SISTEMAS OPERATIVOS II

Tercer curso Ingeniería Técnica Informática Sistemas . Curso 2009-2010

Práctica 3: Procesos en UNIX: Credenciales, Prioridad, Señales

Continuar la codificación de un intérprete de comandos (shell) en UNIX, que se irá completando en suscesivas prácticas. El shell debe reconocer los comandos que se datallan a continuación, teniendo en cuanta que en dicha descripción:

- 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 ningún 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.

En esta prática se va a proceder a instalar manejadores de señales así como a ignorar y enmascarar señales. El comando *sigaction* permite instalar distintos tipos de manejadores, así como ver como está cada señal. Puede considerarse que el número de señales es 32. El shell llevará unos contadores vara saber el número de veces que se ha ejecutado el manejador de cada señal, de manera que cada vez que se ejecute una de las funciones que es manejador debido a la llegada de una señal, incrementará el contador asociado a dicha señal. Dado que en las máquinas asignadas para prácticas no existen *str2sig* y *sig2tsr* pueden usarse las funciones Senal y NombreSenal disponibles en

`http://www.dc.fi.udc.es/ afyanez/Practicas/sources/senales.c`.

Téngase en cuenta que las opciones `-p, -t, -f` afectan al campo `sa_flags` de la estructura *sigaction* al instalar el manejador y las opciones `-v, -r, -sN` afectan a como se comporta el manejador instalado. Distintas señales han de poder tener instalado un manejador que se comporta de manera distinta (tal vez en este momento estemos pensando en una función para el manejador sin opciones, otra para el de -v, otra para el de -r, otra para el de -s1, otra para el de -s2,..., otra para el de -s99, otra para el de -v -r, otra para el de -v -s1 .., otra para el de -r -s1, ..; total con 400 funciones para manejadores podría llegar, y como hay que considerar 32 señales entonces con $32x400 = 12800$ funciones llegaría..., aunque parece ser que hay quien piensa que puede hacerse con solo una función que no contiene más de 10 líneas...).

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).

sigaction –install|–show|–showncont|–clearcont|–setstack|–showstack

- **sigaction –install [-v] [-r] [-p] [-t] [-f] [-dN] [-mSEN1] [-mSEN2] ... S1 [S2] ...** . Instala un manejador, mediante *sigaction*, para las señales S1, (S2 ...). El manejador incrementará un contador que indica cuantas veces se ha ejecutado para la señal por la que es invocado. Además se admitirán las siguientes opciones

-mSEN El manejador ha de ejecutarse con la señal SEN enmascarada (se añade SEN al miembro sa_mask de la estructura *sigaction*)

-dN El manejador ha de quedar en espera N segundos (con la llamada *sleep*). Debe ser la última instrucción dentro del manejador

-v El manejador debe imprimir en pantalla el nombre de la señal que se ha recibido (la cual está manejando), cuantas veces se ha recibido (el valor de contador) y la dirección de memoria donde está el parámetro que recibe. De no indicarse -v el manejador NO DEBE IMPRIMIR NADA en pantalla.

-r El manejador reenvía al proceso la señal para la cual es manejador. En caso de reenviarse la señal, debe hacerse despues de imprimir en pantalla (en caso de que se imprima) y antes de quedarse en espera (en caso de que se haya especificado -sN)

-t El manejador es temporal se instala con el flag SA_RESETHAND

-p El manejador se ejecuta en la pila alternativa: se instala con el flag SA_ONSTACK

-f El manejador puede ser interrumpido por la propia señal: se instala con el flag SA_NODEFER

Ejemplo

```
-> sigaction --install -r -f -p -d10 -mUSR1 -mHUP INT SEGV
```

Instala, con los flags SA_NODEFER y SA_ONSTACK, un manejador para SIGINT y SIGSEGV. Dicho manejador se ejecuta con SIGUSR1 y SIGHUP enmascaradas (miembro sa_mask). El manejador cunado se ejecute, además de incrementar el contador de veces que se ha recibido la señal, envia la señal al propio proceso y luego se queda en espera 10 segundos.

- **sigaction –show [S1] [S2] . . .** Nos da información del estado de las señales SI, S2 . . . : manejada (con la dirección del menjador, los flags y la máscara asociada), ignorada o acción por defecto, asi como de si está enmascarada o no. Si no se especifican señales nos muestra la información de todas de todas. Ejemplo

```
#sigacion --show INT HUP SEGV
INT Enmacarada manejador 0x30045a00 SA_RESTART SA_NODEFER. mascara asocia
HUP No enmascarada Accion por defecto
SEGV No emmascarada Ignorada
```

- **sigaction –showncont [S1] [S2] . . .** Nos informa de los contadores de los manejadores de las señales S1, S2 . . . . Si no se especifican señales nos muestra los contadores de todas.
- **sigaction –clearcont [S1] [S2] . . .** Pone a cero los contadores de los manejadores de las señales S1, S2 . . . . Si no se especifican señales pone a cero los contadores de todas.

- **sigaction –setstack *tam* [*dir*]** Establece una pila alternativa de tamaño *tam* para la ejecución de las señales en la dirección de memoria *dir*. Si no se especifica dirección, se obtendrá una asignando mediante *malloc* del tamaño que se le indica.
- **sigaction –showstack** Nos muestra la dirección y el tamaño de la pila alternativa

sigadm [**–block|–unblock|–ign|–dfl**]

- **sigadm –block [S1] [S2] ...** Enmascara (mediante *sigprocmask* las señales S1, S2 .... Si no se especifican señales nos informa de las que están enmascaradas.
- **sigadm –unblock [S1] [S2] ...** Desenmascara (mediante *sigprocmask* las señales S1, S2 .... Si no se especifican señales nos informa de las que están enmascaradas.
- **sigadm –ign [S1] [S2] ...** Ignora las señales S1, S2 .... Si no se especifican señales nos informa de las que están ignoradas.
- **sigadm –dfl [S1] [S2] ...** Pone las señales S1, S2 ... a su acción por defecto. Si no se especifican señales nos informa de las que están a su accion por defecto.

bucle Hace que el shell entre en un bucle infinito. Instala un manejador para SIGINT que permite salir del bucle pulsando control-c para seguir ejecutando el shell.

segmentation produce un fallo de segmentación en el shell. (No vale enviar SIGSEGV, tiene que ser un fallo de segmentación de verdad).

fpe produce una excepción de la unidad en punto flotante en el shell. (No vale enviar SIGFPE, tiene que ser un fallo de segmentacin de verdad).

getpri [pid] Mustra la prioridad del proceso *pid*. Si no se suministra *pid* muestra la del shell

setpri valor [pid] Cambia la prioridad del proceso *pid* a valor. Si no se suministra *pid* cambia la del shell

pri valor 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. Si se especifica **LISTADEVARIABLES, la ejecución es mediante execve y el entorno se indica en LISTAVARI-

ABLES. 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. EL proceso se ejecuta con su prioridad cambiada (mediante *setpriority*) a *valor*

background-pri **valor arg1 arg2 ...[** LISTAVARIABLES** Exactamente igual al caso anterior, pero el proceso es ahora en segundo plano.

exec-pri **valor arg1 arg2 ...[** LISTAVARIABLES** Análogo a los anteriores, pero sin crear proceso (el shell reemplaza su código)

infoproc **[-a|-s|-t|-p pid1 pid2 ...]** Mustra información de los procesos que se han lanzado desde el shell en segundo plano. Para cada proceso muestra EN UNA SOLA LINEA: pid, prioridad, estado, linea de comando que ejecuta e instante de comienzo. Ejemplo

```
->infoproc
   997         ACTIVO      Mon Dec 21 13:01:08 2009 0 xterm
  1004   TERM NORMAL (0)  Mon Dec 21 13:01:44 2009 0 xclock -digital -update
  1005 T. SENAL: SIGSEGV  Mon Dec 21 13:01:57 2009 0 xclock -digital -update
->
```

- **infoproc** Muestra información de todos los procesos creados en segundo plano desde el shell
- **infoproc -a** Muestra información de todos los procesos creados en segundo plano desde el shell que están activos
- **infoproc -s** Muestra información de todos los procesos creados en segundo plano desde el shell que han terminado debido a una señal
- **infoproc -t** Muestra información de todos los procesos creados en segundo plano desde el shell que han terminado normalmente
- **infoproc -p pid1 pid2 ...** Muestra información de todos los procesos creados en segundo plano desde el shell con pids *pid1 pid2 ...*

clearproc **[-a|-s|-t|-p pid1 pid2 ...]** Elimina procesos de la lista de procesos en segundo plano (no termina los procesos, simplemente los elimina de la lista del shell)

- **clearproc** Muestra información de todos los procesos creados en segundo plano desde el shell
- **clearproc -a** Elimina de la lista los que están activos
- **clearproc -s** Elimina de la lista los que han terminado debido a una señal
- **clearproc -t** Elimina de la lista los que han terminado normalmente
- **clearproc -p pid1 pid2 ...** Elimina de la lista los de pids *pid1 pid2 ...*

usercred [-u uid|-l login] Establece o muestra la credencial de usuario del proceso

- **usercred** Muestra las credenciales de usuario (real y efectiva) del proceso. Muestra tanto el número (uid) como el login (nombre) asociado
- **usercred -u uid** Cambia (si puede) la credencial de usuario del proceso. *uid* representa un número de usuario
- **usercred -l login** Cambia (si puede) la credencial de usuario del proceso. *uid* representa un login (nombre) de usuario

NOTAS

Información detallada de las llamadas al sistema y las funciones de la librería debe obternerse con man, sección 2 para las llamadas al sistema y sección 3 para las funciones de libreria (sigaction, sigprocmask, sigaltstack, stat, setpriority, getpwent ...). Todo el manejo de señales debe realizarse con las llamadas al sistema de System V R4 (*sigaction, sigprocmask, ...*)

Salvo que se diga explicitamente (p.e., comandos *segmentation* y *fpe*, en ningún caso la práctica puede producir error en tiempo de ejecución.

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 22 ENERO 2010