SISTEMAS OPERATIVOS II

Tercer curso Ingeniería Técnica Informática de sistemas. Curso 2008-2009

Práctica 3: Procesos en UNIX: Entorno y prioridades

Continuar la codificación de un intérprete de comandos (shell) en UNIX. Al igual que en la práctica anterior

- 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.

environ [-m|-e] Muestra la lista de todas las variables de entorno accediendo mediante el tercer argumento de *main()* (opcion -m) o mediante la variable *environ* (opcion -e). 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]). Si no se suministra ningún parámetro 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. Ejemplo

```
#environ -m
```

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

get [-m] [-e] [-p] v1 v2... Devuelve los valores de las variables de entorno var1, var2... obtenidos mediante el tercer argumento de *main* (opción -m), la variable *environ* (opción -e) y/o la función de librería *getenv* (opción -p). 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*).

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

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

withenv modificar los comandos **xec, back** y la ejecución en primer plano para que pueda especificarse un entorno alternativo. El formato es *withenv [lista_variables]*, donde *withenv* y lista_variables deben ir tras *back o xec* y antes del nombre del ejecutable. lista_variables puede contener solamente nombres de variables (en este caso su valor se obtiene de *environ*) o cadenas de la forma variable=valor (en este caso la variable no tiene por que existir previamente). En el caso de especificar *withenv* la ejecución será mediante *execve* y el entorno el que se suministra. En el caso de no indicarse *withenv* la ejecución será como en el caso anterior. Si se especifica *withenv* pero la lista de variables está vacia se entenderá que el programa debe ejecutarse en un entorno vacio. Todo lo dicho para ejecutables en la práctica enterior es válido aqui.
[xec|back] [withenv lista_variables] ej [args]
ejemplos

```
# back withenv TERM DISPLAY HOME HZ=120 N=65 xterm -e ksh

# xec withenv TERM DISPLAY HOME HZ=120 N=65 ./a.out arg1 arg2

# withenv TERM DISPLAY HOME USER NUEVA=PRUEBA /usr/bin/xterm -hold -e ./a.out

# withenv ./a.out
```

Las variables no tienen por que ir en mayúsculas, la primera cadena que no sea una variable (y se sabe que es una variable si existe en *environ* o es una cadena en la forma cad1=cad2) se supone que es el ejecutable

getpriority [pid] Muestra la prioridad del proceso de identificador *pid*. Si no se especifica *pid* se entiende que es el proprio intérprete de comandos

setpriority [pid] valor Establece la prioridad de ceso de identificador *pid* a *valor*. Si no se especifica *pid* se entiende que es el proprio intérprete de comandos

@nuevapri Permite ejecutar programas (sin crear proceso o creando proceso en primer o segundo plano, con o sin entorno cambiado) especificando una nueva prioridad.
[**xec|back**] [**withenv lista_variables**] **ej** [**args**] [**@nuevapri**]
ejemplos

```
# back withenv TERM DISPLAY HOME HZ=120 N=65 xterm -e ksh @12

# back  xterm -e ksh @12

# xec withenv TERM DISPLAY HOME HZ=120 N=65 ./a.out arg1 arg2 @-4

# withenv TERM DISPLAY HOME USER NUEVA=PRUEBA /usr/bin/xterm -hold -e ./a.out @7

# ./a.out arg1 arg2 @-15
```

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

FORMA DE ENTREGA Como en prácticas anteriores

FECHA DE ENTREGA Viernes 12 Diciembre 2008