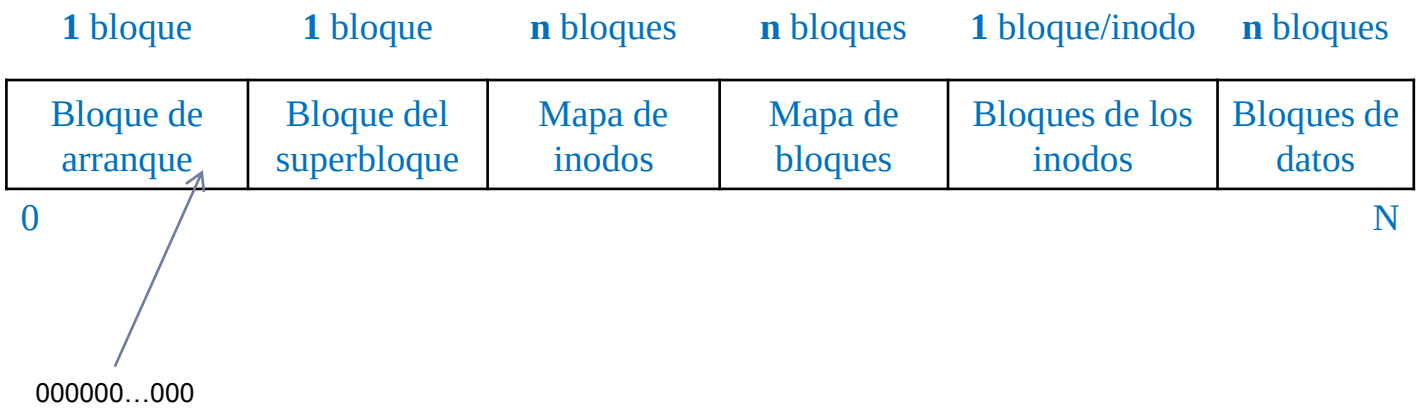
1. Diseñar la organización en disco



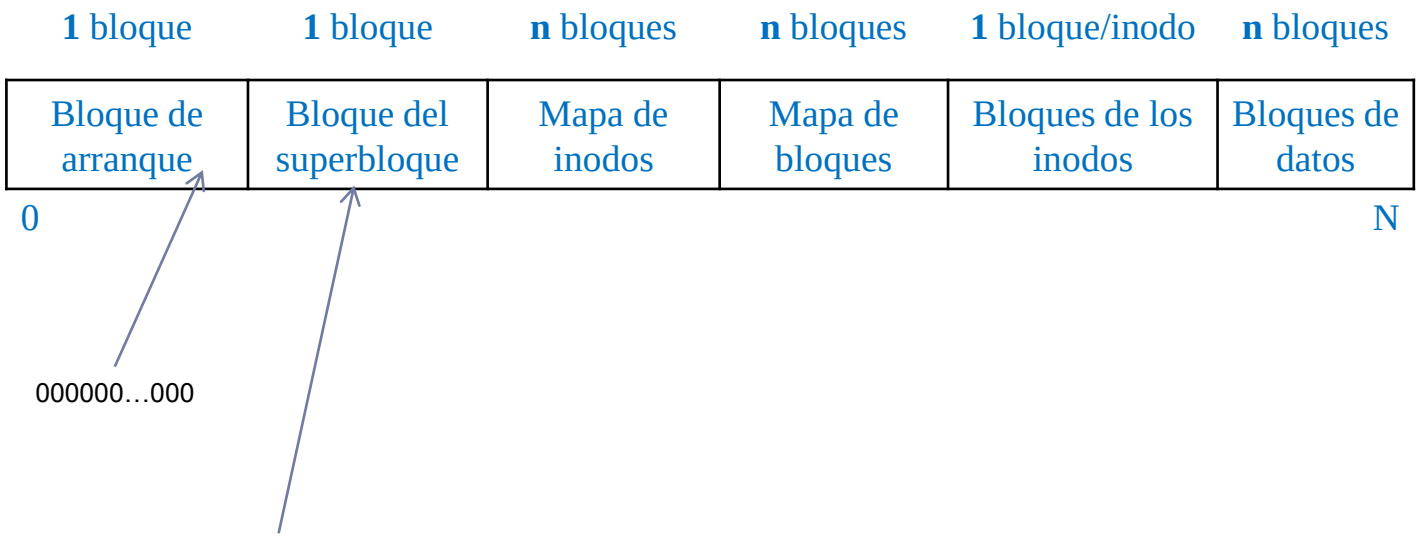
Propuesta de diseño de estructura en disco



Propuesta de diseño de estructura en disco

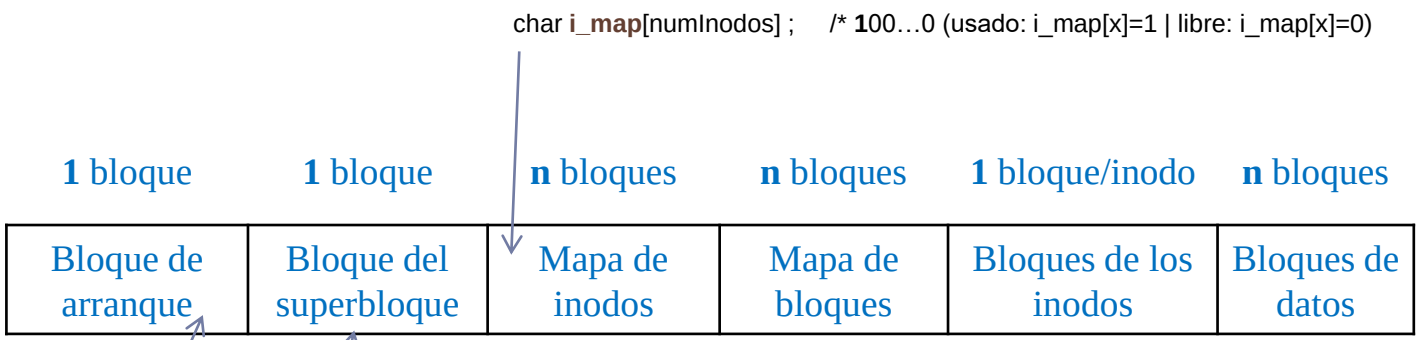


Propuesta de diseño de estructura en disco



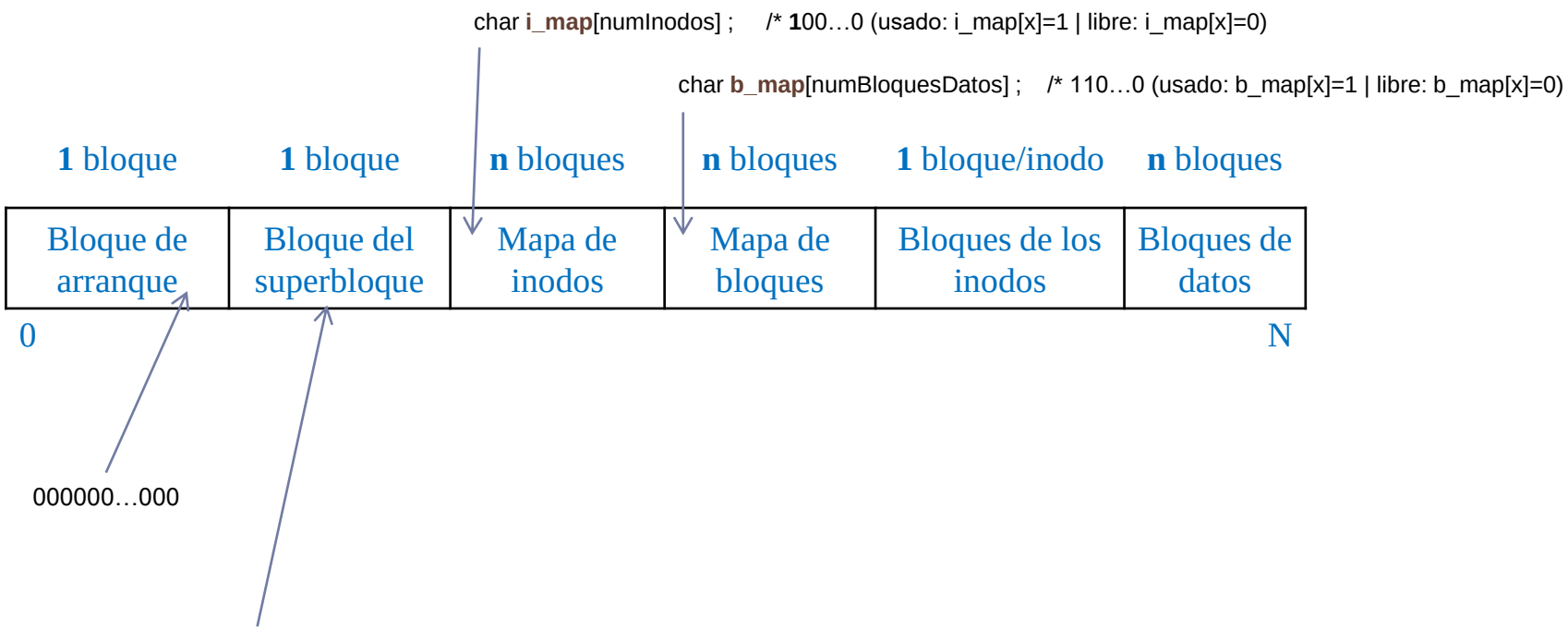

```
typedef struct {
   unsigned int numMagico;                /* Número mágico del superbloque: 0x000D5500 */
   unsigned int numBloquesMapaInodos;     /* Número de bloques del mapa inodos */
   unsigned int numBloquesMapaDatos;      /* Número de bloques del mapa  datos */
   unsigned int numInodos;                /* Número de inodos en el dispositivo */
   unsigned int primerInodo;              /* Número bloque del 1º inodo del disp. (inodo raíz) */
   unsigned int numBloquesDatos;          /* Número de bloques de datos en el disp. */
   unsigned int primerBloqueDatos;        /* Número de bloque del 1º bloque de datos */
   unsigned int tamDispositivo;           /* Tamaño total del disp. (en bytes) */
   char relleno[PADDING_SB];              /* Campo de relleno (para completar un bloque) */
} TipoSuperbloque;
```

Propuesta de diseño de estructura en disco



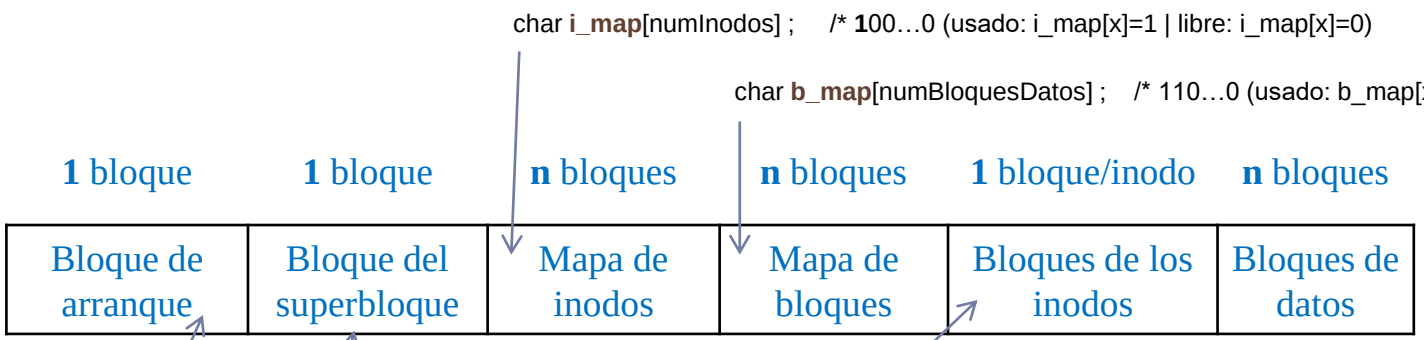

```
typedef struct {
   unsigned int numMagico;                /* Número mágico del superbloque: 0x000D5500 */
   unsigned int numBloquesMapaInodos;     /* Número de bloques del mapa inodos */
   unsigned int numBloquesMapaDatos;      /* Número de bloques del mapa  datos */
   unsigned int numInodos;                /* Número de inodos en el dispositivo */
   unsigned int primerInodo;              /* Número bloque del 1º inodo del disp. (inodo raíz) */
   unsigned int numBloquesDatos;          /* Número de bloques de datos en el disp. */
   unsigned int primerBloqueDatos;        /* Número de bloque del 1º bloque de datos */
   unsigned int tamDispositivo;           /* Tamaño total del disp. (en bytes) */
   char relleno[PADDING_SB];              /* Campo de relleno (para completar un bloque) */
} TipoSuperbloque;
```

Propuesta de diseño de estructura en disco




```
typedef struct {
   unsigned int numMagico;                /* Número mágico del superbloque: 0x000D5500 */
   unsigned int numBloquesMapaInodos;     /* Número de bloques del mapa inodos */
   unsigned int numBloquesMapaDatos;      /* Número de bloques del mapa  datos */
   unsigned int numInodos;                /* Número de inodos en el dispositivo */
   unsigned int primerInodo;              /* Número bloque del 1º inodo del disp. (inodo raíz) */
   unsigned int numBloquesDatos;          /* Número de bloques de datos en el disp. */
   unsigned int primerBloqueDatos;        /* Número de bloque del 1º bloque de datos */
   unsigned int tamDispositivo;           /* Tamaño total del disp. (en bytes) */
   char relleno[PADDING_SB];              /* Campo de relleno (para completar un bloque) */
} TipoSuperbloque;
```

Propuesta de diseño de estructura en disco




```
typedef struct {
   unsigned int tipo;                  /*  T_FICHERO o T_DIRECTORIO */
   char nombre[200];                   /* Nombre del fichero/directorio asociado */
   unsigned int inodosContenidos[200]; /* tipo==dir: lista de los inodos del directorio */
   unsigned int tamanyo;               /* Tamaño actual del fichero en bytes */
   unsigned int bloqueDirecto;         /* Número del bloque directo */
   char relleno[PADDING_INODO];        /* Campo relleno para llenar un bloque */
} TipoInodoDisco;
```

```
typedef struct {
   unsigned int numMagico;             /* Número mágico del superbloque: 0x000D5500 */
   unsigned int numBloquesMapaInodos;  /* Número de bloques del mapa inodos */
   unsigned int numBloquesMapaDatos;   /* Número de bloques del mapa  datos */
   unsigned int numInodos;             /* Número de inodos en el dispositivo */
   unsigned int primerInodo;           /* Número bloque del 1º inodo del disp. (inodo raíz) */
   unsigned int numBloquesDatos;       /* Número de bloques de datos en el disp. */
   unsigned int primerBloqueDatos;     /* Número de bloque del 1º bloque de datos */
   unsigned int tamDispositivo;        /* Tamaño total del disp. (en bytes) */
   char relleno[PADDING_SB];           /* Campo de relleno (para completar un bloque) */
} TipoSuperbloque;
```

Ejercicio
solución b)

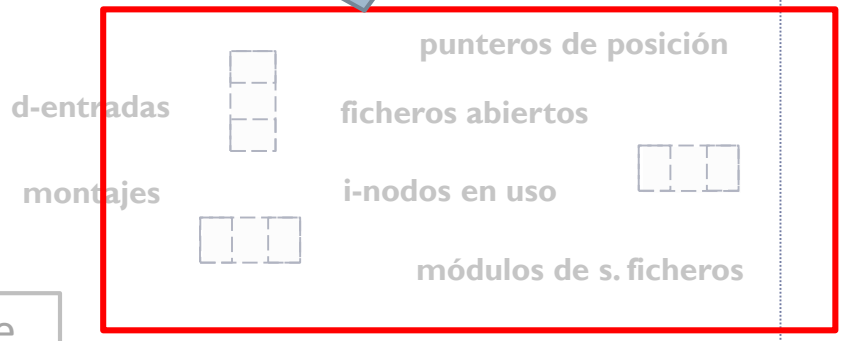
Llamadas al sistema de archivos

Descriptor	Uso de *namei*	Asig. i-n.	Atributos	E/S.	Sist. Arch.	Vista
op cre dup	chdir stat mount chroot link umount	link unlink	st			

2. Crear variables para las estructuras en memoria

Algoritmos de bajo nivel del sistema de archivos

namei		ialloc	alloc	
iget	iput	ifree	free	bmap



Algoritmos de gestión de bloques/caché

getblk	brelse	bread	breada	bwrite

Propuesta de diseño de estructuras en memoria

```
// Información leída del disco
TipoSuperbloque sbloques [1] ;
char i_map [numInodo] ;
char b_map [numBloquesDatos] ;
TipoInodoDisco inodos [numInodo] ;

// Información extra de apoyo
struct {
    int posicion;
    int abierto;
} inodos_x [numInodo] ;
```

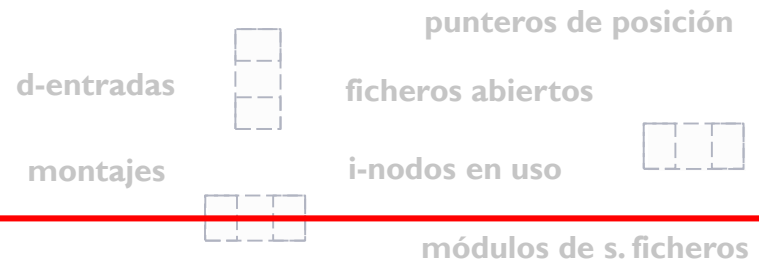
Ejercicio
solución c)

Llamadas al sistema de archivos

Descriptor	Uso de *namei*	Asig. i-n.	Atributos	E/S.	Sist. Arch.	Vista
open pipe creat close dup	open chown unlink creat chmod mknod chdir stat mount chroot link umount	creat mknod link unlink	chown chmod stat	read write lseek	mount umount	chdir chroot

Algoritmos de bajo nivel del sistema de archivos

namei		ialloc	alloc	
iget	iput	ifree	free	bmap



Algoritmos de gestión de bloq[illegible]hé

4. Diseñar las rutinas de gestión
   - Lectura y escritura en disco de estructuras en memoria

Propuesta de diseño de ialloc y alloc

```
int ialloc ( void )
{

    // buscar un i-nodo libre
    for (int=0; i<sbloques[0].numInodos; i++)
    {
        if (i_map[i] == 0) {
            // inodo ocupado ahora
            i_map[i] = 1;
            // valores por defecto en el i-nodo
            memset(&(inodos[i]), 0,
                    sizeof(TipoInodoDisco));
            // devolver identificador de i-nodo
            return i;
        }
    }

    return -1;
}
```

```
int alloc ( void )
{
    char b[BLOCK_SIZE];

    for (int=0; i<sbloques[0].numBloquesDatos; i++)
    {
        if (b_map[i] == 0) {
            // bloque ocupado ahora
            b_map[i] = 1;
            // valores por defecto en el bloque
            memset(b, 0, BLOCK_SIZE);
            bwrite(DISK, i+sbloques[0].primerInodo, b);
            // devolver identificador del bloque
            return i;
        }
    }
    return -1;
}
```

Propuesta de diseño de ifree y free

```
int ifree ( int inodo_id )
{
    // comprobar validez de inodo_id
    if (inodo_id > sbloques[0].numInodos)
        return -1;

    // liberar i-nodo
    i_map[inodo_id] = 0;

    return 1;
}
```

```
int free ( int block_id )
{
    // comprobar validez de block_id
    if (block_id > sbloques[0].numBloquesDatos)
        return -1;

    // liberar bloque
    b_map[block_id] = 0;

    return 1;
}
```

Propuesta de diseño de namei y bmap

```
int namei ( char *fname )
{
   // buscar i-nodo con nombre <fname>
   for (int=0; i<sbloques[0].numInodos; i++)
   {
        if (! strcmp(inodos[i].nombre, fname))
             return i;
   }

    return -1;
}
```

```
int bmap ( int inodo_id, int offset )
{
    // comprobar validez de inodo_id
    if (inodo_id > sbloques[0].numInodos)
        return -1;

    // bloque de datos asociado
    if (offset < BLOCK_SIZE)
        return inodos[inodo_id].bloqueDirecto;

    return -1;
}
```

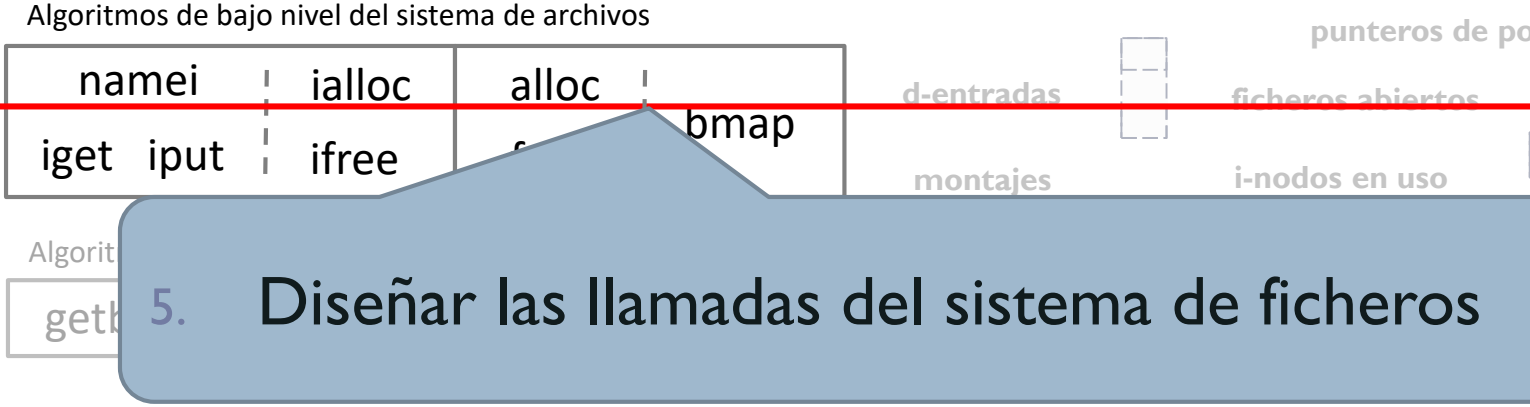
Ejercicio
solución d)

Llamadas al sistema de archivos

Descriptor	Uso de *namei*	Asig. i-n.	Atributos	E/S.	Sist. Arch.	Vista
open pipe creat close dup	open chown unlink creat chmod mknod chdir stat mount chroot link umount	creat mknod link unlink	chown chmod stat	read write lseek	mount umount	chdir chroot

Algoritmos de bajo nivel del sistema de archivos

namei	ialloc	alloc
iget iput	ifree	bmap



Propuesta de diseño de mount

```
int mount ( void )
{
    // leer bloque 1 de disco en sbloques[0]
    bread(DISK, 1, &(sbloques[0]) );

    // leer los bloques para el mapa de i-nodos
    for (int i=0; i<sbloques[0].numBloquesMapaInodos; i++)
          bread(DISK, 2+i, ((char *)i_map + i*BLOCK_SIZE) ;

    // leer los bloques para el mapa de bloques de datos
    for (int i=0; i<sbloques[0].numBloquesMapaDatos; i++)
         bread(DISK, 2+i+sbloques[0].numBloquesMapaInodos, ((char *)b_map + i*BLOCK_SIZE);

    // leer los i-nodos a memoria
    for (int i=0; i<(sbloques[0].numInodos*sizeof(TipoInodoDisco)/BLOCK_SIZE); i++)
         bread(DISK, i+sbloques[0].primerInodo, ((char *)inodos + i*BLOCK_SIZE);

    return 1;
}
```

Propuesta de diseño de sync

```
int sync ( void )
{
    // escribir bloque 1 de sbloques[0] a disco
    bwrite(DISK, 1, &(sbloques[0]) );

    // escribir los bloques para el mapa de i-nodos
    for (int i=0; i<sbloques[0].numBloquesMapaInodos; i++)
          bwrite(DISK, 2+i, ((char *)i_map + i*BLOCK_SIZE) ;

    // escribir los bloques para el mapa de bloques de datos
    for (int i=0; i<sbloques[0].numBloquesMapaDatos; i++)
         bwrite(DISK, 2+i+sbloques[0].numBloquesMapaInodos, ((char *)b_map + i*BLOCK_SIZE);

    // escribir los i-nodos a disco
    for (int i=0; i<(sbloques[0].numInodos*sizeof(TipoInodoDisco)/BLOCK_SIZE); i++)
         bwrite(DISK, i+sbloques[0].primerInodo, ((char *)inodos + i*BLOCK_SIZE);

    return 1;
}
```

Propuesta de diseño de umount

```
int umount ( void )
{
     // asegurarse de que todos los ficheros están cerrados
     for (int=0; i<sbloques[0].numInodos; i++) {
          if (inodos_x[i].abierto == 1) {
              return 0;
          }
    }

    // escribir a disco los metadatos
    sync();
    return 1;
}
```

Propuesta de diseño de mkfs

```
int mkfs ( void )
{
    // inicializar a los valores por defecto del superbloque, mapas e i-nodos
    sbloques[0].numMagico = 1234;
    sbloques[0].numInodos = 201;
    …
    for (int=0; i<sbloques[0].numInodos; i++)
          i_map[i] = 0; // free
    for (int=0; i<sbloques[0].numBloquesDatos; i++)
          b_map[i] = 0; // free
    for (int=0; i<sbloques[0].numInodos; i++)
         memset(&(inodos[i]), 0, sizeof(TipoInodoDisco) );

    // escribir los valores por defecto al disco
    sync();
    return 1;
}
```

Propuesta de diseño de open y close

```
int open ( char *nombre )
{
    int inodo_id ;

 // buscar el inodo asociado al nombre
    inodo_id = namei(nombre) ;
    if (inodo_id < 0)
        return inodo_id ;
// Controlo que no esté ya abierto
   if (inodos_x[inodo_id].abierto   == 1)
       return -1;


 // iniciar sesión de trabajo
    inodos_x[inodo_id].posicion = 0;
    inodos_x[inodo_id].abierto   = 1;

    return inodo_id;
}
```

```
int close ( int fd )
{

    // comprobar descriptor válido
    if ( (fd < 0) || (fd > sbloques[0].numInodos-1) )
        return -1 ;

    // cerrar sesión de trabajo
    inodos_x[fd].posicion = 0;
    inodos_x[fd].abierto   = 0;

    return 1;
}
```

Propuesta de diseño de creat y unlink

```
int creat ( char *nombre )
{
    int b_id, inodo_id ;

    inodo_id = ialloc() ;
    if (inodo_id < 0) { return inodo_id ; }
    b_id = alloc();
    if (b_id < 0) { ifree(inodo_id); return b_id ; }

    inodos[inodo_id].tipo = 1 ; // FICHERO
    strcpy(inodos[inodo_id].nombre, nombre);
    inodos[inodo_id].bloqueDirecto = b_id ;
    inodos_x[inodo_id].posicion = 0;
    inodos_x[inodo_id].abierto   = 1;

    return 1;
}
```

```
int unlink ( char * nombre )
{
    int inodo_id ;

    inodo_id = namei(nombre) ;
    if (inodo_id < 0)
        return inodo_id ;

    free(inodos[inodo_id].bloqueDirecto);
    memset(&(inodos[inodo_id]),
                0,
                sizeof(TipoInodoDisco));
    ifree(inodo_id) ;

   return 1;
}
```

Propuesta de diseño de read y write

```
int read ( int fd, char *buffer, int size )
{
   char b[BLOCK_SIZE] ;
   int b_id ;

   if (inodos_x[fd].posicion+size > inodos[fd].size)
      size = inodos[fd].size - inodos_x[fd].posicion;
   if (size =< 0)
      return 0;

   b_id = bmap(fd, inodos_x[fd].posicion);
   bread(DISK, b_id, b);
   memmove(buffer,
                b+inodos_x[fd].posicion,
                size);
   inodos_x[fd].posicion += size;

   return size;
}
```

```
int write ( int fd, char *buffer, int size )
{
   char b[BLOCK_SIZE] ;
   int b_id ;

   if (inodos_x[fd].posicion+size > BLOCK_SIZE)
      size = BLOCK_SIZE - inodos_x[fd].posicion;
   if (size =< 0)
      return 0;

   b_id = bmap(fd, inodos_x[fd].posicion);
   bread(DISK, b_id, b);
   memmove(b+inodos_x[fd].posicion,
                buffer, size);
   bwrite(DISK, b_id, b);
   inodos_x[fd].posicion += size;

   return size;
}
```

Ejercicio
solución

1. Planteamiento inicial
    1. Estado inicial del sistema
    2. Estudio de qué hay que modificar
2. Responder a las preguntas
3. Revisar las respuestas

Fallos a evitar

1) Contestar a la primera pregunta de un apartado únicamente (y no contestar al resto de preguntas/peticiones)
2) Contestar a otra pregunta de la pedida.
3) Respuestas largas:
    1) Quitan tiempo para realizar el resto del examen.
    2) Contestar más de lo pedido puede suponer fallos extra.
    3) Importante que las partes claves del ejercicio estén correctas.
4) Usar el planteamiento del problema como respuesta.

Ejercicios

Sistemas de ficheros

Grupo ARCOS

Diseño de Sistemas Operativos

Grado en Ingeniería Informática

Universidad Carlos III de Madrid