Grupo: ..... NIA: .....
Nombre y apellidos: .....

Ejercicio 1 (1,5 puntos)

TEST-A:

1	2	3	4	5	6	7	8	9	10
a	b	c	c	d	c	c	c	b	a

TEST-B:

1	2	3	4	5	6	7	8	9	10
d	d	b	c	b	c	c	a	c	c

Grupo: ..... NIA: .....
Nombre y apellidos: .....

Ejercicio 2 (3,5 puntos)

a) **Constantes, variables globales**:

- Constantes globales:
  - **#define SCHED_RR 100**
  - **#define SCHED_FIFO 200**
  - **#define LET_SELECTION_FROM_KEYBOARD 232323**
- Variable global:
  - **int krn_is_allow = 0**
  - **int krn_sch_used = SCHED_RR**

A las variables globales se les da un valor por defecto, por lo que no sería preciso modificar la llamada de arranque del sistema operativo para tal fin.

Llamada al sistema nueva:

Llamada al sistema (en espacio de usuario) let_selection_from_keyboard (int is_allow)

- **R0 = LET_SELECTION_FROM_KEYBOARD**
- **R1 = is_allow**
- **TRAP**
- return R0

Llamada al sistema (en kernel) let_selection_from_keyboard ( )

- R0 = 0
- **SI (R1 < 0)**
  - return
- **krn_is_allow = R1**
- **SI (krn_is_allow == 0)**
  - **krn_sch_used = SCHED_RR**
- return

Grupo: ..... NIA: .....
Nombre y apellidos: .....

b) Gestión resultante del reloj:

Funcionalidad	**Descripción**
Int. Hardware	• Ticks = Ticks + 1 • <Fragmento 4>
Int. Software	• **SI krn_sch_used == SCHED_FIFO** o **return** • procesoActual->rodaja = procesoActual->rodaja - 1 • SI (procesoActual->rodaja == 0) o InsertarAlFinal (procesoActual, listaProcesosListos) o procesoActual->estado = LISTO o o <Fragmento 2>

Grupo: ..... NIA: .....
Nombre y apellidos: .....

c) Gestión resultante del teclado:

Funcionalidad	Descripción

Int. Hardware	• Insertar_tecla(tecla, Teclado.BufferTeclas)
	• **SI krn_is_allow > 0**
	  o **SI tecla == 'r'**
	  ▪ **krn_sch_used = SCHED_RR**
	  o **SI tecla == 'f'**
	  ▪ **krn_sch_used = SCHED_FIFO**
	• <Fragmento 3>

Int. Software	• <Fragmento 1>

read(fd,buf,size)	• MIENTRAS estaVacio(Teclado.BufferTeclas)
	  o Insertar_proceso_actual(Teclado.Bloqueados)
	  o procesoActual->estado = BLOQUEADO
	  o
	  o <Fragmento 2>
	• Buffer[0] = extraerTecla(Teclado.BufferTeclas)
	• return 1

Grupo: ..... NIA: .....
Nombre y apellidos: .....

Ejercicio 3 (2,5 puntos)

a) Metadatos persistentes en disco:

Superbloque

- **int Número mágico = 0x22223**
- **int Tamaño de bloque = 1024 // R1**
- **int Número de bloques = 800 // R2**
- **int Tamaño del inodo = 20 // R6**
- **int Número de inodos = 50 // R3**
- **char bloques_estado[800] = {'1','1','0','0',...} // R5**
- **char inodos_estado[50] = {'0','0',...} // R5**
- **char relleno[1024-20-800-50]**

I-nodo

- **int bloque_ind // R7**
- **char tipo = REGULAR // R7**
- **int tamaño = 0**
- **char relleno[20-4-1-4]**

char nombres[50][20]
int bloque_indirecto[1024/4]

b) Metadatos en memoria:

Superbloque sb
Inodo inodos[50]
char nombres[50][20] // R8
Sesion sesiones[50]

Donde Sesión es una estructura con, al menos, los siguientes campos:

- **Int Posición // R9**
- **char Abierto // R9**
- **Int PID // R9**

Grupo: ..... NIA: .....
Nombre y apellidos: .....

c) bmap:

```
int bmap ( int inodo_id, int offset )
{
    int bloque[1024 / 4] ;
    int bid ;

    if (offset > inodos[inodo_id].tamaño)
        return -1;

    bid = offset / sb.tamaño_de_bloque;
    if (bid > (1024/4))
        return -1;

    bread(inodos[inodo_id].bloque_ind, bloque);
    return bloque[bid];
}
```

Grupo: ..... NIA: .....
Nombre y apellidos: .....

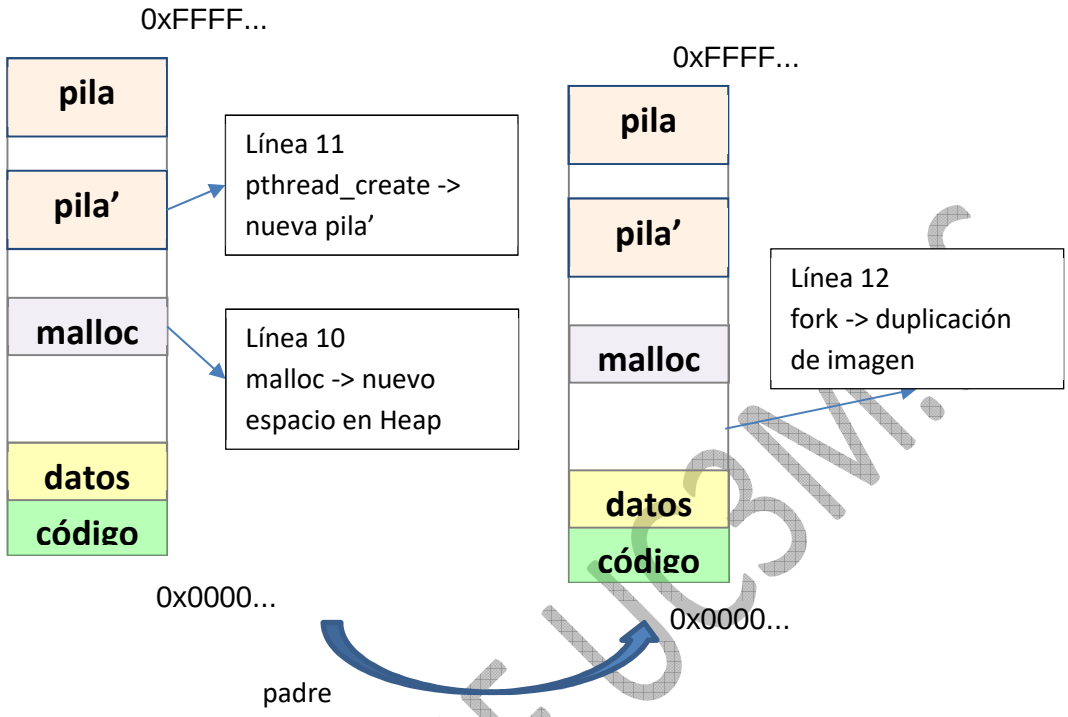
d) ialloc, ifree y namei:

```
int ialloc ( void )
{
   for (int=0; i< sb.numero_de_inodos; i++)        // para cada inodo…
   {
       if (sb.inodos_estado[i] == '0') {            // si inodo libre
          memset(&(inodos[i]), 0, sizeof(inodo));   // poner valores por defecto
          sb.inodos_estado[i] = '1';                // ocupar inodo
          return i;                                 // devolver identificador
       }
   }
   return -1;                                       // si no hay inodo libre, devolver -1
}


void ifree ( int inodo_id )
{
   sb.inodos_estado[inodo_id] = '0';                // liberar i-nodo
}


int namei ( char *fname )
{
    for (int=0; i< sb.numero_de_inodos; i++)        // buscar inodo con nombre <fname>
    {
        if (! strcmp(nombres[i], fname))
            return i;                               // devolver identificador
    }
    return -1;                                      // si no está el inodo, devolver -1
}
```

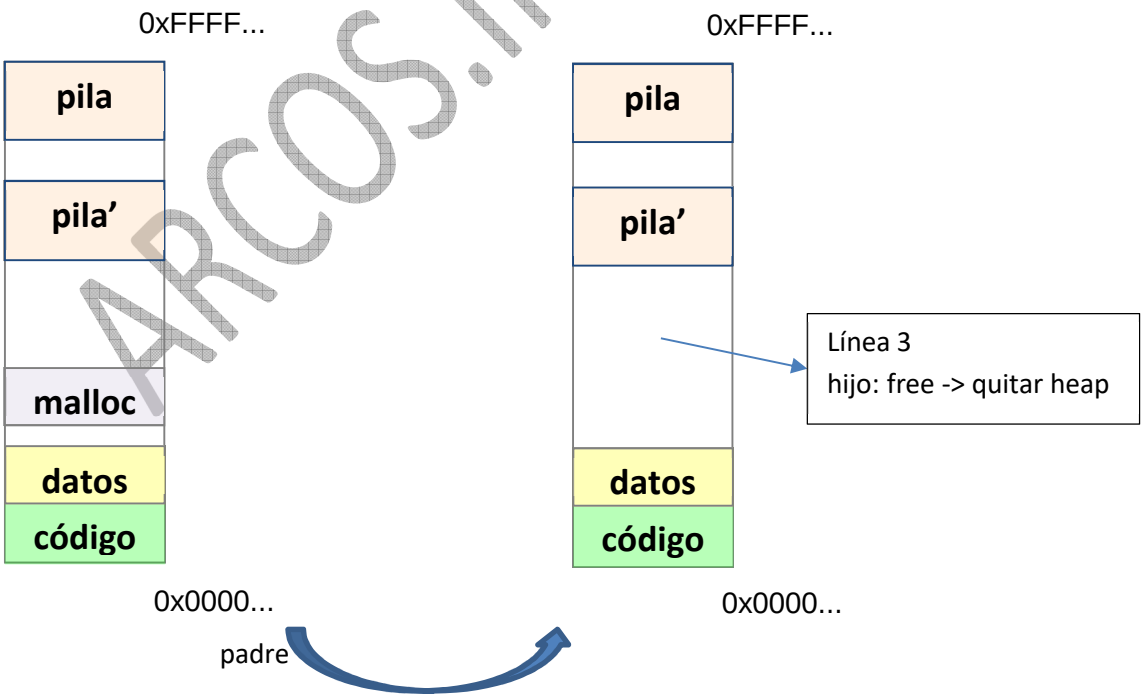
Grupo: ..... NIA: .....
Nombre y apellidos: .....

Ejercicio 4 (2,5 puntos)

a) A -> B

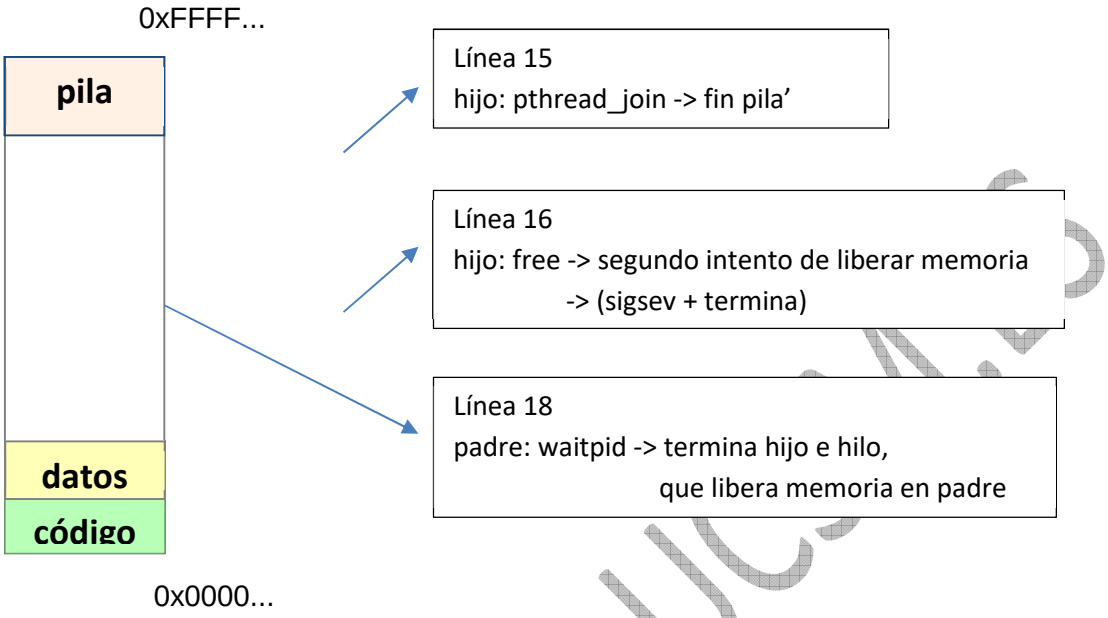


b) B->C



Grupo: ..... NIA: .....
Nombre y apellidos: .....

c) C->D



d) Si, en el proceso hijo se libera la memoria dos veces,
y en el proceso padre no se libera la memoria dinámica pedida.
Tampoco se usa pthread_join en el padre, por lo que los recursos del hilo no son liberados.