

SISTEMAS OPERATIVOS II

Tercer curso Ingeniería Técnica Informática Sistemas . 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 sucesivas 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 `perro()` (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 explícitamente
- Las direcciones de memoria deben mostrarse en **hexadecimal**.
- La información que se muestra en pantalla no debe incluir en ningún 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.

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

quit Termina la ejecución del intérprete de comandos.

fin Termina la ejecución del intérprete de comandos.

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

cd [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.

getpid Muestra el pid del proceso y 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 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);
}
```

shared -c|-d|-i *m* [N] Manejo de zonas de memoria compartida, en las que el shell puede guardar ficheros

– **shared -c *m* N**

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

– **shared -d *m*** Elimina la zona de memoria compartida de clave *m*

– **shared -i *m***

El programa informa del tamaño de la zona de memoria compartida especificada por *m*, así como de los detalles (tamaño, permisos en formato *rw-rw-rw*, fecha de último acceso y nombre) de cada uno de 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 com-

partida. En el caso de especificar la clave, la mapeará de nuevo y no será desmapeada hasta que se especifique el comando *detach*

- shared -t|-r *mem* *fich***
- **shared -t *mem* *fich*** copia el fichero *fich* a la zona de memoria especificada por *mem*. *mem* puede representar tanto una clave como una direccion 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 -r *mem* *fich*** copia el fichero *fich* desde la zona de memoria especificada por *mem* a disco. *mem* puede representar tanto una clave como una direccion 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*. Si el fichero ya existe no lo sobrescribirá.
- shared -m|-n|-e *m* *fich***
- **shared -m *mem* *fich*** copia el fichero *fich* a la zona de memoria especificada por *mem*, eliminando el fichero original. *mem* puede representar tanto una clave como una direccion 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 -e *mem* *fich*** elimina el fichero *fich* de la zona de memoria especificada por *mem*. *mem* puede representar tanto una clave como una direccion 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 -n *mem* *fich*** copia el fichero *fich* desde la zona de memoria especificada por *mem* a disco y lo elimina de la zona de memoria compartida. *mem* puede representar tanto una clave como una direccion 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*. Si el fichero ya existe en disco no lo sobrescribirá y no lo eliminará de la memoria compartida.

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 tamaño 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 no hara la comprobación anterior (podría producir error)

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 librería *free*). Si la memoria no fue asignada con el comando previo *malloc* nos dará un aviso y no la liberará. Si se especifica -f no hara la comprobación anterior (podría producir error)

detach [-f] [dir] desencadena del espacio de direcciones la memoria asociada a *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 no hara la comprobación anterior. Si no se especifica dirección nos informará de todas las direcciones obtenidas mediante *shmget*. 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 (tamaño, permisos, ...)

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)

direcciones Muestra las direcciones de memoria

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

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

Salvo que se diga explícitamente, en ningún 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. apellido_i representa el apellido del componente i del grupo de prácticas.
4. No hay espacios antes y después 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 16 NOVIEMBRE 2007