SISTEMAS OPERATIVOS II

Tercer curso Ingeniería Técnica Informática Sistemas . Curso 2011-2012

Práctica 2: Procesos en Unix: Ejecución, entorno, credenciales y prioridad

Continuar la codificación de un intérprete de comandos (shell) en UNIX. Nótese que los comandos aquí descritos deben interpretarse de la siguiente manera

- Los argumentos entre corchetes [] son opcionales.
- Los argumentos separados por `|` indican que debe ir uno u otro, pero no ambos simultaneamente.
- El intérprete de comandos debe aceptar y entender la sintaxis aquí propuesta, pero no tiene que forzarla. (por ejemplo, si hay varios argumentos deben aceptarse en el orden especificado, pero puede resultar mas cómodo de programar asumiendo que pueden ir en cualquier orden)

Además deben tenerse en cuenta las siguientes indicaciones

- **En ningún caso debe producir un error de ejecución (segmentation, bus error ...)**. La práctica que produzca un error en tiempo de ejecución no será puntuada. Excepcionalmente se admitirá un error en tiempo de ejecución en algunos comandos: en estos casos se indicará explicitamente (***)
- No debe dilapidar memoria (ejemplo: variable que se asigna cada vez que se llama a una función y no se libera). **NO SE REFIERE A DECLARAR LOS ARRAYS DE TAMAÑO PEQUEÑO**
- 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).
- Las direcciones de memoria deben mostrase en **hexadecimal**.
- La información que se nuestra en pantalla no debe incluir en ningun caso líneas en blanco (ni líneas de '*' ni de '=',...).
- 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.

El shell debe llevar una lista de directorios (análoga al PATH del *bash*) donde busca los ejecutables, de manera que cuando se indica un orden que no es un comando interno del shell, el shell la ejecutará si

- la orden representa la trayectoria completa a un ejecutable, comenzando por "/", "./" o "../"
- la orden es el nombre de un ejecutable que está en uno de los directorios de la lista antes citada.

La implementación de dicha lista es libre pero NO DEBE implementarse como una variable de entorno. El comando del shell *ruta* muestra y/o manipula dicha lista. Los directorios, salvo el raiz ("/") se especificarán sin la barra al final (/usr/bin y no /usr/bin/)

En está práctica, además, el shell llevará una lista de todos los procesos en segundo plano que se lanzan desde él; el comando *listarproc* muestra dicha lista y el comando *borrarproc* la manipula. La implementación de dicha lista es libre

Comandos a implementar en esta práctica

fork El shell crea un proceso hijo mediante la llamada al sistema *fork* y se queda en espera a que dicho proceso hijo termine

cambiarpri **[pid] [valor]** Establece la prioridad del proceso *pid* a valor. Si no se especifica *valor* muestra la prioridad del proceso *pid* y si no se especifica *pid* muestra la prioridad del intérprete de comandos. Cuando muestre prioridades debe mostrar la prioridad obtenida *getpriority* y también la política de planificación a la que pertenece el proceso obtenida mediante *sched_getscheduler*

schedpri **pol pri [pid]** Pone al proceso *pid* en la política de planificación *pol* (OTHER, BATCH, IDLE, RR o FIFO) con una prioridad *pri*. Si no se especifica *pid* lo hace para el intérprete de comandos

mostraruid Muestra las credenciales de usuario del proceso. Para cada credencial muestra el número, el login asociado y en nombre real del usuario

cambiaruid [-l] [uid] Cambia la credencial efectiva de usuario del proceso a uid. Si se especifica -l, uid representa un login, en caso contrario es el valor numérico de la credencial. Si no se especifica ni -l ni uid, *cambiaruid* es equivalente a *mostraruid*

entorno **[-set|-main] [arg1...]** Muestra o modifica el entorno del shell

- **entorno VAR1 VAR2 ....** Muestra los valores de las variables de entorno *VAR1, VAR2 ....* Para cada variable muestra:
  - **accediendo mediante el tercer argumento de *main*:** Su valor como cadena, su valor como puntero (la dirección de memoria donde se almacena la cadena) y la dirección de memoria donde se almacena el puntero

  * **accediendo mediante *environ*:** Su valor como cadena, su valor como puntero (la dirección de memoria donde se almacena la cadena) y la dirección de memoria donde se almacena el puntero
  * **accediendo mediante la función de librería *getenv*:** Su valor como cadena y su valor como puntero.

```
-> entorno  TERM HOME
mediante arg main 0xbf9a6810--> TERM=xterm (0xbf9a69f2)
mediante environ 0xbf9a6810--> TERM=xterm (0xbf9a69f2)
mediante getenv -->xterm (0xbf9a69f7)
mediante arg main 0xbf9a6864--> HOME=/home/antonio (0xbf9a6ec8)
mediante environ 0xbf9a6864--> HOME=/home/antonio (0xbf9a6ec8)
mediante getenv -->/home/antonio (0xbf9a6ecd)
->
```

- **entorno -set [main|environ|putenv] var=nuevovalor . . .** Cambia el valor de la variable de entorno *var* a *nuevovalor*. main indica acceso mediante el tercer argumento de *main*, environ mediante *environ* y putenv mediante la función de librería *putenv*. Si la variable var no existe no la creará, a no ser que se especifique putenv. En este caso *putenv* creará la variable automaticamente si no existe

```
->entorno -set putenv TERM=xterm
```

- **entorno -main** Muestra TODO el entorno del proceso. Se accede mediante el tercer argumento de *main*
- **entorno** Muestra TODO el entorno del proceso. Se accede mediante *environ*. Para cada variable de entorno mostrará
  * La variable y su valor
  * la direccion de memoria donde se almacena la variable
  * La direccion de memoria donde se almacena el puntero
  * ejemplo, si la variable fuese env[i] se podría hacer así

```
printf ("%p--> %s(%p)\n", &env[i],env[i],env[i]);
```

Estas dos úlimas formas del comando *entorno* (*entorno -main* y *entorno*) **ADEMAS DE MOSTRAR LAS VARIABLES mostrarán los valores (como puntero) del tercer argumento de *main* y *environ* y las direcciónes donde se almacenan**

```
->entorno
........
0xbfaa6388->environ[3]=(0xbfaa7716) TERM=xterm
0xbfaa638c->environ[4]=(0xbfaa7721) SHELL=/bin/bash
```

```
........
0xbfaa6404->environ[34]=(0xbfaa7fde) _=./entorno.out
0x8049854->environ=0xbfaa637c
0xbfaa62f8->env=0xbfaa637c
```

ruta [**arg**] Accede a la lista de directorios de búsqueda del shell (equivalente al PATH del sistema)

- **ruta** Muestra la lista de directorios
- **ruta +dir** Añade el directorio *dir* a la lista de directorios. Si no se especifica *dir* muestra la lista de directorios
- **ruta -dir** Elimina el directorio *dir* de la lista de directorios. Si no se especifica *dir* muestra la lista de directorios
- **ruta :anula** Vacia la lista de directorios
- **ruta importapath** Añade los directorios de la variable de entorno PATH a la lista de dirwctorios del shell
- **ruta ejec** Nuestra la trayectoria completa al fichero *ejec* si este está en uno de los directorios de la lista Es análogo al *which* del sistema.

ejecutar [LISTAVARIABLES] **prog arg1 ... @pri** Ejecuta, sin crear proceso (es decir REEMPLAZANDO el código del shell), el programa *prog* con sus argumentos. *prog* representa un ejecutable externo y para poder ser encontrado debe especificarse una trayectoria completa hasta él (comenzando por "/", "./" o "../") o residir en uno de los directorios del *ruta*. Debe usarse la llamada *execv*. Si se especifica LISTAVARIABLES, la ejecución será mediante *execve* y el entorno se especifica en LISTAVARIABLES. LISTAVARIABLES puede contener nombres de variables de entorno (el valor se obtiene de environ) o cadenas de la forma NOMBRE=VALOR (se suministra ya la variable con su valor, y la variable no tiene que existir previamente). Si LISTAVARIABLES es "NULLENV" la ejecución será mediante *execve* y con un entorno vacío (ver ejemplos). Si se indica @pri, el programa se ejecutará con su prioridad establecida previamente a *pri*

segundoplano [LISTAVARIABLES] **prog arg1 ... @pri** El shell crea un proceso que ejecuta en segundo plano el programa *prog* con sus argumentos. *prog* representa un ejecutable externo y para poder ser encontrado debe especificarse una trayectoria completa hasta él (comenzando por "/", "./" o "../") o residir en uno de los directorios del *ruta*. Debe usarse la llamada *execv*. Si se especifica LISTAVARIABLES, la ejecución será mediante *execve* y el entorno se especifica en LISTAVARIABLES. LISTAVARIABLES puede contener nombres de variables de entorno (el valor se obtiene de environ) o cadenas de la forma NOMBRE=VALOR (se suministra ya la variable con su valor, y la variable no tiene que existir previamente). Si LISTAVARIABLES es "NULLENV" la ejecución será mediante *execve* y con un entorno vacío. Si se indica @pri, el programa se ejecutará con su prioridad establecida previamente a *pri*. El proceso se añadirá a la lista de procesos en segundo plano dl shell

[LISTAVARIABLES] **prog arg1 . . . @pri** El shell crea un proceso que ejecuta en primer plano el programa *prog* con sus argumentos. *prog* representa un ejecutable externo y para poder ser encontrado debe especificarse una trayectoria completa hasta él (comenzando por "/", "./" o "../") o residir en uno de los directorios del *ruta*. Debe usarse la llamada *execv*. Si se especifica LISTAVARIABLES, la ejecución será mediante *execve* y el entorno se especifica en LISTAVARIABLES. LISTAVARIABLES puede contener nombres de variables de entorno (el valor se obtiene de environ) o cadenas de la forma NOMBRE=VALOR (se suministra ya la variable con su valor, y la variable no tiene que existir previamente). Si LISTAVARIABLES es "NULLENV" la ejecución será mediante *execve* y con un entorno vacío. **NO DEBE ESCRIBIRSE "prog" PARA EJECUTAR ALGO. "prog" REPRESENTA EL NOMBRE DEL EJECUTABLE**. Si se indica @pri, el programa se ejecutará con su prioridad establecida previamente a *pri*
Ejemplos

```
*) ruta +/bin
*) ruta -/usr/bin
*) ruta  +.
*) segundoplano /usr/bin/xclock
*) segundoplano TERM HOME DISPLAY USER xterm -e bash @15
*) xclock -digital
*) ejecutar NULLENV ./a.out
```

listarproc [**all|term|sig|stop|act**] Muestra la lista de procesos en segundo plano. Para cada proceso debe mostrar (en una sola línea) su pid, su prioridad, la linea de comando que ejecuta, el instante de inicio y su estado (activo, terminado normalmente, parado o terminado por señal) indicado, en su caso, el valor devuelto o la señal causante de su terminación o parada. Si no se indican argumentos suficientes (p.e. *listarproc* sin argumentos) se listarán todos.

- **listarproc all** Muestra todos los procesos
- **listarproc term** Muestra los procesos terminados normalmente
- **listarproc sig** Muestra los procesos terminados por señal
- **listarproc stop** Muestra los prcesos parados
- **listarproc act** Muestra los procesos activos

borrarprocs [**all|term|sig|stop|act**] Elimina de la lista de procesos en segundo plano los procesos que se le especifican. (Es un comando de manipulación de la lista de procesos en segundo plano, no tiene que terminar los procesos). Si no se indican argumentos suficientes (p.e. *borrarproc* sin argumentos) se listarán todos.

- **borrarproc all** Vacia la lista
- **borrarproc term** Elimina de la lista los procesos terminados normalmente
- **borrarproc sig** Elimina de la lista los procesos terminados por señal
- **borrarproc stop** Elimina de la lista los prcesos parados
- **borrarproc act** Elimina de la lista los procesos activos

Información detallada de las llamadas al sistema y las funciones de la librería debe obternerse con man (waitpid, exec, execve, putenv, getenv, setpriority, sched_setscheduler, 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 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_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. 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.
7. No debe incluirse la letra ñ ni vocales acentuadas en los nombres

FECHA DE ENTREGA Viernes 2 Diciembre de 2012