SISTEMAS OPERATIVOS II

Tercer curso Ingeniería Informática. Curso 2007-2008

Práctica 1: Procesos en UNIX. Recursos IPC.

Comenzar la codificación de un intérprete de comandos (shell) en UNIX, que se irá completando en suscesivas prácticas. Hay que tener en cuenta que

- Los argumentos entre corchetes [] son opcionales.
- Los argumentos separados por | indican que debe ir uno u otro, pero no ambos simultaneamente.
- No debe dilapidar memoria (ejemplo: variable que se asigna cada vez que se llama a una función y no se libera).
- Cuando el shell no pueda ejecutar una acción por algún motivo, debe indicarlo con un mensaje como el que se obtiene con `sys_errlist[errno]` o con *perror()* (por ejemplo, si no puede cambiar de directorio debe indicar por qué).
- En ningún caso debe producir un error de ejecución (segmentation, bus error ...), salvo que se diga explicitamente
- Las direcciones de memoria deben mostrase en **hexadecimal**.
- La información que se nuestra en pantalla no debe incluir en ningun caso líneas en blanco.
- El shell leerá de su entrada estándar y escribirá en su salida estándar, de manera que podría ser ejecutado un archivo de comandos invocando al shell con su entrada estándar redireccionada a dicho archivo.

En esta primera práctica, el intérprete de comandos, además de algunos comandos de uso general nos permitirá:

- crear sistemas de ficheros sencillos en bloques de memoria compartida. Los ficheros se podrán copiar y mover a/desde esta zona de memoria al disco. Además se podrán listar y eliminar. La familia de comandos *shared-** permite acceder estos sistemas de ficheros
- asignar y desasignar memoria dentro del shell, asi como inspeccionar sus contenidos

- mapear y desmapear ficheros en memoria así como leer y/o escribir ficheros a/desde la memoria del proceso
- el shell mantendrá la contabilidad (mendiante una o varias listas) de todas las direcciones de memoria obtenidas mediante la llamadas *shmat* y *mmap* y el comando *malloc*, de manera que en cualquier momento (mediante los comandos malloc, mmap, detach o direcciones) podamos conocer cuales son estas direcciones de memoria. Los comandos *detach, munmap* y *free* permiten, en su caso, eliminar una dirección de memoria de esta contabilidad.

Los comandos que incluirá el intérprete de comandos en esta primera práctica son

quit Termina la ejecución del intérprete de comandos.

exit Termina la ejecución del intérprete de comandos.

autores Indica los nombres y los logins de los autores de la práctica.

chdir [dir] Cambia el directorio actual a *dir*. Si no se le suministra argumento informa del directorio actual.

prompt *prompt* Cambia el indicador (prompt) del intérprete de comandos.

pid [-p] Muestra el pid del proceso con `-p` muestra tambian el de su proceso padre

fork El intérprete de comandos crea un hijo con fork() y se queda (el propio intérprete de comandos) en espera hasta que dicho hijo (el creado con fork) termine.

rec [-a|-n|-d] [-sN] n Invoca a la función recursiva n veces, la opción -a, -n o -d indica si la memoria asignada con malloc en dicha función debe liberarse (a)ntes de la siguiente llamada recursiva, (d)espués o (n)o liberarse. Si no se indica una opción se supone que la memoria se libera despues de la llamada (-d). En parámetro -sN indica cuanto tiempo (en segundos) debe esperar la última llamada antes de terminar. (ejemplo -s10 indicaría que la ultima iteracción de la función recursiva debería hacer una espera (mediante *sleep*) de 10 segundos. La función recursiva recibe tres parámetros: uno que indica el número de veces que se tiene que invocar, otro que indica cuando liberar la memoria asignada con malloc y un tercero que indica cuanto tiene que esperar la última iteración Además

esta función tiene 3 variables, un array automatico de 1024 caracteres, un array estático de 1024 caracteres y un puntero a caracter.
Esta función debe hacer lo siguiente

1. asignar memoria (mediante malloc) al puntero para 1024 caracteres.
2. imprimir
   - el valor del parámetro que recibe así como la dirección de memoria donde se almacena.
   - el valor del puntero así como la dirección de memoria donde se almacena.
   - la dirección de los dos arrays (el nombre del array como puntero).
3. si se ha especificado la opción -a liberar la memoria asignada al puntero.
4. invocarse a si misma con (n-1) como parámetro (si n>0).
5. si es la última iteración (n=0) y se ha especificado `-sN` esperar N segundos (por medio de *sleep.*
6. *si se ha especificado la opción -d liberar la memoria asignada al puntero.*

Un posible código para la función recursiva (sin las definiciones de constantes) podría ser:

```
void recursiva (int n, int liberar, int delay)
{
char automatico[TAMANO];
static char estatico[TAMANO];
void * puntero;

puntero=(void *) malloc (TAMANO);
printf ("parametro n:%d en %p\n",n,&n);
printf ("valor puntero:%p en direccion: %p\n", puntero,&puntero);
printf ("array estatico en:%p \n",estatico);
printf ("array automatico en %p\n",automatico);
if (liberar==ANTES)
   free (puntero);
```

```
if (n>0)
   recursiva(n-1,liberar,delay);
if (liberar==DESPUES)
   free (puntero);
if (n==0 && delay)        /* espera para la ultima recursividad*/
   sleep (delay);
}
```

malloc [tamano] asigna en el shell la cantidad de memoria que se le especifica (utilizando la llamada *malloc*). y nos informa de la direccion de la memoria asignada. Si no se especifica tamaño los dará una lista de las direcciones de memoria asignadas **con el comando malloc**

free [-f] dir libera la memoria asignada a dir (usando la función de librerá *free*). Si la memoria no fue asignada con el comando previo *malloc* nos dará un aviso y no la liberará. Si se especifica -f intentará liberarla aunque la comprobación anterior no de resultado positivo (podría producir error)

display dir [cont] Mustra los contenidos de *cont bytes* a partir de la posición de memoria *dir*. Si no se especifica cont imprime 25 bytes. Para cada byte imprime, en distintas líneas el caracter asociado (en caso de no ser imprimible imprime in espacio en blanco) y su valor en hexadecimal. Imprime 25 bytes por línea. (podría producir error)

stat fich Nos da información del fichero *fich* (nombre, tamaño, permisos, fecha de ultimo acceso)

delete fich Elimina el fichero *fich*

list [-l][dir] Lista los ficheros del directorio *dir*. Si no se especifica dir se entiende que es el directorio actual. -l indica que han de darse tambien los detalles de los ficheros (tamanño, permisos, ...)

mmap [fichero] mapea en memoria el fichero especificado. Se mapeará el fichero en toda su longitud a partir del *offset* 0. Devuelve la dirección de memoria donde lo ha mapeado. Utiliza la llamada *mmap*. Si no se especifica fichero nos informa de las direcciones de memoria donde hay mapeados ficheros, indicandonos la dirección, el tamanño del mapeo y el fichero que hay mapeado en ella

munmap [-f] dir desmapea la dirección de memoria *dir* del fichero que haya mapeado en ella. Si en esa posición de memoria no hay nada mapeado con el comando *mmap* nos dará un aviso y no la desmapeará. Si se especifica -f

intentará desmapearla aunque la comprobación anterior no de resultado positivo (podría producir error)

read file dir Copia el fichero *file* en la dirección de memoria *dir* (podría producir error)

write [-f] file dir cont Copia desde la dirección de memoria *dir cont* bytes en el fichero *file*. -f especifica que se sobreesciba el fichero (si existe) (podría producir error)

Los siguienes comandos se refieren al sistema de ficheros que se crea en memoria compartida. En los comandos *shared-creat y shared-del* la zona de memoria se especifica por la clave. En el resto de comandos el número que especifica la memoria compartida puede representar tanto una clave como una dirección de memoria donde está mapeada. En el caso de que especifique una clave, se obtendrá con *shmget* y *shmat*, aunque ello implique que la misma región de memoria compartida esté mapeada en varias direcciones. En los comandos *shared-cp, shared-mv* y *shared-rm* se utilizará la forma "id:nombre" para especificar un fichero en la zona de memoria compartida de identificador *id* (id puede representar una clave o una dirección)

shared-creat m N Crea una zona de memoria compartida con la clave m de tamaño N bytes, y la mapea en memoria. Imprime la dirección donde se ha mapeado en memoria. Si la zona ya existe nos informará de ello.

shared-del m Elimina la zona de memoria compartida de clave m

shared-info m El programa informa del tamaño de la zona de memoria compartida especificada por m, asi como de los detalles (tamaño, permisos en formato rwxrwxrwx, fecha de último acceso y nombre) de cada uno los ficheros en ella contenidos.(UNA SOLA LÍNEA por cada fichero). En este caso m puede especificar tanto una clave como la dirección de memoria donde se ha mapeado la memoria compartida. En el caso de especificar la clave, la mapeará de nuevo y no será desmapeada hasta que se especifique el comando *detach*

shared-cp fich1 fich2 copia el fichero *fich* en *fich2*. Tanto *fich1* como *fich2* pueden representar un fichero de disco o un fichero en una zona de memoria compartida. En caso de ser un fichero en una zona de memoria compartida se denotara por "id:nombre" donde id es la identificación de la zona de memoria compartida (recuérdese, clave ó dirección) .

Ejemplo

```
#shared-cp prueba.c 124:prueba.c
```

(copia el fichero prueba.c a la zona de memoria de clave 124 con el nombre prueba.c)

```
#shared-cp /etc/pruebas/prueba.c 0xdc800000:nada.c
```

(copia el fichero /etc/pruebas/prueba.c a la zona de memoria mapeada en la dirección 0xdc800000 con el nombre nada.c)

```
#shared-cp 123:prueba.c 0xdc800000:pru2.c
```

(copia el fichero prueba.c de la zona de memoria con clave 124 a la zona de memoria mapeada en la dirección 0xdc800000 con el nombre pru2.c)

```
#shared-cp 123:prueba.c ../pru3.c
```

(copia el fichero prueba.c de la zona de memoria con clave 124 al disco con el nombre pru3.c, en el directorio padre del directorio actual)

shared-mv fich1 fich2 mueve el fichero *fich* a *fich2*. Exactamente igual a *shared-cp* salvo que destruye el fichero original.

shared-rm fich elimina el fichero *fich*. si *fich* es de la forma "id:nombre" elimina el fichero *nombre* de la zona de memoria de identificador id (otra vez: id puede representar una clave o una dirección). Si *fich* no es de la forma id:nombre entonces este comando es igual al comando delete.

detach [-f] [dir] desencadena del espacio de direcciones la memoria asociada a la dirección *dir* (usando la llamada *shmdt*). Si *dir* no representa una dirección obtenida mediante *shmat* nos informará de ello y no hará nada. Si se especifica -f hará la llamada *shmdt* aun en el caso de que la comprobación anterior no de resultado positivo. Si no se especifica dirección nos informará de todas las direcciones donde hay mapeada una zona de memoria compartida.

direcciones [-p|-v] Muestra las direcciones de memoria

  - de zonas de memoria compartida
  - ficheros mapeados en memoria
  - zonas de memoria asignadas con el comando *malloc*

  con la opción [-v] muestra las direcciones de memoria de
  - las variables globales
  - las variables locales de *main*
  - las funciones del programa llamadas desde *main*

Información detallada de las llamadas al sistema y las funciones de la librería debe obternerse con man (stat, shmget, shmat, shmdt, shmctl, read, write, opendir, malloc, free, mmap, munmap ...)

Salvo que se diga explicitamente, en NINGUN CASO la práctica puede producir error en tiempo de ejecución.

FORMA DE ENTREGA Va a ser utilizado el servicio de recogida de prácticas suministrado por el Centro de Cálculo de esta Facultad y parte del proceso de corrección de las prácticas va a ser automático (compilación, listado de practicas entregadas etc) por lo cual deben entregarse **exactamente** como se indica a continuación:

- Se colocará el código fuente de la práctica en el directorio asignado para ello antes de la fecha tope de entrega de la práctica.
- Se entregará UN SOLO fichero fuente por práctica, de nombre pN.c (N el número de práctica). Por ejemplo, para esta práctica será p1.c (en minúsculas).
- Los grupos de prácticas son de **2 (DOS)** alumnos. La práctica SOLO DEBE SER ENTREGADA POR UNO DE LOS MIEMBROS DEL GRUPO
- en el código fuente de la práctica debe figurar como comentario el nombre de los autores **exactamente** en el siguiente formato

  ```
  /*
  AUTOR:apellido11 apellido12, nombre1:login_en_el_que_se_entrega
  AUTOR:apellido21 apellido22, nombre2:login_en_el_que_se_entrega
  */
  ```

  donde:

1. La palabra autor aparece en mayúsculas.
2. Los apellidos y el nombre de los autores están totalmente en minúsculas.
3. apellidoij representa el apellidoj del componente i del grupo de prácticas.
4. No hay espacios antes y despues de los dos puntos.
5. El login que aparece es el del que entrega la práctica (aparece el mismo login en las dos líneas).
6. Los símbolos de comentarios están en líneas distintas.
7. No debe incluirse la letra ñ ni vocales acentuadas en los nombres

FECHA DE ENTREGA VIERNES 11 ABRIL 2008