SISTEMAS OPERATIVOS II

Tercer curso Ingeniería Informática. Curso 2004-2005

Práctica 1: Procesos en UNIX. Entorno

Comenzar la codificación de un intérprete de comandos (shell) en UNIX, que se irá completando en suscesivas prácticas. Dicho intérprete incluirá de momento los comados 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. Los argumentos que van entre [] son opcionales. No debe dilapidar memoria (ejemplo: variable que se asigna cada vez que se lee de teclado y no se libera). Cuando el shell no pueda ejecutar una acción por algún motivo, debe indicarlo (por ejemplo, si no puede cambiar de directorio debe indicar por qué) con un mensaje como el que se obtiene con `sys_errlist[errno]` o con *perror()*. 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.

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

fin Termina la ejecución del intérprete de comandos.

exit Termina la ejecución del intérprete de comandos.

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

entorno [-e] Muestra la lista de todas las variables de entorno accediendo mediante el tercer argumento de main. Para cada variable de entorno muestra la variable (nombre=valor), la dirección de memoria donde se almacena la variable (el valor del puntero, e[i]) y la dirección de memoria donde se almacena el puntero (&e[i]). Después de las variables se muestra el valor (como puntero) del tercer argumento de main y de la variable *environ* así como las direcciones donde se almacenan. Si se invoca como *entorno -e* hace exactamente lo mismo pero utiliza la variable *environ* en lugar del tercer argumento de main para acceder a las variables entorno. Ejemplo

```
#entorno
```

```
......
.......
ffbefcb8->env[29]=(ffbeff86) PATH=/usr/bin:/usr/local/bin:
ffbefcbc->env[30]=(ffbeffa5) LC_MONETARY=es
ffbefcc0->env[31]=(ffbeffb4) LC_COLLATE=es
ffbefcc4->env[32]=(ffbeffc2) _INIT_NET_STRATEGY=none
ffbefcc8->env[33]=(ffbeffda) _=./entorno
20a34->environ=ffbefc44 | ffbefc24->env=ffbefc44  | ffbefc20->argv=ffbefc3c
#
```

set [-e] [-f] var=valor Establece el valor de la variable de entorno *var*. Accede mediante el tercer argumento de *main*. En caso de invocarse como set -e hace exactamente los mismo pero utiliza la variable *environ* en lugar del tercer argumento de main para acceder a las variables de entorno. Si la variable de entorno que pretende modificarse no existe no hace nada. La opción -f indica que debe desasignarse (en caso de ser posible) la zona de memoria que ocupaba el valor antiguo de la variable.

put var=valor Establece el valor de la variable de entorno *var*. Usa la función de librería *putenv*. Si la variable no existe la crea.

get var Devuelve los valores de la variable var obtenidos mediante el tercer argumento de *main*, *environ* y la función de librería *getenv*. Para cada caso imprime además la dirección de memoria donde está la variable, y las direcciones donde se almacenan los punteros. Ejemplo:

```
#get TERM
   env:     "TERM=vt100" (ffbeff7b) puntero -> ffbefcd8
   environ: "TERM=vt550" (   2692a) puntero ->    21fb0
   getenv:       "vt550" (   2692f)
#
```

sus [-e] [-f] V1 V2=val Sustituye la variable de entorno V1 por la variable V2=valor. Opera con el tercer argumento de main, salvo que se especifique la opción -e, en cuyo caso utiliza la variable *environ* en lugar del tercer argumento de main para acceder a las variables de entorno. La opción -f indica que debe desasignarse (en caso de ser posible) la zona de memoria que ocupaba el valor antiguo de la variable.

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

search [-a|-d|-n|-p] [dir] Manipula la ruta de búsqueda del intérprete de comandos. La ruta de búsqueda del intérprete de comandos es el conjunto de directorios donde el shell busca los **ejecutables** (Su misión es análoga a la variable

de entorno PATH en el shell del sistema). NO DEBE implementarse con una variable de entorno. Los directorios se especificarán sin el caracter '/' al final Ejemplos

```
#search -a /usr/local/bin /* correctos */
#search -a /
#search -a .
......
#search -a /usr/local/bin/ /*incorrecto*/
#search -a ./              /*incorrecto*/
```

search -a dir Añade dir a la ruta de búsqueda.

search -d dir Elimina dir de la ruta de búsqueda.

search -n Vacía la ruta de búsqueda.

search -p Añade los directorios de la variable de entorno PATH a la ruta de búsqueda.

search dir Indica si dir está en la ruta de búsqueda.

search Muestra la ruta de búsqueda.

whereis ejecutable Busca *ejecutable* 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). Ejemplo

```
#whereis xterm
/usr/openwin/bin/xterm
#
```

getpid Devuelve el pid del proceso y de su proceso padre.

fork El intérprete de comandos crea un hijo y se queda (el propio intérprete de comandos) en espera.

exec 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 (*search*)o bien especificarse la trayectoria completa hasta él (comenzando por /, . o ..). Debe usarse la llamada al sistema *execv*. Ejemplo

```
#exec /usr/bin/ls -l /home  /*ejecuta /usr/bin/ls*/
  ....
```

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

execе LV com Igual que el caso anterior, salvo que ahora en lugar de *execv()* ha de usarse *execve()* LV representa una lista de nombres de variables de entorno, que constituirán el nuevo entorno del proceso; los valores de dichas variables de entorno deben obtenerse de *environ*. Ejemplo

```
#execе TERM HZ HOME DISPLAY xterm -e a.out
```

ejcuta xterm -e a.out en un entorno que sólo contiene las variables de entorno TERM, HZ, HOME y DISPLAY

comando 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 . 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 aniadido */
              /* a la ruta de busqueda */
```

.

LV comando Totalmente análogo al comando execе, 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 .
Ejemplo

```
#TERM HZ HOME DISPLAY xterm -e a.out
```

crea un proceso que ejecuta xterm -e a.out en un entorno que sólo contiene las variables de entorno TERM, HZ, HOME y DISPLAY

com & Análogo a com pero la ejecución es en segundo plano (el shell no espera a que el proceso termine).. Afecta a tanto a *comando* como a

LV comando. Ejemplos

```
TERM HZ HOME DISPLAY xterm -e a.out &
```

crea un proceso que ejecuta xterm -e a.out en un entorno que sólo contiene las variables de entorno TERM, HZ, HOME y DISPLAY. La ejecución es en segundo plano.

```
#xterm &
```

crea un proceso que ejecuta xterm en segundo plano.

procs [-n|-s|-a] Muestra una lista de los procesos que ha mandado este shell en segundo plano. (dicha lista es mantenida por el propio shell). Para cada proceso debe indicar, la línea de comando con que fue lanzado, la hora a la que se inició, y el estado (activo, parado, terminación normal, terminación debido a una seal), asi como el valor devuelto (o la señal en caso de parado o terminado debido a una señal). El listado debe imprimir SOLO UNA LINEA POR CADA PROCESO. Las opciones -n, -s y -a se utilizan para eliminar de la lista los procesos que ya hayan terminado: la opción -n elimina de la lista los procesos que hayan terminado normalmente, la opción -s los que hayan terminado debido a una señal y la opción -a todos los que hayan terminado.

recursiva [-f] n Invoca a la función recursiva n veces. La funcin recursiva recibe como parmetro el número de veces que se tiene que invocar; tiene 3 variables, un array automatico de 200 caracteres, un array esttico de 200 caracteres y un puntero a caracter.

```
char automatico[200];
static char estatico[200];
char * puntero;
```

Esta función debe hacer lo siguiente

1. asignar memoria (mediante malloc) al puntero para 200 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 -f liberar la memoria asignada al puntero.

4. invocarse a si misma con (n-1) como parmetro (si n>0).
5. si no se ha especificado la opción -f liberar la memoria asignada al puntero. Es decir, la opción -f indica si la memoria asignada al puntero debe liberarse antes o despueés de llamarse recursivamente.

direcciones [-f|-v] Imprime las nombres y las direcciones de las funciones que implementan los comandos del shell (-f), de las variables globales (-v). Si no se especifica ni -f ni -v imprime las direcciones de las funciones y de las variables globales.

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

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 automt́ico (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_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 15 ABRIL 2005