SISTEMAS OPERATIVOS II
Tercer curso Ingeniería Informática
Curso 2006-2007

Práctica 3: Procesos y sistema de ficheros en UNIX: Entorno y redirección. Redirección. Añadir al intérprete de comandos de las prácticas anteriores las siguientes funciones de manipulación del entorno. Al igual que en otras 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.

entorno [-e |-a |-d] 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 de la variable e[i] como puntero) y la dirección de memoria donde se almacena el puntero (&e[i]). El acceso será mediante la variable *environ* (-e) o mediante el tercer argumento de main (-a).
entorno -d muestra 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. Si no se especifica argumento, mostrará TODO (todas las variables accediendo de las dos manejras y ADEMAS las derecciones y el valor como punteros de *environ y el tercer argumento de main*) Ejemplo

```
#entorno -e
......
```

```
.......
.......
ffbefcb8->environ[29]=(ffbeff86) PATH=/usr/bin:/usr/local/bin:
ffbefcbc->envviron[30]=(ffbeffa5) LC_MONETARY=es
ffbefcc0->environ[31]=(ffbeffb4) LC_COLLATE=es
ffbefcc4->environ[32]=(ffbeffc2) _INIT_NET_STRATEGY=none
#entorno -d
20a34->environ=ffbefc44 | ffbefc24->env=ffbefc44  | ffbefc20->argv=ffbefc3c
# entorno
......
.......
.......
ffbefcb8->environ[29]=(ffbeff86) PATH=/usr/bin:/usr/local/bin:
ffbefcbc->envviron[30]=(ffbeffa5) LC_MONETARY=es
ffbefcc0->environ[31]=(ffbeffb4) LC_COLLATE=es
ffbefcc4->environ[32]=(ffbeffc2) _INIT_NET_STRATEGY=none
20a34->environ=ffbefc44 | ffbefc24->env=ffbefc44  | ffbefc20->argv=ffbefc3
#
```

- **env [-e |-a |-p |-s |-c] [arg [arg2] ...]** Permite ver, modificar y crear variables de entorno

  env Sin argumentos muestra el entorno (equivalente al comando *entorno*)

env 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*).

```
#env HOME TERM
mediante arg main 0xbfffff41: HOME=/home/antonio (&0xbffffaf4)
mediante environ 0xbfffff41: HOME=/home/antonio (&0xbffffaf4)
mediante getenv 0xbfffff46->/home/antonio
mediante arg main 0xbffffbd8: TERM=xterm (&0xbffffabc)
mediante environ 0xbffffbd8: TERM=xterm (&0xbffffabc)
mediante getenv 0xbffffbdd->xterm
#
```

env -e|-a|-p var valor Establece el valor de la variable de entorno *var* a *valor* accediendo a ella mediante *environ* (-e), el tercer argumento de main (-a) o la función de librería *putenv*. Si la variable no existe no la creará (informará de ello), salvo que se haya especificado que se haga mediante *puntenv*, entonces será creada.

env -s|-c var nue=val Sustituye de la variable de entorno *var* por *nue=val* accediendo a ella mediante *environ* (-s) o el tercer argumento de main (-c). Si la variable no existe informará de ello.

- Modificar la ejecución en primer plano, en segundo plano y la ejecución sin crear proceso para que permita la redirección de la entrada estándar, la salida estándar y/o el error estándar y ademas la ejecucion con un entorno reducido (usando *execve*)
  - **[exec] [LISTA] com <fich [@pri] [&]** Ejecuta (en primer o segundo plano, o sin crear proceso, tanto especificándole el entorno como sin hacerlo) el comando *com* (ejecutable con parámetros) con la entrada estándar redireccionada al fichero *fich*.
  - **[exec] [LISTA] com >fich [@pri] [&]** Ejecuta (en primer o segundo plano, o sin crear proceso, tanto especificándole el entorno como sin hacerlo) el comando *com* (ejecutable con parámetros) con la salida estándar redireccionada al fichero *fich*.
  - **[exec] [LISTA] com #fich [@pri] [&]** Ejecuta (en primer o segundo plano, o sin crear proceso, tanto especificándole el entorno como sin hacerlo) el comando *com* (ejecutable con parámetros) con el error estándar redireccionado al fichero *fich*.
  - *LISTA* Representa una lista de variables de entorno que constituirán el nuevo entorno en el que se ejecutará el programa especificado por *com*: la lista contiene nombres de variables de entorno (su valor ha de obtenerse de *environ*) o cadenas de la forma variable=valor si quiere especificarse una nueva o que alguna cambie de valor (NO DEBE MODIFICARSE EL ENTORNO DEL PROCESO ACTUAL, SOLO AFECTA A LOS PARAMETROS A EXECVE). Si el primer elemento de la lista es la cadena "NULL", el programa se ejecutará con un entorno vacio. Si no se especifica lista procederá como en la prática anterior (ejecutando mediante *execv*)
- **pipe com1 |com2** Ejecuta *com1* (ejecutable con parámetros) redireccionando su salida estándar a la entrada estándar de *com2* (ejecutable con parámetros). Tanto *com1* como *com2* representan ejecutables con parámetros.
  - Los comandos de redirección >, <, # deben ser compatibles entre sí y con la ejecución en segundo plano, la ejecución sin crear proceso y con la ejecución con las prioridades cambiadas, y con la especificación de un entorno. Ejemplos:

    ```
    #TERM HOME DISPLAY NUEVA=nueva  a.out parm1 parm2 <f1 >f2 #f3 @15 &
    ```

    ejecuta en segundo plano *a.out parm1 parm2 parm3* en un entorno que solo contiene las variables TERM HOME DISPLAY y NUEVA, con la entrada, la salida y el error estándar redireccionados, además establece su prioridad a 15.

`#NULL ./a.out`

ejecuta ./a.out en un entorno que NO CONTIENE ninguna variable de entorno

`#pipe ls -lR . . / | wc -l -c`

ejecuta ps -lR . . / y redirecciona la salida a la entrada estándar de un proceso que ejecuta wc -l -c

FORMA DE ENTREGA

Como en las prácticas anteriores

FECHA DE ENTREGA Viernes 1 Junio de 2007