SISTEMAS OPERATIVOS II

Tercer curso Ingeniería Técnica de Sistemas. Curso 2008-2009

Práctica 4: Memoria en UNIX.

Continuar la codificación de un intérprete de comandos (shell) en UNIX. Al igual que en la práctica anterior

- 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.

El shell debe mantener en memoria varias (o una) listas con la siguiente información.

- La memoria que se asigna con el comando *malloc*. Guardará la dirección de memoria obtenida y el tamaño asignado de cada vez. El comando *malloc -f*, si procede, eliminará un elemento de esta lista.
- Los ficheros mapeados con el comando *mmap*. Guardará la dirección de memoria y el tamaño del mapeo, así como el nombre de fichero mapeado. El comando *mmap -u*, si procede, eliminará un elemento de esta lista.
- Las direcciones de memoria compartida que ha mapeado el proceso con la llamada *shmat* (comando *shmat* o *vfs-**,en caso de especificar clave). EL comando *shmat -d* si procede, eliminará un elemento de esta lista.

Dentro de los comandos a implementar en el shell los que comienzan por *vfs-* se refieren todos a operaciones sobre un sistema de ficheros virtual creado en una zona de memoria compartida. En todos los casos la zona de memoria se puede especificar de dos maneras: por clave (*cl*) o por dirección (*dir*). La implementación del sistema de ficheros en memoria compartida es libre (puede usuarse asignación contigua, enlazada o indexada), pero no debe tener fragmentación externa, por lo que si se usa asignación contigua (lo mas sencillo) es necesario contemplar la posibilidad de una compactación

- **clave**. Se accederá a la memoria por medio de *shmget()* a partir de la clave y se maperá mediante *shmat*, aunque ello suponga tener la misma memoria mapeada en varios sitios. Se añadirá la dirección obtenida con *shmat()* a la lista de direcciones de memoria compartida del proceso. NO SE BUSCARA EN LA LISTA SI YA HAY UNA MEMORIA CON ESA CLAVE PARA USARLA
- **dirección**. En este caso se comprobará (buscando en la lista) que la dirección que se suministra es en efecto una dirección de una zona de memoria compartida.

vfs-creat cl|dir [tam] Crea un sistema de ficheros virtual en una zona de memoria compartida, es decir, inicializa las estructuras de dicho sistema de ficheros. En el caso de suministrar la clave es necesario especificar tambien el tamaño; si ya existe una memoria compartida con esa clave producirá un error. Si se especifica una dirección (y dicha dirección existe) esta operacion supondrá el *"formateo"* del sistema de ficheros.

vfs-cp cl|dir fich [nu] Copia el fichero *fich* a la zona de memoria especificada por *cl* o *dir*. Si se especifica *nu*, éste será el nombre de la copia.

vfs-pc cl|dir fich [nu] Copia el fichero *fich* de la zona de memoria especificada por *cl* o *dir* a disco. Si se especifica *nu*, éste será el nombre de la copia

vfs-mv cl|dir fich [nu] Mueve (copia eliminando el original) el fichero *fich* a la zona de memoria especificada por *cl* o *dir*. Si se especifica *nu*, éste será el nombre de la copia destino.

vfs-vm cl|dir fich [nu] Mueve (copia eliminando el original) el fichero *fich* de la zona de memoria especificada por *cl* o *dir* a disco. Si se especifica *nu*, éste será el nombre de la copia destino.

vfs-rm cl|dir fich Elimina *fich* de la zona de memoria compartida especificada por *cl* o *dir*

vfs-ls cl|dir Muestra información de los ficheros contenidos en la zona de memoria especificada por *cl* o *dir*

vfs-destroy cl|dir Elimina (mediante *shmctl(IPC_RMID...)* La zona de memoria especi-

ficada por *cl* o *dir*. NO DEBE DESMAPEARLA

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 y la dirección del parámetro que recibe.
   - el valor y la dirección del puntero.
   - 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) esperar, de haberse especificado, el tiempo requerido.
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);
}
```

mmap [-u] [fichero] mapea (o desmapea) 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. Si se especifica *-u* el fichero será desmapeado y la dirección de memoria correspondiente será eliminada de la lista.

malloc [-f] [tam] asigna (o desasigna) 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**. Si se especifica *-f* hace *free* de una de las zonas de tamaño *tam* asignadas **con el comando malloc** y la dirección de memoria correspondiente será eliminada de la lista.

shmget cl [tam] Crea (o accede) a una zona de memoria especificada por la clave *cl* y de tamaño *tam*. Nos informa del identificador obtenido. El tamaño solo es relevante en el caso de que la zona sea creada.

shmat [[-d] id] Mapea una zona de memoria compartida de identificador *id* (obtenido con shmget) en el espacio de direcciones del proceso. Nos informa de la dirección de memoria obtenida y la añade a la lista de direcciones de memoria compartida. Si se especifica -d se entiende que el segundo argumento (*id*) es una dirección de memoria obtenida con *shmat()* y procederá a desmapearla con *shmdt* y a eliminarla de la lista correspondiente. Si no se especifican argumentos, nos mostrará las direcciones de memoria donde hay mapeadas zonas de memoria compartida, indicando para cada dirección, el tamaño y la clave de la zona de memoria compartida en ella mapeada.

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)

display dir [cont] Muestra 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 lo ser imprimible imprime in espacio en blanco) y su valor en hexadecimal. Imprime 25 bytes por línea. (podría producir error). Ejemplo

```
->malloc 100000
asignado 100000 en 0x8053088
->read shell.c 0x8053088
Leidos 57115 bytes
     57115 rwxr-xr-x Thu Nov 20 13:32:22 2008 shell.c
->display 0x8053088 300
  #  i  n  c  l  u  d  e     <  u  n  i  s  t  d  .  h  >     #  i  n  c
 23 69 6E 63 6C 75 64 65 20 3C 75 6E 69 73 74 64 2E 68 3E 0A 23 69 6E 63 6
  u  d  e     <  s  t  d  i  o  .  h  >     #  i  n  c  l  u  d  e     <
 75 64 65 20 3C 73 74 64 69 6F 2E 68 3E 0A 23 69 6E 63 6C 75 64 65 20 3C 7
  t  r  i  n  g  .  h  >     #  i  n  c  l  u  d  e     <  s  t  d  l  i
 74 72 69 6E 67 2E 68 3E 0A 23 69 6E 63 6C 75 64 65 20 3C 73 74 64 6C 69 6
  .  h  >     #  i  n  c  l  u  d  e     <  s  y  s  /  t  y  p  e  s  .
 2E 68 3E 0A 23 69 6E 63 6C 75 64 65 20 3C 73 79 73 2F 74 79 70 65 73 2E 6
  >     #  i  n  c  l  u  d  e     <  s  y  s  /  s  t  a  t  .  h  >
 3E 0A 23 69 6E 63 6C 75 64 65 20 3C 73 79 73 2F 73 74 61 74 2E 68 3E 0A 2
```

direcciones Muestra las direcciones de memoria de

- las zonas de memoria compartida
- ficheros mapeados en memoria
- zonas de memoria asignadas con el comando *malloc*
- las variables globales
- tres funciones del programa
- tres variables locales
- tres parámetros a función

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

FORMA DE ENTREGA

Como en las prácticas anteriores

FECHA DE ENTREGA VIERNES 24 ENERO DE 2009