SISTEMAS OPERATIVOS II

Tercer curso Ingeniería Informática. Curso 2011-2012

Práctica 1: Procesos en Unix: Entorno

Comenzar 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 al shell la ejecutará si

- la orden es un comando interno del shell (p.e. *autores, getpid ...*)
- 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 (lista anĺoga al PATH).

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

Además el shell llevará control (mediante una lista) de los procesos que se lanzan desde él en segundo plano. Para cada proceso ha de poder informarnos de su pid, instante de creación, línea de comando que está ejecutando, y estado (activo, parado, terminado normalmente o terminado por señal, y en el caso de los terminados, del valor devuelto, o la señal que lo ha terminado). El comando *jobs* nos permite acceder o modificar dicha lista

Comandos a implementar en esta práctica

fin Termina la ejecución del intérprete de comandos.

exit Termina la ejecución del intérprete de comandos.

quit Termina la ejecución del intérprete de comandos.

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

pid Muestra el pid del proceso que ejecuta el shell y del proceso padre del shell

pwd Muestra el directorio actual del shell

cd dir Cambia el directorio actual del shell a *dir*. Si no se especifica *dir*, muestra el directorio actual del shell (igual que *pwd*)

path [**-add|-del|-find|-list|-clear|-path**] [**arg**]

- **path** Muestra la lista de directorios donde el shell busca los ejecutables
- **path -add** [**arg**] Añade el directorio *arg* a la lista de directorios. Si no se especifica *arg* muestra la lista de directorios
- **path -del** [**arg**] Elimina el directorio *arg* de la lista de directorios. Si no se especifica *arg* muestra la lista de directorios
- **path -find** [**arg**] Nuestra la trayectoria completa al fichero *arg* si este está en uno de los directorios de la lista. Si no se especifica *arg* muestra la lista de directorios. (TOTALMENTE análogo al comando *which* del sistema)

- **path -clear** Vacia la lista de directorios
- **path -list** Muestra la lista de directorios
- **path -path** Añade los directorios de la variable de entorno PATH a la lista de directorios del shell

environ [-mode|-sus|-putenv|-addr] [arg1. . .] Muestra o modifica el entorno del shell

- **environ 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.

```
-> environ 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)
->
```

- **environ -mode [main|environ]**. Establece el método de acceso a las variables de entorno para su modificación. *main* indica que el acceso será mediante el tercer argumento de main y *environ* indica que el acceso será mediante *environ*. Si no se especifica *main* o *environ*, `environ -mode` nos indica cómo es el acceso. Por defecto es mediante *environ*
- **environ var=nuevovalor var2=nuevovalor2 . . .** Cambia el valor de la variable de entorno *var* a *nuevovalor*, el de la variable de entorno *var2* a *nuevovalor2 . . . .* El acceso será mediante el tercer argumento de main o environ segun se haya establecido con *environ -mode*

```
->environ TERM=xterm DISPLAY=localhost:0.0
```

- **environ -sus var nuevavar=nuevovalor** Sustituye la variable de entorno *var* por la variable *nuevavar* con el valor *nuevovalor*. El acceso será mediante el tercer argumento de main o environ segun se haya establecido con *environ -mode*

```
->environ -sus TERM TERMINAL=xterm
```

– **environ -putenv var=nuevovalor var2=nuevovalor2 . . .** Cambia el valor de la variable de entorno *var* a *nuevovalor*, el de la variable de entorno *var2* a *nuevovalor2* . . . . Lo hace utilizando la función de librería *putenv*

```
->environ -putenv TERM=xterm DISPLAY=localhost:0.0
```

– **environ** Muestra TODO el entorno del proceso. El acceso será mediante el tercer argumento de main o environ segun se haya establecido con *environ -mode*. Para cada variable muestra

  - ∗ 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]);
    ```

```
->environ
........
0xbfaa6388->environ[3]=(0xbfaa7716) TERM=xterm
0xbfaa638c->environ[4]=(0xbfaa7721) SHELL=/bin/bash
........
0xbfaa6404->environ[34]=(0xbfaa7fde) _=./entorno.out
```

– **environ -addr** Muestra los valores (como puntero) del tercer argumento de *main* y *environ* y las direcciónes donde se almacenan.

```
0x8049854->environ=0xbfaa637c
0xbfaa62f8->env=0xbfaa637c
```

fork El shell crea un hijo y se queda en espera a que ese hijo termine. (El hijo continua ejecutando el código del shell)

execute prog arg1 . . . 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 puede especificarse una trayectoria completa hasta él (comenzando por "/", "./" o "../") o residir en uno de los directorios del la lista (*path*) del shell. Debe usarse la llamada *execv*

execute prog arg1 . . . ** LISTAVARIABLES Análogo al comando *execute* anterior salvo que ahora la ejecución es mediante *execve* y el entorno se especifica en *LISTAVARIABLES*. *LISTAVARIABLES* es la lista de variables que conforman el nuevo entorno del proceso. Para ello la ejecución será memdiante *execve*. *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 vacía (y se ha indicado el argumento **), la ejecución será mediante *execve* pero con un entorno vacío.

prog arg1 ... 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 puede especificarse una trayectoria completa hasta él (comenzando por "/", "./" o "../") o residir en uno de los directorios de la lista (*path*) del shell. Debe usarse la llamada *execv*

prog arg1 ... ** LISTAVARIABLES Análogo al comando anterior salvo que ahora la ejecución (creando proceso en primer plano) es mediante *execve* y el entorno se especifica en *LISTAVARIABLES*. *LISTAVARIABLES* es la lista de variables que conforman el nuevo entorno del proceso. Para ello la ejecución será memdiante *execve*. *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 vacía (y se ha indicado el argumento **), la ejecución será mediante *execve* pero con un entorno vacío.

background prog arg1 ... 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 puede especificarse una trayectoria completa hasta él (comenzando por "/", "./" o "../") o residir en uno de los directorios de la lista (*path*) del shell. Debe usarse la llamada *execv*. **Además añadirá el proceso a la lista de procesos en segundo plano del shell**

background prog arg1... ** LISTAVARIABLES Análogo al comando anterior salvo que ahora la ejecución (creando proceso en **segundo** plano) es mediante *execve* y el entorno se especifica en *LISTAVARIABLES*. *LISTAVARIABLES* es la lista de variables que conforman el nuevo entorno del proceso. Para ello la ejecución será memdiante *execve*. *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 vacía (y se ha indicado el argumento **), la ejecución será mediante *execve* pero con un entorno vacío. **Además añadirá el proceso a la lista de procesos en segundo plano del shell**
ejemplos

```
-> path -add /usr/bin
-> path -add /bin
-> execute ./a.out
-> execute ./a.out ** TERM HOME DISPLAY NUEVA=nada USER
-> /usr/bin/xterm -e csh
-> xterm -e csh ** TERM HOME DISPLAY USER NUEVAVAR=desconocida
-> ls -l /home
-> background  xterm -e csh ** TERM HOME DISPLAY USER NUEVAVAR=desconocida
-> background /usr/bin/xterm -e csh
```

jobs [**-del**] [**all|term|sig|stop|act**] Muestra (o manipula) la lista de procesos en segundo plano del shell.

- **jobs** [**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. *procesos* sin argumentos) se listarán todos.
  - **jobs all** Muestra todos los procesos
  - **jobs term** Muestra los procesos terminados normalmente
  - **jobs sig** Muestra los procesos terminados por señal
  - **jobs stop** Muestra los prcesos parados
  - **jobs act** Muestra los procesos activos
- **jobs -del** 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. *procesos -del* sin argumentos) se listarán todos.
  - **jobs -del all** Vacia la lista
  - **jobs -del term** Elimina de la lista los procesos terminados normalmente
  - **jobs -del sig** Elimina de la lista los procesos terminados por señal
  - **jobs -del stop** Elimina de la lista los prcesos parados
  - **jobs -del act** Elimina de la lista los procesos activos
- **jobs pid** Muestra la información del proceso de pid *pid*. Además de mostrar la información que figura en la lista, en el caso de que el proceso haya finalizado nos da tambien la información de uso de recursos por parte del proceso (`struct rusage`). Debe usarse

por tanto la llamda *wait4*

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

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 automtico (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 23 Marzo de 2010