SISTEMAS OPERATIVOS II

Tercer curso Ingeniería Informática. Curso 2009-2010

Práctica 3: Procesos en Unix: Prioridad, credenciales y entorno

Continuar la codificación de un intérprete de comandos (shell) en UNIX, la funcionalidad de la presente practica se AÑADIRÁ a la de la práctica anterior. Como en la práctica anterior 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.

Comandos a implementar en esta práctica

priority [**-get**|**-set**] [**arg...**] Muestra o establece la prioridad de un proceso o del propio shell

- **priority -get** [**pid**] Muestra la prioridad del proceso *pid*. Si no se especifica *pid* muestra la prioridad del intérprete de comandos
- **priority -set valor** [**pid**] Establece la prioridad del proceso *pid* a valor. Si no se especifica *pid* muestra la prioridad del intérprete de comandos

uid [**-get**|**-set**] [**arg...**] Muestra o establece la credencial del shell

- **uid -get** Muestra las credenciales de usuario (real y efectiva) del proceso. Muestra tanto el número (uid) como el login (nombre) asociado
- **uid -set cred** Establece (si puede) la credencial efectiva de usuario del proceso. *cred* puede representar un número o un login (nombre) de usuario

environ [**-get**|**-set**|**-sus**|**-env**] [**arg1...**] Muestra o modifica el entorno del shell

- **environ -get 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 -get  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 -set** [**-a**|**-e**|**-p**] **var nuevovalor ...** Cambia el valor de la variable de entorno *var* a *nuevovalor*.-a indica acceso mediante el tercer argumento de *main*, -e mediante *environ* y -p mediante

la función de librería *putenv*. Si la variable var no existe no la creará, a no ser que se especifique -p. En este caso *putenv* creará la variable automaticamente si no existe

```
->environ -set -p TERM xterm
```

- **environ -sus [-a|-e] var nuevavar=nuevovalor . . .** Sustituye la variable de entorno *var* por la variable *nuevavar* con el valor *nuevovalor*.-a indica acceso mediante el tercer argumento de *main* y -e mediante *environ*.

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

- **environ -env [-a|-e]** Muestra TODO el entorno del proceso. -a indica que se acceda mediante el tercer argumento de *main* y -e 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]);
    ```

  Además mostrará los valores (como puntero) del tercer argumento de *main* y *environ* y las direcciónes donde se almacenan.

  ```
  ->environ -env -e
  ........
  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
  ```

- Modificar la ejecución en primer plano, segundo plano y sin crear proceso para que permita además especificar un entorno alternativo y/o un cambio en la prioridad. La sintaxis es

[**exec**] [**LISTAVAR**] **prog** [**args. . .**] [**@pri**] [**&**]. Donde

- *exec* (opcional) indica si la ejecución es sin crear proceso
- *LISTAVAR* (opcional) es la lista de variables que conforman el nuevo entorno del proceso. Para ello la ejecución sería memdiante *execve*. Si no se especifica *LISTAVAR* la ejecución será mediante *execv*. LISTAVAR 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 LISTAVAR

es "EMPTY" se entiende que se quiere ejecutar con un entorno vacío.

- *prog* El ejecutable que se quiere ejecutar. Todas las consideraciones de la práctica 1 sobre la ubicación de los ejecutables, son de aplicación aquí
- *args...* (opcional) Los argumentos al ejecutable. NO PUEDE SUPONERSE QUE HAY SOLO UNO
- *@pri* (opcional) Indica que el programa debe ejecutarse con una prioridad *pri*. Úsese *setpriority* para establecerla
- *&* (opcional) Indica que la ejecución es en segundo plano. Todas las consideraciones de la práctica 1 sobre la lista de procesos en segundo plano, son de aplicación aquí

```
--> exec TERM USE GDM_LANG LOGNAME ./a.out
--> TERM HOME DISPLAY USER NUEVA=nuevavar LANG xterm -e csh @5 &
--> EMPTY a.out
```

Información detallada de las llamadas al sistema y las funciones de la librería debe obternerse con man (execve, getenv, putenv, setuid, getuid, getpwent...)

FORMA DE ENTREGA Como en practicas anteriores

FECHA DE ENTREGA VIERNES 4 JUNIO DE 2010