SISTEMAS OPERATIVOS II

Tercer curso Ingeniería Técnica de Sistemas. Curso 2005-2006

Práctica 1: Procesos en UNIX. Entorno y Credenciales

Comenzar la codificación de un shell (intérprete de comandos) en UNIX, que se irá completando en suscesivas prácticas. El shell incluirá de momento los siguientes comandos:

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

ruta [-i|-p|-a|-d] [dir] Manipula la ruta de búsqueda de l shell

- -i Inicializa la ruta de búsqueda del shell a vacía.
- -p Incorpora los directorios de la variable en entorno PATH a la ruta de búsqueda.
- -a *ruta -a dir* añade dir a la ruta de búsqueda.
- -d *ruta -d dir* elimina dir de la ruta de búsqueda.
- – *ruta* sin argumentos muestra la ruta de búsqueda del shell.

La ruta de búsqueda del shell es el conjunto de directorios donde el shell busca los ejecutables. El comando *exec* y la ejecución en primer y segundo plano buscan los ejecutables en esta lista de directorios en caso de que no se les especifique la trayectoria completa hasta los mismos. NO COINCIDE NECESARIAMENTE con la variable de entorno PATH (su misión para este shell es análoga a la de la variable de entorno PATH para el shell del sistema). No debe implementarse como una variable de entorno. Inicialmente está vacía. Los directorios deben poder especificarse sin la / al final (p.e. *ruta /usr/bin* en lugar de *ruta /usr/bin/*).

donde com Si el ejecutable *com* reside en elguno de los directorios de la ruta de búsqueda nos indica la trayectoria completa hasta el ejecutable.

fork El shell crea un hijo y despues de crearlo se queda en espera a que el hijo termine.

comando El shell crea un proceso que ejecuta en primer plano el programa especificado por *comando*. *comando* representa un ejecutable con sus parámetros. Debe usarse la llamada al sistema *execv*.

LISTA comando El shell crea un proceso que ejecuta en primer plano el programa especificado por *comando*. *comando* representa un ejecutable con sus parámetros. El nuevo proceso tiene un entorno formado únicamente por las variables de entorno que especificadas en LISTA. Debe usarse la llamada al sistema *execve*, que permite especificar un entorno. LISTA contiene los nombres de las variables de entorno que van a configurar el nuevo entorno, no sus valores, los cuales deben obtenerse de *environ*.

comando & El shell crea un proceso que ejecuta en segundo plano el programa especificado por *comando*. *comando* representa un ejecutable con sus parámetros. Debe usarse la llamada al sistema *execv*.

LISTA comando & El shell crea un proceso que ejecuta en segundo plano el programa especificado por *comando*. *comando* representa un ejecutable con sus parámetros. El nuevo proceso tiene un entorno formado únicamente por las variables de entorno que especificadas en LISTA. Debe usarse la llamada al sistema *execve*, que permite especificar un entorno. LISTA contiene los nombres de las variables de entorno que van a configurar el nuevo entorno, no sus valores, los cuales deben obtenerse de *environ*.

exec com Análogo a *comando* salvo que el shell no crea un nuevo proceso, sino que reemplaza su código por el del programa especificado en *com*. *com* representa un ejecutable con sus parámetros.

exec LISTA com Análogo a *LISTA comando* salvo que el shell no crea un nuevo proceso, sino que reemplaza su código por el del programa especificado en *com*. *com* representa un ejecutable con sus parámetros.

Tanto en la ejecución en primer plano, como en la ejecución en segundo plano, como en la ejecución sin crear proceso, el ejecutable debe residir en uno de los directorios de la ruta de búsqueda del shell o bien especificarse la trayectoria completa hasta él (comenzando por /, . o ..). Tngase en cuenta que ejecutables en el directorio actual se ejecutarán si se añade "." a la ruta de búsqueda o se especifica su nombre comenzando con "./". Ejemplos

```
$ xterm &
/*...... produce error */
$ /usr/openwin/bin/xterm
/*.......no produce error */
$ ruta -a /usr/openwin/bin
$ xterm &
```

```
/*....... ahora no produce error */
$ exec TERM DISPLAY HOME xterm
```

history Muestra una lista histórica de todos los procesos ejecutados en segundo plano por ese shell. Para cada proceso mostrará, EN UNA SOLA LINEA, el pid, el momento en que fue lanzado, información del estado actual (activo, parado o terminado, junto con valor devuelto o señal según corresponda), y la línea de comando con que fue lanzado.

clearhistory Elimina de la lista histórica los procesos ya terminados

pids Indica el pid del proceso (del shell) y de su proceso padre.

uid [nuevouid] Indica las credenciales de usuario del proceso. Indica tanto la real como la efectiva. Tanto para la real como la efectiva debe indicar el valor numérico y el login asociado. Si se suministra parámetro, intentará establecer la credencial efectiva de usuario a *nuevouid*. *nuevouid* puede ser tanto el valor numérico de la credencial como el login asociado.

gid [nuevogid] Indica las credenciales de grupo del proceso. Indica tanto la real como la efectiva. Tanto para la real como la efectiva debe indicar el valor numérico y el nombre del grupo asociado. Si se suministra parámetro, intentará establecer la credencial efectiva de grupo a *nuevogid*. *nuevogid* puede ser tanto el valor numérico de la credencial como el nombre del grupo asociado.

entorno [-m] Muestra la lista de todas las variables de entorno. Para cada variable de entorno muestra, en una misma línea 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]). El acceso será mediante *environ* salvo si se indica el parametro -m en cuyo caso el acceso a las variables será mediante el tercer argumento de main. **ADEMAS** debe mostrase el valor (como punteros) del tercer argumento de main y de *environ* así como las direcciones de memoria donde están almacenados el tercer argumento de main y environ. 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 [-m] var=valor Establece el valor de la variable de entorno *var*. Si se indica -m el acceso a las variables será mediante el tercer argumento de main, en caso contrario mediante *environ*. Si la variable de entorno no existe no la crea.

putenv 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 var1, var2 ... Devuelve los valores de las variables de entorno var1, var2 .. 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 (su valor como puntero) y la dirección donde se almacena el puntero (salvo en el caso de obtenerla con *getenv*.

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

rec n -M [-t*dim*] Invoca a la función recursiva n veces. La función recursiva recibe como parmetro el número de veces que se tiene que invocar, cuando debe liberar la memoria y el tamaño de su array automatico (y de lo que tiene que asignar con *malloc*). Esta función tiene 3 variables, un puntero a caracter un array automatico de *dim* caracteres y un array esttico de 512 caracteres. Si no se especifica *dim* se supone que vale 512.

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

Esta función debe hacer lo siguiente

1. asignar memoria (mediante malloc) al puntero para *dim* 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 M=1 liberar la memoria asignada al puntero.
4. invocarse a si misma con (n-1) como parmetro y el resto de los parámetros sin variación(si n>0).
5. si se ha especificado la opción M=2 liberar la memoria asignada al puntero. Es decir, la opción -M indica si la memoria asignada al puntero debe liberarse antes, después de llamarse recursivamente o no liberarse (M=0) dependiendo del valor de M.

Ejemplo

`$rec 45 -0 -t1000`

llama a la funcion recursiva 45 veces, ésta utliza un array de 1000 bytes, se asigna 1000 bytes para su variable puntero y no libera la memoria.

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

prompt cadena Establece el prompt del shell a cadena.

exit Sale del shell.

Información detallada de las llamadas al sistema y las funciones de la librería debe obternerse con man (exec, chdir, fork, setuid, setgid, getpwent, getgrent ...)

Para que un proceso de un usuario distinto del root pueda ejecutar con éxito la llamada setuid es necesario que ejecute un fichero propiedad de otro usuario que tenga activado el bit *setuid*, lo cual se consigue poniéndole a dicho fichero los permisos 04755. Criterios de seguridad aconsejan que no puedan crearse ficheros *setuid* en sistemas de ficheros exportados por NFS, de ahi que no pueden crearse ficheros *setuid* en los directorios HOME de los usuarios. Hay que crearlos en un directorio local de la máquina donde se trabaja. El único directorio local donde los usuarios normales tienen permiso de escritura es /tmp.

En ningún caso la práctica puede producir error en tiempo de ejecución.

En caso de no poder realizar alguna de las acciones solicitadas el shell debe informar de ello y del motivo.

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 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 2 Diciembre 2005