SISTEMAS OPERATIVOS II

Tercer curso Ingeniería Informática. Curso 2008-2009

Práctica 2: Procesos en UNIX. Ejecución en esgundo plano, credenciales, prioridad, redirección.

Continuar la codificación de un intérprete de comandos (shell) en UNIX, que se irá completando en suscesivas prácticas.

- 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.

Nuestro shell debe mantener una nueva lista:

- *Lista de procesos en segundo plano.* Lista los procesos en segundo plano lanzados desde el shell. El comando *jobs* mostrará esa lista, y el comando *jobs -d* permitirá eliminar procesos de esta lista. Para cada proceso se mostrará (en una línea):
  - su pid
  - el instante en que comenzó su ejecución
  - la linea de comando que está ejecutando

- su estado (activo, parado, terminado o terminado por señal) asi como el valor devuelto en el caso de haber terminado o la señal que lo terminó, en el caso de haber terminado por señal
- la prioridad que tiene en ese instante

getpriority [pid] Muestra la prioridad del proceso *pid*. Si no se especifica pid, muestra la prioridad del intérprete de comandos.

setpriority [pid] valor Estable la prioridad del proceso *pid*. Si no se especifica pid, establece la prioridad del intérprete de comandos.

uid [-n] [valor] En caso de no indicar ni *-n* ni *valor* muestra las credenciales de usuario del proceso; muestra la real y la efectiva, tanto su valor numérico como el login asociado. Si se especifica *valor* intenta establecer la credencial efectiva del proceso a *valor*; *-n* indica que se suministra el valor numérico de la credencial, en otro caso valor representa el login.

back [LISTAVAR] cmd [args] Ejecuta, creando un proceso en segundo plano, el programa especificado por cmd con los argumentos especificados por args. Si no se especifica LISTAVAR la ejecución será mediante *execv*. LISTAVAR representa una lista de nombres de variables de entorno, y en caso de especificarse la ejecución será mediante *execve* (los valores se obtendrán mediante *environ*. Si LISTAVAR es "NULLENV" (sin las comillas) la ejecicuón será mediante *execve* pero en un entorno vacio. Si *cmd* es un nombre que comienza por /, ./ o ../ debe entenderse como el nombre absoluto del ejecutable, en otro caso se buscará dicho ejecutable en la lista de directorios con la que el shell implementa el concepto de *path*. Debe añadir el proceso creado a la lista de procesos en segundo plano. Si no se especifica ni lista de variables, ni ejecutable ni argumentos es decir, *back* unicamente, mostrará la lista de procesos en segundo plano, igual que *jobs*
ejemplo

```
# back /usr/bin/xterm -e csh
# back TERM HOME DISPLAY HZ /usr/bin/xterm -e bash -T prueba
```

jobs [-d] [[-term] [-sign] [-stop] [-actv] |[pid1 [pid2]...] Con la opción -d, elimina procesos de la lista de procesos en segundo plano; sin -d muestra los procesos en segundo plano con pids *pid1*, *pid2* .... Alternativamente, en lugar de especificar los pids puede usarse

-term todos los que ha terminado normalmente

-sign todos los que han terminado debido a una señal

-stop todos los que están parados

-actv 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 es todos
La lista 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 proceso
- instante de comienzo del proceso
- línea de comando que se está ejecutando

Si no se especifica sobre que conjunto de procesos quiere actuarse (bien mediante lista de pids o bien mediante las opciones -term, -stop ...) se listarán todos los procesos en segundo plano, es decir, tanto *jobs* como *jobs -d* LISTAN TODOS los procesos en esgundo plano.

- **pri valor [back|exec][LISTAVAR] cmd [args]** Ejecuta, (sin crear proceso creando proceso en primer o segundo plano, dependiendo de si se indica *back* o *exec*) en programa especificado por *cmd [args]* con su prioridad cambiada a valor.
- **redirección** Se podrá redireccionar la entrada, salida y/o error estandar de un proceso, tanto de los ejecutados en primer plano como de los ejecutados en segundo plano o de los ejecutados mediante exec. Las redirecciones han de ser compatibles entre si y la sintaxis es
- **[pri val] [back|exec][LISTAVAR] cmd [args] [*If1] [*Of2] [*Ef3]** donde
  - **If1*, si está presente, indica que la entrada estándar debe redirigirse al fichero *f1*
  - **Of2*, si está presente, indica que la salida estándar debe redirigirse al fichero *f2*

– **Ef3*, si está presente, indica que la entrada estándar debe redirigirse al fichero *f3*

ejemplos

```
#gcc jdsdksh sjkdh skdj skdhskjdh skjdh *Esalida_error
#exec ls -l /usr/local %Olista_licheros
```

pipe **cmd1 [args1] % cmd2 [args2]** Crea dos procesos, uno que ejecuta *cmd1* con los argumentos *args1* y otro que ejecuta *cmd2* con los argumentos *args2* de manera que la salida estándar del proceso que ejecuta *cmd1* se redirecciona a la entrada estándar del proceso que ejecuta *args2*

Información detallada de las llamadas al sistema y las funciones de la librería debe obternerse con man (exec, fork, strtok, waitpid, getpriority, setpriority, dup, setuid, getuid getpwent)

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 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 JUEVES 30 ABRIL 2009