Rutinas Dependientes

Impresora

1. Estructuras de datos para la gestión de la impresora

1.- La impresora dispone de un descriptor de dispositivo (DD_Impr) que es creado al inicializar el sistema. Este DD, al igual que el del resto de dispositivos de Onion, contiene una parte común y otra no común.

Descriptor de Dispositivo Impressora:

```
/* parte común */
                        char nom[12];
                        int caract;
                        int opens;          /* herencias */
                        int sem_mutex;
                        int sem_gestor;
                        int (*p_obrir) ();
                        int (*p_tancar) ();
                        int (*p_llegir) ();
                        int (*p_escriure) ();
                        int (*p_posicionar) ();
                            . . .

/* parte específica impresora */
                        struct cua *q_io_fin;
                        struct cua *q_iorb;
                        struct cua *q_pid;        /* cola pid pendiente asignar en modo Nospool*/
                        struct cua *q_fitx_temp;        /* id_fitxer para spooler*/
                        int propietari;
                        char prot;
                        int proc;           /*pid primer proceso que abre en modo Nospool*/
```

A diferencia de los ficheros, el control de las protecciones se realiza a nivel sistema. Por ello disponemos del campo *prot*, que indica los permisos de lectura, escritura y ejecución para el dueño (*propietari*) y para el resto.
El campo *propietari* nos indica el *uid* del proceso que tiene asignada la impresora. Si ningún proceso la tiene abierta, ó trabaja en modo Spool, el campo propietari tiene por valor -1.
El campo *proc* nos indica el *pid* del proceso que tiene asignada la impresora. Si ningún proceso la tiene abierta, ó trabaja en modo Spool, el campo proc tiene por valor -1.
La cola *q_fitx_temp* contiene los identificadores de los ficheros temporales que están listos para ser impresos por el *spooler* (proceso de sistema). Sólo se utiliza cuando la impresora trabaja en modalidad Spool.

La cola *q_pid* contiene los identificadores de proceso que tienen abierta la impresora en modo Nospool pero no pueden escribir en ella (sólo uno puede escribir). Sólo se utiliza cuando la impresora trabaja en modalidad Nospool.

2.- Los elementos que se encolan en la cola *q_fitx_temp* tienen dos campos, en uno se especifica el identificador del fichero a imprimir, y en el otro la modalidad en la que va a pasar a trabajar la impresora. Si es Nospool, el spooler se bloquea y pasa a trabajar el gestor de la impresora. Ver esquema de cambio de modo.

```
struct peticion_spooler
  {
      struct item elem;
      int id_fitx;
      int caract;
  }
```

3.- Como diseñadores del sistema hemos tratado de simplificar el tratamiento de la impresora y por ello consideramos que el proceso sólo puede trabajar con ella por un único canal, tanto en modo Spool como en Nospool (es transparente al usuario). Para no recorrer los canales cada vez que llamamos a *obrir_imp*, podemos tener un campo en el pcb que nos indique si ya estamos trabajando con ella.

```
struct pcbsys
  {
      ...
      int impr;
  }
```

4.- Para no tener problemas con la herencia de canales en el momento de crear procesos nuevos, vamos a permitir que trabajando en modo Nospool varios procesos puedan tener abierto el dispositivo impresora (en Spool es obvio). Sólo uno, el primer proceso que la abrió, puede escribir en ella. Al resto, cuando intenten escribir se les envía un mensaje de error. Cuando el proceso que la tiene asignada, cierre el canal, la rutina de *tancar_impr* elige el primero de la cola *q_pid* como nuevo proceso dueño de la impresora.

```
struct proc_impr
  {
      struct item elem;
      int idp;
      int idu;
  }
```

2. Modos de trabajo.

La impresora tiene dos modos de trabajo: Nospool y Spool.

- Modo Nospool

Si trabajamos en modo Nospool, la impresora sólo puede ser utilizada por un proceso. Además este proceso sólo la puede abrir por un canal.
El resto de procesos la pueden abrir (para evitar problemas con las herencias), pero si intentan escribir, se les envía un mensaje de error.
Por tanto, todos los iorb's que le llegan al gestor de la impresora pertenecen al mismo proceso. El gestor realiza la operación de e/s haciendo llamadas a la rutina de bfs.
Cuando el proceso cierre el canal asociado a la impresora, dejará libre el dispositivo para que pueda ser utilizado por otro proceso. Ver tancar_impresora.

```
escriure_impr (eldd, buff, lon)
 struct dd * eldd;
 char * buff;
 long * lon;
 {
    if (eldd.proc==quisoc())
       { obtener id_io;
         obtener y rellenar iorb;
         encuar(eldd.q_iorb, iorb);
         signal (sem_gestor_impr);
         return (id_io);
       }
    else return(error);
}
```

Vuelve después de encolar la petición. El funcionamento es **asíncrono**.

```
gestor_impr()
{
     for (;;)
     {
          wait (sem_gestor_impr);
          extraer iorb;
          ...
          while (i<lon)
          {
               escriure_impressora_bfs(car,mod);
          }

          liberar_iorb;
          encuar io_fin;
          signal (id_io);
     }
}
```

- Modo Spool

Si trabajamos en modo Spool, la impresora puede ser compartida por n procesos. En realidad estos procesos no escriben directamente sobre la impresora sino que lo hacen sobre un dispositivo intermedio (normalmente el disco). Es decir, los procesos cada vez que abren la impresora, están abriendo un nuevo fichero temporal . Ésto es totalmente transparente al usuario. Los procesos sólo pueden abrir la impresora desde un canal (como en el caso Nospool).

Las escrituras sobre el canal se realizan sobre el fichero temporal asociado a dicho canal, por tanto, los iorb's se envían al gestor(es) de ficheros, tal y como vimos en las llamadas dependientes de fichero.

Cuando el proceso cierre el canal asociado a la impresora, lo que hace en realidad es cerrar el canal asociado al fichero temporal. Esta rutina es diferente a la de *tancar_fitxer* correspondiente a ficheros

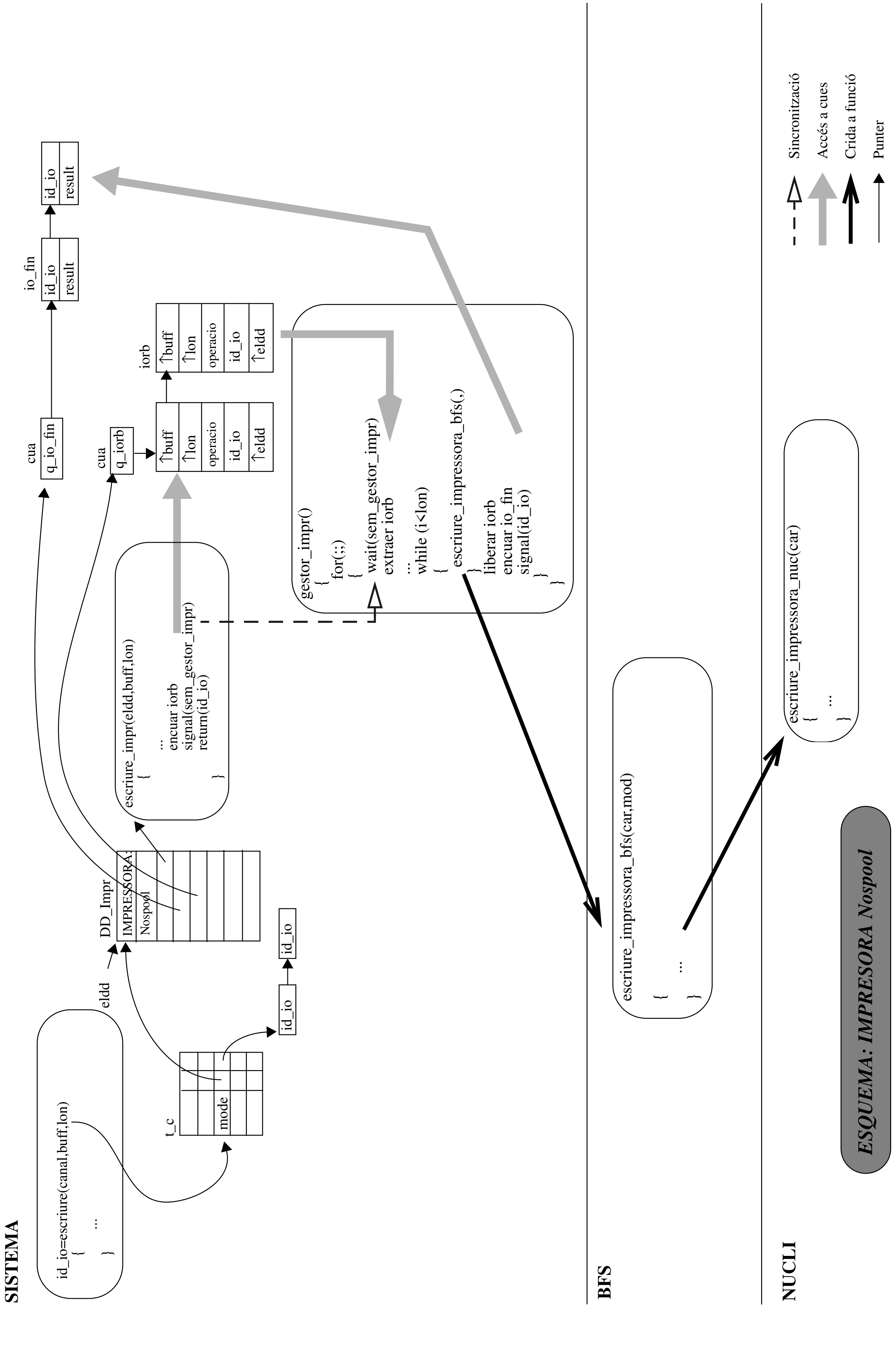
.

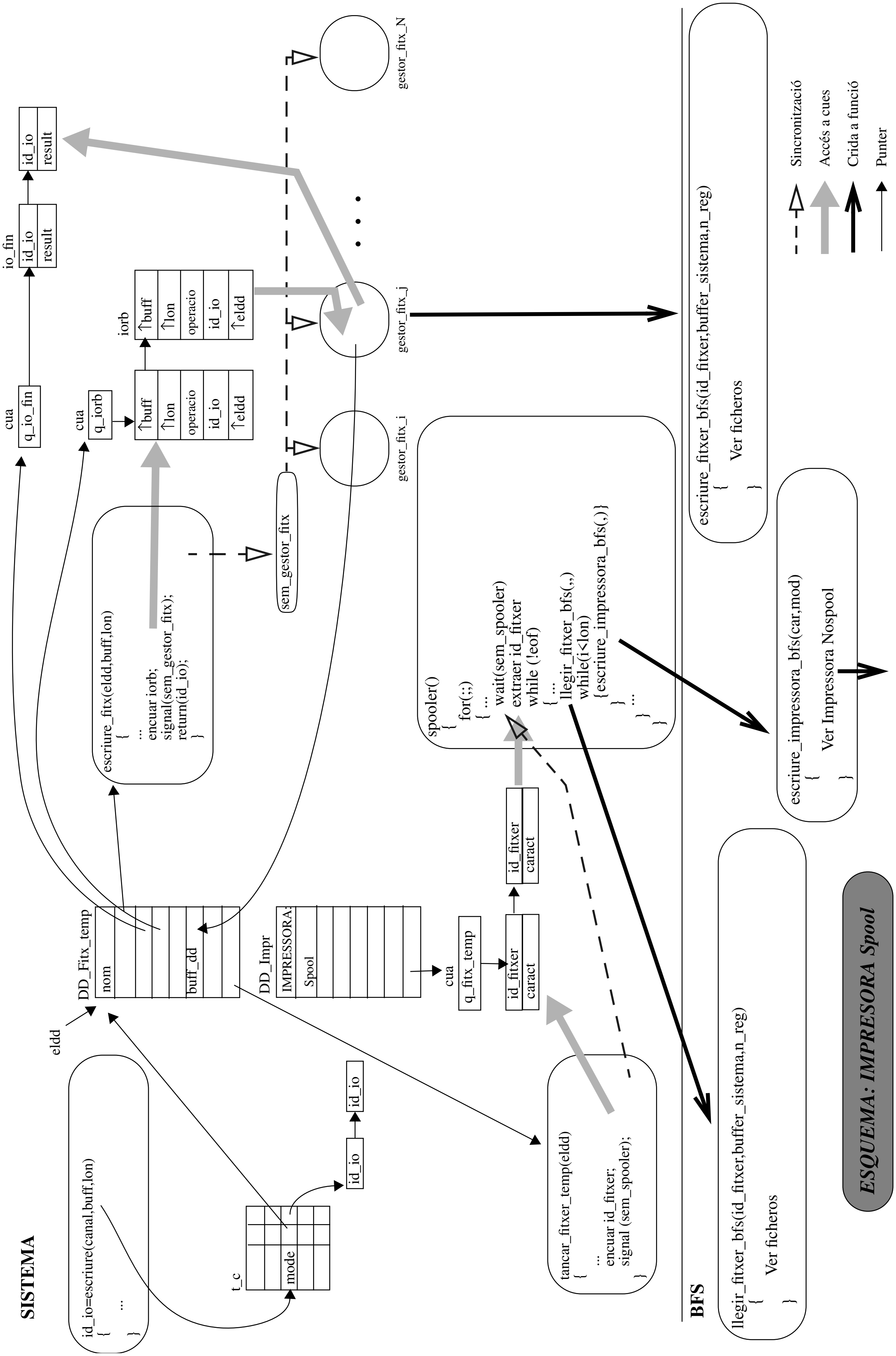
```
tancar_fitxer_temp (eldd)
struct dd * eldd;
  {
      obtener y rellenar peticion
           {peticion.id_fitx=id_fitxer
            peticion.caract=impressora->caract;
           }
      encuar(impressora.q_fitx_temp, peticion);
      liberar DD_Fitx_temp;
      signal (sem_spooler);
      buscar_pcb(quisoc())->impr=FALSO;
  }
```

```
spooler()
  {
    for (;;)
      { ...
        wait (sem_spooler);
        extraer peticion.id_fitx;

        while (!eof)
          { ...
             llegir_fitxer_bfs(id_fitx,buff,lon)
             while (i<lon)
                 { ...
                 escriure_impressora_bfs(car,mod)
                 }
          }
        ...

      }
  }
```


SISTEMA
BFS
NUCLI
id_io=escriure(canal,buff,lon)
{
...
}
t_c
mode
eldd
id_io
id_io
DD_Impr
IMPRESSORA:
Nospool
escriure_impr(eldd,buff,lon)
{
...
encuar iorb
signal(sem_gestor_impr)
return(id_io)
}
cua
q_io_fin
io_fin
id_io
result
id_io
result
cua
q_iorb
↑buff
↑lon
operacio
id_io
↑eldd
iorb
↑buff
↑lon
operacio
id_io
↑eldd
gestor_impr()
{
for(;;)
{
wait(sem_gestor_impr)
extraer iorb
...
while (i<lon)
{
escriure_impressora_bfs(,)
}
liberar iorb
encuar io_fin
signal(id_io)
}
}
escriure_impressora_bfs(car,mod)
{
...
}
escriure_impressora_nuc(car)
{
...
}
Sincronització
Accés a cues
Crida a funció
Punter
ESQUEMA: IMPRESORA Nospool


SISTEMA
BFS
ESQUEMA: IMPRESORA Spool
Sincronització
Accés a cues
Crida a funció
Punter
id_io=escriure(canal,buff,lon)
{
...
}
t_c
mode
id_io
id_io
eldd
DD_Fitx_temp
nom
buff_dd
DD_Impr
IMPRESSORA:
Spool
cua
q_fitx_temp
id_fitxer
caract
id_fitxer
caract
tancar_fitxer_temp(eldd)
{
...
encuar id_fitxer;
signal (sem_spooler);
}
escriure_fitx(eldd,buff,lon)
{
...
encuar iorb;
signal(sem_gestor_fitx);
return(id_io);
}
sem_gestor_fitx
cua
q_io_fin
cua
q_iorb
↑buff
↑lon
operacio
id_io
↑eldd
iorb
↑buff
↑lon
operacio
id_io
↑eldd
io_fin
id_io
result
id_io
result
gestor_fitx_i
gestor_fitx_j
gestor_fitx_N
spooler()
{
for(;;)
{ ...
wait(sem_spooler)
extraer id_fitxer
while (!eof)
{ ...
llegir_fitxer_bfs(,,)
while(i<lon)
{escriure_impressora_bfs(,)}
} ...
}
}
llegir_fitxer_bfs(id_fitxer,buffer_sistema,n_reg)
{
Ver ficheros
}
escriure_impressora_bfs(car,mod)
{
Ver Impressora Nospool
}
escriure_fitxer_bfs(id_fitxer,buffer_sistema,n_reg)
{
Ver ficheros
}

3. Abrir Impresora.

A los procesos sólo se les permite trabajar con la impresora por un canal.
Si la característica es Spool, pasamos a abrir un fichero temporal sobre el que realizaremos las escrituras. Suponemos que en este modo, la impresora no tiene dueño. La impresora es un dispositivo no compartible pero que nos permite, mediante el modo Spool, simular su compartición.
Si la característica es Nospool, y la impresora no ha sido abierta por ningún proceso, el proceso que la abre pasa a ser el dueño. Los siguientes procesos que quieran trabajar con la impresora, la pueden abrir (evitar problemas con herencias), pero cuando intenten escribir se les envía un mensaje de error (*escriure_impr*). Cuando el proceso que la tiene asignada, cierre la impresora, la cede al primero de la cola.

```
obrir_impr (eldd, modo)
struct dd * eldd;
int modo;
{
    ...
    aux = buscar_pcb (quisoc());

    if (aux->impr == TRUE)
            { return ("el proceso tiene abierta la impresora") }
      else { if (eldd->caract == SPOOL)
                  { obrir_fitxer_temp (eldd) }
                  /* Nospool sin propietario*/
             else { if (propietari == -1)
                        {   eldd->propietari = aux->uid;
                            eldd->proc = quisoc ();
                            eldd->prot = --- -wx;
                        };
                        /*Nospool y asignada a otro proceso */
                  else  {  proc_impr = obtener_struct_proc_impr();
                            proc_impr -> idp = quisoc ();
                            proc_impr -> idu = aux->uid;
                            encuar (eldd -> q_pid, proc_impr);
                         };
                  };
             aux->impr = TRUE;
            }
}
```

4. Cerrar Impresora.

En modo Spool, cerrar impresora equivale a encolar el fichero temporal en la cola de peticiones al spooler, tal y como vimos al estudiar el modo spool (*tancar_fitxer_temp (eldd)*). Esta rutina es apuntada desde el DD_Fitx_temp y por tanto no afecta directamente al dispositivo impresora.
Por este motivo, al hablar de cerrar impresora nos referimos únicamente al caso Nospool. La rutina dependiente es *tancar_impr (eldd)*. Esta rutina comprueba la característica de la impresora (por si se hubiera modificado).
En el caso de que sea Nospool y existan procesos en la cola *q_pid*, extrae el primero de ellos y actualiza los campos uid y proc del DD_Impr. Si no hay procesos, libera el DD_Impr (propietari = -1, proc = -1, prot = -wx -wx)
En el caso de que sea Spool (se ha modificado), obtiene un iorb, tal y como lo hace la rutina dependiente *escriure_impr* pero únicamente rellena el campo *operacio = spool*. Lo encola, y hace el signal al gestor de la impresora. Éste se encarga de desbloquear al spooler (*signal (oper_spool)*) y bloquearse a la espera de que se vuelva a modificar la característica. Cuando sea Nospool, el spooler lo desbloquea mediante un (*signal (oper_nospool)*). Ver esquema de cambio de modo.

```
tancar_impr (eldd)
struct dd * eldd;
{   ...
    if (eldd->caract == Nospool)
            {  if (buida (q_pid))
                   {  eldd -> propietari = -1;
                      eldd -> proc = -1;
                      eldd -> prot = -wx -wx;
                   };
            else  {  proc_impr = primer (eldd ->q_pid);
                      eldd -> propietari = proc_impr -> idu;
                      eldd -> proc = proc_impr -> idp;
                      liberar proc_impr ();
                   };
            };
     else   {  iorb = obtener iorb ();
               iorb -> operacio = spool;
               encuar (eldd -> q_iorb, iorb);
               signal (sem_gestor_impr);
            };
     aux->impr == FALSE;
};
```

5. Cambio de modo.

Trabajando en modo Nospool, sólo un proceso puede escribir en la impresora. Si antes de que la cierre modificamos su característica, los procesos que quieran abrir la impresora a partir de ese momento la encontrarán en modo Spool y por tanto escribirán en un fichero temporal. Para dar mayor facilidad a los procesos de usuario permitimos que mientras uno trabaja en modo Nospool, el resto, si se ha modificado la característica, pueda hacerlo en modo Spool.

Para que no se interfieran el gestor y el spooler, utilizaremos semáforos de sincronización, ya que los dos escriben por impresora. Cuando el proceso que tiene asignada la impresora la libere, el gestor de impresora desbloquea al spooler y luego se bloquea.

En modo Spool permitimos que, si se ha modificado la característica, uno pueda hacerlo en modo Nospool. En este caso, el spooler es el que comprueba en cada petición si ha de pasar el control al gestor de la impresora.

```
tancar_fitxer_temp (eldd)
struct dd * eldd;
  {
      ...
      encuar(impressora.q_fitx_temp, peticion);;
      signal (sem_spooler);
  }
```

```
escriure_impr (eldd, buff, lon)
 struct dd * eldd;
 char * buff;
 long * lon;
 { ...
     encuar(eldd.q_iorb, iorb);
     signal (sem_gestor_impr);
 }
```

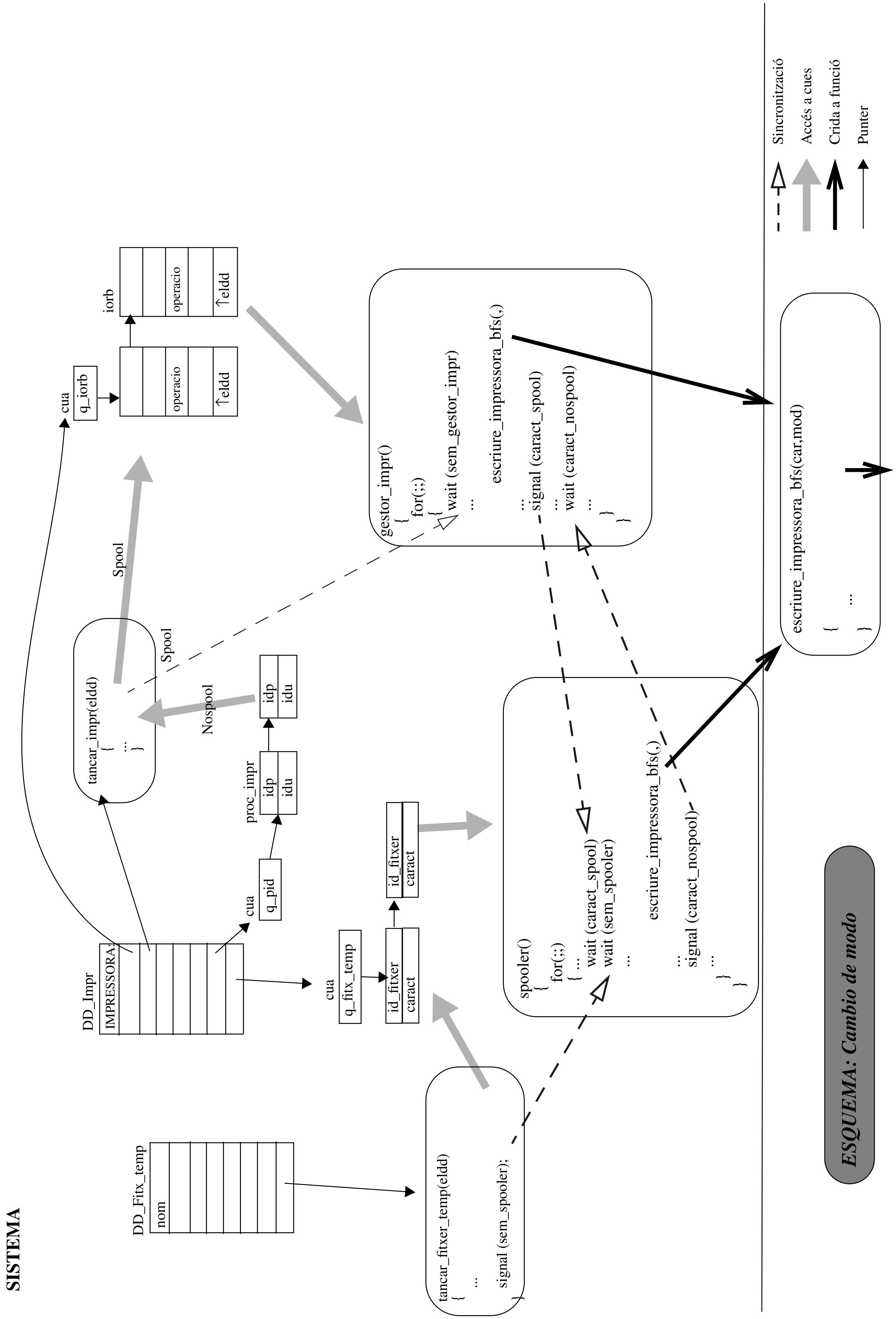
```
gestor_impr()
{
      for (;;)
      {
            wait (sem_gestor_impr);
            extraer iorb;
            switch (iorb -> operacion)
            {     case escriure:
                        {while (i<lon)
                              {escriure_impressora_bfs(car,mod); }
                        liberar_iorb;
                        encuar io_fin;
                        signal (id_io);
                        }
                  case spool:
                        {signal (caract_spool);
                         liberar_iorb;
                         wait (caract_nospool);
                        }
                  case set_dev: ...
            }
            ...
      }
}
```

```
spooler()
   {
     for (;;)
        { wait (caract_spool);
          wait (sem_spooler);
          extraer peticion.id_fitx;

          while (!eof)
            { ...
               llegir_fitxer_bfs(id_fitx,buff,lon)
               while (i<lon)
                    { ...
                     escriure_impressora_bfs(car,mod)
                    }
            }
          if (peticion.caract == nospool)
               { signal (caract_nospool); }
           else { signal (caract_spool); }
        }
   }
```

petición pendiente

sincronización


SISTEMA
DD_Fitx_temp
nom
tancar_fitxer_temp(eldd)
{
...
signal (sem_spooler);
}
DD_Impr
IMPRESSORA:
cua
q_fitx_temp
id_fitxer
caract
id_fitxer
caract
cua
q_pid
proc_impr
idp
idu
idp
idu
Nospool
tancar_impr(eldd)
{
...
}
Spool
Spool
cua
q_iorb
operacio
↑eldd
iorb
operacio
↑eldd
spooler()
{
for(;;)
{ ...
wait (caract_spool)
wait (sem_spooler)
...
escriure_impressora_bfs(,)
...
signal (caract_nospool)
...
}
}
gestor_impr()
{
for(;;)
{
wait (sem_gestor_impr)
...
escriure_impressora_bfs(,)
...
signal (caract_spool)
...
wait (caract_nospool)
...
}
}
escriure_impressora_bfs(car,mod)
{
...
}
Sincronització
Accés a cues
Crida a funció
Punter
ESQUEMA: Cambio de modo