SISTEMAS OPERATIVOS II
Tercer curso Ingeniería Infomática.
Curso 2008-2009

Práctica 4: Procesos en UNIX: memoria compartida. Sistemas de ficheros.

El shell creará un sistema de ficheros sencillo (**ver información el final del enunciado**) en memoria compartida. Los comandos *copy*, *move delete*, *stat*, *listfs* y *makefs* acceden a dicho sistema de ficheros. El shell además debe mantener una lista (puede usarse una de las listas implementadas en prácticas anteriores) de las direcciones zonas de memoria compartida que tiene mapeadas en memoria. Para cada zona de emoria debe guardar además el tamaño. En los casos en que los comandos antes mencionados reciban un fichero como parámetro, la sintaxis sera:

nombre_de_fichero@identificacion_zona_memoria_compartida

La identificación de la zona de memoria compartida puede ser

a **por clave**. En este caso **HAY QUE MAPEARLA** con *shmget* y *shmat* (si ya está mapeada se vuelve a mapear) Ejemplo:
   - `ficherillo.dat@159` especifica el fichero de nombre *ficherillo.dat* en la zona de memoria de clave 159.

b **por dirección**. La dirección representa una dirección de memoria donde está mapeada un zona de memoria compartida Ejemplo:
   - `ficherillo.dat@0xdfc00000` especifica el fichero de nombre *ficherillo.dat* en la zona de memoria compartida de mapeada en la dirección 0xdfc00000.

c Si no se especifica identificación de la zona de memoria compartida se entiende que es un fichero de disco.
   Ejemplo:
   - `datos.txt` el fichero de nombre *datos.text* en el directorio actual del shell

Para distiguir si la identificación es una direccion de memoria o una clave se mirará primero si existe una zona de memoria compartida mapeada en dicha dirección (comprobando la lista) en caso contrario se asumirá que representa una clave y se procederá a mapearla con *shmget* y *shmat*. Siempre que se mapee una zona de memoria compartida con *shmat*, se añadirá la dirección obtenida a la lista de direcciones de memoria compartida. Tengase en cuenta que es posible (e incluso muy probable) **que la misma zona de memoria esté mapeada varias veces en el espacio de direcciones del proceso**

Se añadirán al intérprete de comandos de las prácticas anteriores las siguientes funciones

- **makefs [id_memoria [tamaño]]**
  Crea un sistema de ficheros en la zona de memoria especificada por *id_memoria*. *id_memoria* puede ser una clave o una dirección de memoria.
  - Si es una dirección de memoria, debe existir una zona de memoria compartida mapeada en dicha dirección (puede saberse comprobando la lista) y no se especifica el tamaño
  - Si se trata de una clave debe obtenerse la memoria con *shmget* y *shmat* (y añadirse a la lista): si no se especifica el tamaño se entiende que ya existe una zona de memoria creada con esa clave (aunque no esté mapeada); si se especifica el tamaño intentará crear una zona de memoria compartida de ese tamaño
  - Para saber si se trata de una dirección de memoria o de una clave se procederá como se ha descrito anteriormente, se supone que es una dirección de memoria y se comprueba en la lista de direcciones de memoria obtenidas com shmat. En caso de no ser una dirección válida se asume que *id_memoria* representa una clave

  Nótese que en caso de ya existir dicha zona de memoria crear el sistema de ficheros consiste en inicializar las estructuras de dicho sistema de ficheros, es decir, "formaterarlo".
  En el caso de no suministrar argumentos, listará las zonas de memoria compartida que tiene mapeadas el proceso.
- **listfs [id_memoria]**
  Lista los contenidos del sistema de ficheros que hay en la zona de memoria compartida identificada por *id_memoria*. Al igual que en el caso anterior *id_memoria* puede representar una dirección o una clave. Se aplica todo lo dicho en el apartado anterior.
  En el caso de no suministrar argumentos, listará las zonas de memoria compartida que tiene mapeadas el proceso.
- **delete [id_fichero]**
  Elimina el fichero *id_fichero*. *id_fichero* es de la forma mombre_fichero@id_memoria de manera que, tal como se ha descrito anteriormente, si no se especifica zona de memoria se entiende que es un fichero de disco y ... el comando es compatible con el de la práctica 1.
  En el caso de no suministrar argumentos, listará las zonas de memoria compartida que tiene mapeadas el proceso.
- **stat [id_fichero]**
  Muestra información del fichero *id_fichero*. *id_fichero* es de la forma

mombre_fichero@id_memoria de manera que, tal como se ha descrito anteriormente, si no se especifica zona de memoria se entiende que es un fichero de disco y ... el comando es compatible con el comando *stat*de la práctica 1.

En el caso de no suministrar argumentos, listará las zonas de memoria compartida que tiene mapeadas el proceso.

- **copy id_fichero1 [id_fichero2]**
  Copia *id_fichero1* en *id_fichero2*. Tanto *id_fichero1* como *id_fichero2* son de la forma nombre_fichero@id_memoria. Con lo que este comando puede usarse para copiar ficheros de disco a disco, de disco al sistema de ficheros en memoria, del sistema de ficheros en memoria a disco o entre sistemas de ficheros en memoria. Si *id_fichero2* es de la forma @id_memoria, se entiende que se copia con el mismo nombre que tiene en*id_fichero1*. Si *id_fichero2* no aparece, se entiende que es al disco y con el mismo nombre que *id_fichero1*. Este comando debe poder copiar ficheros dentro del mismo sistema de ficheros en memoria.
- **move id_fichero1 [id_fichero2]**
  Mueve *id_fichero1* en *id_fichero2*. Exactamente igual que `copy id_fichero1 [id_fichero2]` pero eliminando el fichero origen *id_fichero1*
- **detach [direccion_memoria]**
  El argumento es una dirección de memoria del proceso donde hay mapeada una memoria compartida: la desmapeará (con *shmdt*) y la eliminará de la lista de direcciones de memoria compartida mapeadas. Si se invoca sin argumentos, lista las zonas de memoria compartida que tiene mapeadas el proceso.

EJEMPLOS

- `-> makefs 1500 1000000`
  Crea un sistema de ficheros nuevo de 1000000 bytes en una zona de memoria compartida de clave 1500. Si ya existe dicha zona de memoria mostrará un mensaje de error
- `-> makefs 1500`
  Crea un sistema de ficheros nuevo (inicializa el que hay) en una zona de memoria compartida de identificador 1500. SI no existe nada en la dirección de memoria 1500, asume que 1500 es una clave. Se supone que dicha zona de memoria ya existe; en caso contrario mostrará un mensaje de error
- `-> listfs 10`
  Lista los contenidos del sistema de ficheros que hay en la zona de memoria compartida identificada por 10. Si no hay nada en la dirección de

memoria 10 del shell, asumirá que 10 es una clave.

```
->listfs 10
Memoria con clave 10 en 0xb47b7000: 12 ficheros
     57115 rwxr-xr-x Wed Apr  1 12:32:28 2009 shell-ii-2007.c
     54872 rwxr-xr-x Wed Apr  1 12:32:28 2009 a.out
  13432886 rw-r--r-- Wed Apr  1 12:32:33 2009 Unix-SistemaFicheros.pdf
      4495 rw-r--r-- Wed Apr  1 12:32:28 2009 senales.c
      6439 rw-r--r-- Wed Apr  1 12:32:33 2009 OcultaFicheros.c
     14165 rw-r--r-- Wed Apr  1 12:32:28 2009 unix-varios.pdf
   6619071 rwxr----- Wed Apr  1 12:32:30 2009 Unix-Procesos.pdf
        37 rw-r--r-- Wed Apr  1 12:32:28 2009 ENTRADA
   4082047 rw-r--r-- Wed Apr  1 12:32:29 2009 Unix-Memoria2.pdf
   4082047 rw-r--r-- Wed Apr  1 12:32:29 2009 Unix-Memoria.pdf
     37824 rwxr-xr-x Wed Apr  1 12:32:28 2009 shell-old.c
     62492 rw-r--r-- Wed Apr  1 12:32:33 2009 presentacion.pdf
->
```

- `-> stat senales.c@0xb47b7000`
  Nos muestra información del fichero senales.c en el sistema de ficheros que está en la memoria identificada por 0xb47b7000. (Nótese que stat senales.c@10 produciría la misma salida pero habría mapeado la memoria de clave 10 una vez más, suponiendo, claro está, que la zona de memoria compartida con clave 10 está mapeada en la dirección 0xb47b7000)

  ```
  -> stat senales.c@0xb47b7000
        4495 rw-r--r-- Wed Apr  1 12:32:28 2009 senales.c
  ->
  ```

- `-> copy p2.c prueba.c@0xb47b7000`
  Copia el fichero p2.c al sistema de ficheros que está en la memoria identificada por 0xb47b7000 con el nombre prueba.c (Nótese que `copy p2.c prueba.c@10` haría los mismo pero habría mapeado la memoria de clave 10 una vez más)

- `-> copy p2.c @0xb47b7000`
  Igual que le anterior pero ahora la copia tiene el mismo nombre.

- `-> copy p2.c@0xb47b7000 prueba.c`
  Copia el fichero p2.c desde sistema de ficheros que está en la memoria identificada por 0xb47b7000 al disco con nombre prueba.c (Nótese que `copy p2.c@10 prueba.c` haría lo mismo pero habría mapeado la memoria de clave 10 una vez más) Si hicésemos `copy p2.c@10` o `copy p2.c@0xb47b7000` el nombre de la copia sería p2.c

- `-> copy p2.c@0xb47b7000 prueba.c@0xb0fc0000`

Copia el fichero p2.c desde sistema de ficheros que está en la memoria identificada por 0xb47b7000 a sistema de ficheros que está en la zona de memoria compartida identificada por 0xb0fc0000 con nombre prueba.c (Nótese que `copy p2.c@10 prueba.c@0xb0fc0000` haría lo mismo pero habría mapeado la memoria de clave 10 una vez más) Si hicésemos `copy p2.c@10 @0xb0fc0000` o `copy p2.c@0xb47b7000 @0xb0fc0000`, el nombre de la copia sería p2.c

NOTAS SOBRE EL SISTEMA DE FICHEROS EN MEMORIA COMPARTIDA

El sistema de ficheros ha de usar asignación contigua y está definido por las estructuras que se muestran a continuación (deben usarse en el código) El miembro *desplazamiento* de los datos de un fichero se mide respecto al comienzo de la zona de datos del sistema de ficheros (es 0 para el fichero cuyos datos están al principio del area de datos). El miembro *tamano* del sistema de ficheros es el tamanño TOTAL (el de la zona de memoria compartida donde está dicho sistema) e incluye las estructuras de dicho sistema de ficheros. El miembro *elibre* es el espacio disponible para datos de los ficheros, es decir *tamano* menos lo que ocupen las estructuras del sistema de ficheros y ls datos de los ficheros en él contenidos en un instante dado. *nfich* es el número de ficheros en el sistema de ficheros: 0 representa que está vacio. Los miembros *reservado* son de uso libre, pero el resto de los items deben tener información coherente de manera que la información que se almacene los miembros em reservados no sea imprescidible para el manejo del sistema de ficheros.

```
#define SFM_MARCA 0x0a2308ea   /*marca del sistema de ficheros*/
#define SFM_MAXFICH  128    /*numero maximo de ficheros*/
#define SFM_MAXNOMBRE 256   /*tamano maximo del nombre en este sistea ficheros*/
#define SFM_RESERVADOS 8

struct SB { /*info del sistema de ficheros*/
  unsigned marca; /*la marca, debe se SFM_MARCA*/
  unsigned tamano; /*tamano total sistema de ficheros*/
  unsigned elibre; /*espacio libre en sistema de ficheros*/
  unsigned nfich;  /*numero de ficheros en el sistema de ficheros*/
  unsigned reservado[SFM_RESERVADOS]; /*por si se quieren usar mas cosas*/
  };

struct NODOFICH{  /*informacion de un fichero */
  char nombre[SFM_MAXNOMBRE]; /*el nombre*/
  mode_t modo;                /*los permisos*/
  uid_t uid;                  /*propietario original*/
```

```
  off_t tamano;                /*tamano en bytes*/
  time_t fecha;                /*fecha en que se ha copiado al SF*/
  off_t desplazamiento;        /*desplazamiento a sus datos desde el principio*/
  unsigned enuso;              /*0 esta estructura NO contiene info de un fichero*/
  unsigned reservado [SFM_RESERVADOS];
  };

struct SF {    /*el sistema de ficheros*/
    struct SB sb;
    struct NODOFICH n[SFM_MAXFICH]; /*los nodos de los ficheros*/
    unsigned reservado[SFM_RESERVADOS];
    char datos[1];        /*el area de datos */
    };
```

FORMA DE ENTREGA

Como en las prácticas anteriores

FECHA DE ENTREGA Viernes 12 Junio 2009