Rutines Independents

1. Obrir

Obrir independent del dispositiu

- Comprovar que el canal és correcte
  - Número de canal
  - Canal no està ja assignat
- Buscar DD associat a nom, o bé crear-lo si és fitxer (o segons la implementació MBX si ¬∃)
- Cridar a obrir dependent del dispositiu, la qual haurà de:
  - Incrementar nombre d'herències al DD
  - Si és el primer en obrir el dispositiu, marcar propietari
- Si tot ha estat correcte:
  - Actualitzar entrada de la taula de canals

Observacions

- Síncrona
- L'accés a cues i variables globals s'ha de fer en **Exclussió Mútua a totes les rutines**

```
int obrir (canal, nom, mode, prot)
int canal, mode;
char * nom;
char prot;
{
 struct dd * eldd;
 struct pcbsys * aux;
 int error, ret;

 aux = obtenir_pcb(quisoc());
 if (canal ∉ 0..4) return(-9);
 if (aux->t_c[canal].pdd != NIL) return(-7);

 /* No hi han errors */
  if ((eldd = buscar_DD(nom)) == NIL) return(-1);

  /* Crida a la rutina dependent */
  ret = (* eldd->obrir) (eldd, mode, prot);
  if (ret == 0) {
     aux->t_c[canal].pdd = eldd;
     aux->t_c[canal].mode = mode;
     inicialitzar_cua(aux->t_c[canal].q_io);
     eldd->opens++;
  }
  return(ret);
}

struct dd * buscar_DD(nom)
char * nom;
{
  struct dd * eldd

  switch (tipus_dd(nom)) {
    case CONSOLA:      return (DD_CONSOLA);

    case TERMINAL:     return (DD_TTY);

    case IMPRESSORA:   return (DD_IMPRESSORA);

    case NUL:          return (DD_NUL);

    case MBX**:
         if ((eldd=buscar_cua(q_DD_MBX, nom)) == NIL) {
                eldd = copiar(MBXplantilla,nom);
                encuar(q_DD_MBX, eldd);
         }
         return (eldd);

    default:     if (nom_incorrecte(nom)) return(NIL);
                 eldd = copiar(FITXplantilla,nom);
                 return (eldd);
} }
```

2. Llegir

Llegir independent del dispositiu

- Comprovacions
  - Canal és correcte
    - Número de canal
    - Canal està ja assignat a un dispositiu o fitxer
  - Permís d'operació (mode L o L/E): A l'obrir el dispositiu incloem el tipus d'operació que demanem
  - No hem arribat al nombre màxim d'E/S pendents
- Per ser una crida **asíncrona**:
  - Cridar a la rutina dependent, la qual retornarà el id_io corresponent a aquesta E/S
  - Encuar id_io a la cua d'io pendents de l'entrada de la taula de canals corresponent: Cal aquesta informació per què abans de tancar un canal caldrà esperar per totes les E/S pendents del canal
  - Incrementar el nombre d'E/S pendents
  - Retornar id_io:
    Permetrà esperar o cancel.lar aquesta operació

Per **escriure** seria similar

```
int llegir (canal, pbuffer, lon)
int canal;
char * pbuffer;
long * lon;
{
 struct dd * eldd;
 struct pcbsys * aux;
 int id_io;
 struct io * p_io;

 aux = obtenir_pcb(quisoc());

 if (canal ∉ 0..4) return(-2);
 if ((aux->t_c[canal].pdd == NIL) return (-1);
 if (aux->t_c[canal].mode > 1)) return (-4);
          /* Modes: L = 0, L/E = 1, E = 2 */
 if (aux->n_es == MAX_ES) return(-3);

 /* No hi han errors */

  eldd = aux->t_c[canal].pdd;
  id_io = (* eldd -> llegir) (eldd, pbuffer, lon);
  if (id_io ≥ 0) {
   p_io = assignar_mem(quisoc(), sizeof(struct io));
   if (p_io == (char *)0) return(-5);

   p_io->id_io = id_io;
   encuar (&(aux->t_c[canal].q_io), p_io);
   aux->n_es++;
  }
  return(id_io);
 }
}
```

3. Posicionar

Independent del dispositiu

- Comprovar que el canal és correcte
  - Número de canal
  - Canal està ja assignat
- Cridar a la rutina dependent del dispositiu
- Retornar resultat

Observacions

- Només permés en fitxers
- Síncrona
- Esperar per totes les E/S pendents d'aquest canal (ho farà la rutina dependent)

```
int posicionar (canal, posicio_ptr)
int canal;
long * posicio_ptr;
{
  struct dd * eldd;
  struct pcbsys * aux;
  int error;

  aux = obtenir_pcb(quisoc());

  if (canal ∉ 0..4) return (-2);
  if (aux->t_c[canal].pdd == NIL) return(-1);

       /* No hi han errors */

  eldd = aux->t_c[canal].pdd;
  error = (* eldd -> posicionar) (eldd, posicio_ptr);
  return(error);
 }
}
```

4. Esperar

- Comprovar que ∃ id_io
  - Buscar l'item corresponent a id_io a les cues q_io de tots els canals: si el trobem coneixem el canal i el DD associat
  - Extreure'l
- Síncrona: wait (id_io)
- Obtenir_resultat:
  - Buscar l'item corresponent a id_io a la cua q_io_fin del DD
  - Extreure'l
- Decrementar el nombre d'E/S pendents del procés (al pcbsys)
- Retornar el resultat

Observacions

- Alliberar
  - estructura de l'item extret de la cua q_io
  - estructura de l'item extret de la cua q_io_fin
  - id_io (semàfor associat de finalització)
- Només a nivell sistema
- Només necessitem part independent

```
int esperar (id_io)
int id_io;

     /* Bloqueja fins que acabi l'E/S anb identificador id_io
     * Retorna:
     *  >= 0 -> correcte
     *     -1 -> id_io inexistent
     *     -2 -> EOF (El gestor s'encarrega de deixar-ho al
     *                resultat
     */

{
  struct dd * eldd;
  struct pcbsys * aux;
  int canal, res;

  aux = obtenir_pcb(quisoc());
  canal = obtenir_canal (aux, id_io);
  if (canal < 0) return (-1);     /* id_io inexistent */

  wait (id_io); /* Cada E/S pendent té associat un semàfor amb
                   número = id_io que està inicialitzat a 0 */
  eldd = aux->t_c[canal].pdd;
  res = obtenir_resultat (eldd->q_io_fin, id_io);
  aux->n_es--;
  return(res);
}
```

- obtenir_canal (aux, id_io)

  Busca a la t_c del proces l'id_io.
  Si el troba:
  - El treu de la cua i allibera l'estructura
  - retorna el canal on es trobaba encuat

  si no:
  - Retorna -1

- obtenir_resultat(cua, id_io)

  Se li passa la cua d'io_finalitzades del DD.
  Extreu l'element de la cua amb identificador == id_io
  Allibera l'estructura de l'element extret
  Allibera el semàfor associat de finalització id_io
  Retorna el camp resultat

5. Cancelar

- Comprovar que ∃ id_io
  - Buscar l'item corresponent a id_io a les cues q_io de tots els canals: si el trobem, coneixem el canal i el DD associat
- Buscar l'item corresponent a id_io a la cua q_iorb del DD. Si s'hi troba, vol dir que encara no ha estat atesa la petició. Extreure'l
- Buscar l'item corresponent a id_io a la cua q_io_fin del DD. S'hi trobara sii ja ha estat atesa la petició. Extreure'l
- Decrementar el nombre d'E/S pendents del procés (al pcbsys)
- Retornar el resultat

Observacions

- Només a nivell sistema
- Només necessitem part independent
- Si no està ni a la cua d'IORBs ni a la cua d'io_fin l'està servint el gestor
- Alliberar
  - estructura de l'item extret de la cua q_io
  - estructura de l'item extret de la cua q_iorb o q_io_fin
  - id_io (semàfor associat de finalització)

```
int cancel_lar (id_io)
int id_io;
/* Cancel.la l'operacio d'E/S amb identificador id_io
 * Retorna:
 *  0 -> correcte
 * -1 -> id_io inexistent
 * -2 -> id_io finalitzada
 */
{
 struct dd * eldd;
 struct pcbsys * aux;
 int canal;

 aux = obtenir_pcb(quisoc());
 canal = quin_canal (aux, id_io);
 if (canal < 0) return(-1);       /* id_io inexistent */

 eldd = aux->t_c[canal].pdd;
 if ((buscar_id_io (eldd->q_iorb, id_io) == 0) {
  aux->n_es--;
  buscar_id_io (&(aux->t_c[canal].q_io), id_io);
  alliberar(id_io);
  return(0);
 }
 if ((buscar_id_io (eldd->q_io_fin, id_io) == 0) {
  aux->n_es--;
  buscar_id_io (&(aux->t_c[canal].q_io), id_io);
  alliberar(id_io);
  return(-2);
 }
 /* Si no: La petició està sent atesa:
     - Valor de retorn com ja finalitzada
     - Deixar id_io a la cua de pendents per a que
       el tancar lliberi el resultat */
 return(-2);
}
```

- quin_canal (aux, id_io)

  Busca a la t_c del procés l'id_io.
  Si el troba:
  - retorna el canal on es troba encuat (No desencua)
  si no:
  - Retorna -1

- buscar_id_io (cua, id_io)

  Busca l'id_io a la cua especificada.
  Si el troba:
  - El desencua i allibera l'estructura
  - Retorna 0
  si no:
  - Retorna -1

6. Tancar

- Comprovar que el canal és correcte
  - Número de canal
  - Canal està ja assignat

- Recorre la q_io del canal que es tanca i per a cada petició pendent extreu l'item corresponent a id_io i fa l'equivalent a esperar l'operació:

  - wait (id_io)
  - Extreure l'item corresponent a id_io a la cua q_io_fin
  - Decrementar el nombre d'E/S pendents del procés (al pcbsys)

Observacions

- És síncrona:
  Espera a que finalitzin les E/S pendents d'aquest canal

- Es perden els resultats: Les operacions es realitzen però no es comprova si han anat bé o no

- Alliberar, per a totes les E/S pendents:
  - estructura de l'item extret de la cua q_io
  - estructura de l'item extret de la cua q_io_fin
  - id_io (semàfor associat de finalització)

- Hi ha rutina dependent: Actuarà segons les característiques del dispositiu

- La rutina dependent decrementarà el nombre d'herències del DD i alliberarà el DD si cal

```
/* Tanca el canal
 * Espera a que finalitzin E/S pendents d'aquest canal
 * Es perden els resultats
 * Rutina Dependent: Actua segons característiques del
 *                           Dispositiu
 */


int tancar (canal)
int canal;
{
 struct dd * eldd;
 struct pcbsys * aux;
 int error;
 item * elem;

 aux = obtenir_pcb(quisoc());
 if (canal ∉ 0..4) return(-2)
 if ((aux->t_c[canal].pdd == NIL) return(-1);

/*No hi han errors */
 eldd = aux->t_c[canal].pdd;
 while (!buida(&(aux->t_c[canal].q_io))) {
       elem = primer(&(aux->t_c[canal].q_io));
       wait (elem->id_io);
       buscar_id_io (eldd->q_io_fin, elem->id_io);
       aux->n_es--;
     }
 error = (*eldd->tancar)(eldd);
 aux->t_c[canal].pdd = NIL;

 return(error);
}
```

- buscar_id_io (cua, id_io)

```
Busca l'id_io a la cua especificada.
Si el troba:
- El desencua i allibera l'estructura
- Retorna 0
si no:
- Retorna -1
```