SISTEMAS OPERATIVOS II

Tercer curso Ingeniería Técnica de Sistemas. Curso 2009-2010

Práctica 2: Procesos en UNIX. Ejecucion de programas y entorno

Continuar la codificación del 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 más 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 (por ejemplo, la función recursiva que desborda la pila), 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.

EJECUCION DE PROGRAMAS:

Cuando al shell se le indique algo que no es una función interna (los coman-

dos del shell que se van añadiendo en las prácticas, como autores, getpid...) el shell supondrá que es un ejecutable externo con sus argumentos e intentará ejecutarlo creando un proceso EN PRIMER PLANO que ejecutará dicho programa con sus argumentos (el comando *background* lo hará en segundo plano). Para que dicho ejecutable pueda ser encontrado debe suministrarse al shell la trayectoria completa (comenzando por "/", "./" o "../", p.ej. /usr/bin/ls) o bien dicho ejecutable debe encontrarse en uno de los directorios que que constituyen la *ruta de búsqueda* o *PATH* del shell (p.ej. se indica al shell "ls" y /usr/bin está en el PATH): **El shell debe implementar el concepto de PATH: lista de directorios donde buscará los programas externos; la implementacion de dicha lista es libre. pero NO DEBE implementarse como una variable de entorno; los comandos path-* muestran y o manipulan dicha lista**. Toda la ejecución de programas se hará con *execv* o *execve*, no con *execp* ni *execlp*

Comandos a implementar en esta práctica

displayenv [-e] Muestra el entorno del shell. El acceso es mediante el tercer argumento de *main* a no ser que se especifique -e, que indica que se acceda mediante *environ*. Además de la lista de variables de entorno mostrará el valor (como puntero) y la dirección donde se almacenan el tercer argumento de *main* y *environ*

```
-> displayenv
.......
0xbf9a6880->tercer arg. main[29]=(0xbf9a6fa3) LESSCLOSE=/usr/bin/lesspipe
0xbf9a6884->tercer arg. main[30]=(0xbf9a6fc5) XAUTHORITY=/home/antonio/.Xa
0xbf9a6888->tercer arg. main[31]=(0xbf9a6fea) _=./a.out
environ 0xbf9a680c en direccion 0x80c075c
tercer arg de main 0xbf9a680c en direccion 0xbf9a65f8
->
```

displayenv V1 V2 ... Muestra las variables de entorno V1, V2 .... Para cada variable muestra:

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

```
-> displayenv TERM HOME
mediante arg main 0xbf9a69f2: TERM=xterm (&0xbf9a6810)
```

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

changeenv [-s] [-a|-e|-p]... Cambia una de las variables de entorno del proceso

- **changeenv [-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 (*putenv* creará la variable automaticamente si no existe)

  ```
  ->changeenv -p TERM xterm
  ```

- **changeenv -s [-a|-e] var nuevavar=nuevovalor**. Sustituye la variable de entorno *var* por *nuevavar* con valor *nuevovalor*.-a indica acceso mediante el tercer argumento de *main*, -e mediante *environ*. Si la variable var no existe no existe en el entorno especificado nos informará de ello.

  ```
  ->changeenv -s -a USER USUARIO=PanteraRosa
  ```

fork El shell crea un proceso hijo y él (el shell) se queda en espera a que el hijo creado termine. Podría ser

```
void cmd_fork(void)
{
  pid_t pid;

  if ((pid=fork())==0)
     printf ("proceso hijo %d\n", getpid());
  else
     waitpid(pid,NUL,0);
}
```

path Muestra la lista de directorios que constituyen la ruta de búsqueda.

path-add [dir] Añade *dir* a la lista de directorios que constituyen la ruta de búsqueda. Si no se suministra *dir* muestra la lista de directorios que constituyen la ruta de búsqueda. *dir* se indica sin "/" al final, (a no ser que sea el directoiro raiz "/")

path-del [dir] Elimina *dir* de la lista de directorios que constituyen la ruta de búsqueda. Si no se suministra *dir* muestra la lista de directorios que constituyen la ruta de búsqueda.*dir* se indica sin "/" al final, (a no ser que sea el directoiro raiz "/")

path-query [dir] Indica si *dir* está en la lista de directorios que constituyen la ruta de búsqueda. Si no se suministra *dir* muestra la lista de directorios que constituyen la ruta de búsqueda.*dir* se indica sin "/" al final, (a no ser que sea el directoiro raiz "/")

path-path Añade los directorios de la variable de entorno PATH, a la a ruta de búsqueda de nuestro shell (error muy común: no se tiene en cuenta que *strtok* rompe la cadena que se le pasa como parámetro)

path-init Inicializa la lista de directorios que constituyen la ruta de búsqueda.

path-find [file] Si file comienza por "/", "./" o "../" y *file* existe, nos devuelve *file*. Si file no comienza por "/", "./" o "../" busca *file* en la lista de directorios que constituyen la ruta de búsqueda. Es totalmente análogo al comando *which* del sistema.

prog arg1 arg2 .... Crea un proceso que ejeuta en primer plano el prorama *prog* con sus argumentos. Para poder ser encontrado *prog* debe comanzar por "/", "./" o "../" o residir en uno de los directorios del *PATH*

background *prog arg1 ....* Exactamente igual al caso anterior pero el proceso se creará en segundo plano.

exec *prog arg1 ....* Exactamente igual a los casos anteriores pero sin crear previamente un proceso (el shell reemplaza su código mediante *execv*)

*prog arg1 arg2 ... ** LISTAVARIABLES*. Crea un proceso que ejeuta en primer plano el prorama *prog* con sus argumentos. Para poder ser encontrado *prog* debe comanzar por "/", "./" o "../" o residir en uno de los directorios del *PATH*. La ejecución ahora es 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 está vacia, se entiende que la ejecución es con un entorno vacio.

background *prog* **arg1 ... ** LISTAVARIABLES**. Exactamente igual al caso anterior pero el proceso se creará en segundo plano

exec *prog* **arg1 ... ** LISTAVARIABLES**. Exactamente igual a los casos anteriores pero sin crear previamente un proceso (el shell reemplaza su código mediante *execve*)
Ejemplos

```
-> path-add /bin
-> path-add /usr/bin
-> path-find xterm
/usr/bin/xterm
-> xterm -bg grey -fg yellow
```

```
-> xterm -bg grey -fg yellow ** DISPLAY HOME TERM=xterm USER NUEVAVAR=nuev
-> background /usr/sbin/xterm
-> exec xterm -bg grey -fg yellow ** DISPLAY HOME TERM=xterm USER NUEVAVAR
```

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 11 DICIEMBRE DE 2009