SISTEMAS OPERATIVOS II

Tercer curso Ingeniería Técnica Informática de sistemas. Curso 2007-2008

Práctica 2: Procesos en UNIX.

Continuar la codificación de un intérprete de comandos (shell) en UNIX, que se irá completando en suscesivas prácticas. Dicho intérprete incluirá de momento los comandos que se citan a continuación. 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. 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 el shell con su entrada estándar redireccionada a dicho archivo. En esta práctica se implementará la ejecución en primer y segundo plano así como la ejecución con cambio de prioridad.

Nuestro shell debe mantener dos listas:

- *ruta de búsqueda*. Es la lista de directorios donde el shell busca los ejecutables externos (análogo a la variable de entorno PATH del sistema). El comando `searchlist` manipula y/o muestra dicha lista. La implementación es libre (cada uno como quiera) pero NO DEBE implementarse como una variable de entorno (ni usar la variable de entorno PATH para implementarla)
- *lista de procesos en segundo plano*. Cada vez que desde nuestro shell cree un proceso en segundo plano, debe incluirlo en una lista de procesos en segundo plano. El comando back muestra y/o manipula dicha lista. La implementación de la lista es libre

searchlist [op] [dir |file] Manipula y/o muestra la ruta de bsqueda de nuestro intérprete de comandos. Los directorios de esta lista se especificarán sin el caracter '/' al final (salvo el directorio raíz "/"). Se adminten directorios relativos (p.e. "../bin") en esta lista y NO DEBEN convertirse en absolutos).*op* puede ser *add, del, chk, new, pth o fnd* Ejemplos

```
#searchlist add /usr/local/bin /* correctos */
#searchlist del  /
#searchlist  .
......
#searchlist chk /usr/local/bin/ /*incorrecto*/
#searchlist del ./              /*incorrecto*/
```

searchlist add [dir] Añade *dir* a la ruta de búsqueda.

searchlist del [dir] Elimina *dir* de la ruta de búsqueda.

searchlist chk [dir] Indica si *dir* está en la ruta de búsqueda.

searchlist new Vacía la ruta de búsqueda.

searchlist pth Añade los directorios de la variable de entorno PATH a la ruta de búsqueda.

searchlist Muestra la ruta de búsqueda.

searchlist fnd [file] Busca *file* en la lista de directorios que constituyen la ruta de búsqueda del intérprete de comandos y devuelve la trayectoria completa hasta él (análogo al comando *which* del sistema). Si no lo encuentra lo indicará. Si se le indica una trayectoria completa (empezando por ./, por / o por ../) indicará si dicho fichero existe o no
Si no se especifica *dir* o *file* en *add, del, chk* o *fnd*, se mostrará la ruta de búsqueda.

```
#searchlist add /usr/openwin/bin
#searchlist fnd xterm
/usr/openwin/bin/xterm
#searchlist fnd xternp
 xternp: no encontrado
#searchlist fnd ./aa.out
 ./aa.out: no encontrado
#searchlist fnd ./a.out
 ./a.out
```

xec *com* El intérprete de comandos ejecuta, sin crear un nuevo proceso (reemplaza su código), el programa especificado en *com*. *com* representa un ejecutable con sus parámetros. Para poder ser ejecutado, dicho ejecutable debe residir en uno de los directorios de la ruta de búsqueda del intérprete de comandos o bien especificarse la trayectoria completa

hasta él (comenzando por /, ./ o ../). Debe usarse la llamada al sistema *execv*.
Ejemplo

```
#xec /usr/bin/ls -l /home  /*ejecuta /usr/bin/ls*/
  ....
#xec ./a.out -l /*ejecuta a.out,
                 /*que esta en el directorio actual */
......
#xec ls -l   /* para ejecutar esto, es necesario */
              /*que el directorio  /usr/bin,  */
              /*donde esta el programa ls, se haya incluido */
              /* a la ruta de busqueda */
```

com Totalmente análogo al comando exec, salvo que el shell crea el proceso que ejecuta dicho comando y espera a que termine (la ejecución es, por tanto, en primer plano). Todo lo dicho sobre los ejecutables en el comado *exec* es aplicable aquí. ***com*** **representa un ejecutable junto con sus parámetros.**
Ejemplo

```
#/usr/bin/ls -l /home  /*crea un proceso que ejecuta /usr/bin/ls*/
....
#./a.out -l   /* crea un proceso que ejecuta a.out,*/
......        /* que esta en el directorio actual    */
#ls -l        /* para ejecutar esto, es necesario que el directorio */
              /* /usr/bin, donde esta el programa ls, se haya incluido */
              /* a la ruta de busqueda */
```

.

com & Análogo a *com* pero la ejecución es en segundo plano (el shell no espera a que el proceso termine). Además añade *com* a la lista de procesos en segundo plano del shell
Ejemplos

```
#xterm -e csh &
```

crea un proceso que ejecuta xterm en segundo plano.

priority [-s [valor]] [pid] Muestra o establece (-s) la prioridad del proceso pid.

priority [pid] Muestra la prioridad del proceso *pid*. Si no se le especifica *pid* muestra la prioridad del interprete de comandos.

priority -s valor [pid] Establece la prioridad del proceso *pid* a *valor*. Si no se le especifica *pid* cambia la prioridad del interprete de comandos.

xecpri valor com El intérprete de comandos ejecuta, sin crear un nuevo proceso (reemplaza su código), el programa especificado en *com* con la prioridad establecida a *valor*. *com* representa un ejecutable con sus parámetros. Para poder ser ejecutado, dicho ejecutable debe residir en uno de los directorios de la ruta de búsqueda del intérprete de comandos o bien especificarse la trayectoria completa hasta él (comenzando por /, ./ o ../). Debe usarse la llamada al sistema *execv*.
Ejemplo:

```
#xecpri 10 xterm -e csh   /*ejecuta sin crear proceso xterm -e csh*/
                          /*cambiando su prioridad a 10*/
```

....

pri valor com [&] Crea un proceso que ejecuta el comando especificado en *com* con la prioridad establecida a *valor*. Al igual que en *xeq, xeqpri* y en la ejecucón en primer y segundo plano *com* representa un ejecutable con sus parámetros. Para poder ser ejecutado, dicho ejecutable debe residir en uno de los directorios de la ruta de búsqueda del intérprete de comandos o bien especificarse la trayectoria completa hasta él (comenzando por /, ./ o ../). Debe usarse la llamada al sistema *execv*. En el caso de que la ejecución sea en segundo plano, el shell añadirá el proceso a la lista de procesos en segundo plano. Ejemplo:

```
#pri 10 xterm -e csh &   /*crea un  proceso que ejecuta */
                         /*xterm -e csh en segundo plano*/
                         /*cambiando su prioridad a 10*/
```

....

back [**-d** [[**-n**] [**-s**] [**-p**] [**-a**] |[**pid1** [**pid2**]**...**]]] Muestra o elimina (-d) de la lista de procesos en segundo plano los procesos con pids *pid1*, *pid2* .... Alternativamente, en lugar de especificar los pids puede usarse

- -n todos los que ha terminado normalmente
- -s todos los que han terminado debido a una señal
- -p todos los que están parados
- -a todos los activos

Si no se especifica sobre que conjunto de procesos quiere actuarse (bien mediante lista de pids o bien mediante las opciones -a, -s ...) se entenderá que en el caso de listar es todos y en el caso de eliminar es ninguno

En el caso de listar, se indicará, para cada proceso, EN UNA SOLA LINEA

- pid del proceso
- estado, uno de los siguientes
  * ACTIVO
  * TERMINADO POR SEÑAL, se indicará las señal que lo ha parado
  * TERMINADO NORMALMENTE, se indicará el valor devuelto
  * PARADO, se indicará la señal que lo ha parado
- prioridad actual del procesos
- instante de comienzo del proceso
- línea de comando que se está ejecutando

Información detallada de las llamadas al sistema y las funciones de la librería debe obternerse con man (exec, fork, strtok, waitpid, getpid..)

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 automtíco (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 p2.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_del_que_entrega_la_practica
AUTOR:apellido21 apellido22, nombre2:login_del_que_entrega_la_practica
*/}
```

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.

FECHA DE ENTREGA VIERNES 21 DICIEMBRE 2007