SISTEMAS OPERATIVOS II

Tercer curso Ingeniería Técnica Informática Sistemas. Curso 2011-2012

Práctica 3: Procesos en Unix: Señales

Continuar la codificación de un 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 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.

A efectos de programación de está práctica podemos suponer que hay 32 señales en el sistema. Las funciones disponibles en

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

permiten convertir de nombre de señal a númeor de señal y viceversa

Comandos a implementar en esta práctica

senal [-dfl|-ign|man] [-l] [-r] [-resethand] [-onstack] [-nodefer] [-restart] [-sNN] [-mSEN1] [-mSEN2] ... S1 S2...

- **senal -man [-l] [-r] [-resethand] [-onstack] [-nodefer] [-restart] [-sNN] [-mSEN1] [-mSEN2] ... S1 S2...** Instala un manejador (utilizando la llamada *sigaction*) para las señales S1, S2 ... con las siguientes caractrísticas

  El manejador incrementa un contador (para cada señal) que indica cuantas veces se ha ejecutado.

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

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

  -l 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 -l 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 -sNN). Se enviará hasta que el contador de señales recibidas alcance el valor *maxReenvios*. Si *maxReenvios* es 0, se reenvía continuamente

  -resethand El manejador es temporal: se instala con el flag SA_RESETHAND

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

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

  -restart El manejador se instala con el flag SA_RESTART Ejemplo

```
-> senal -man -r -nodefer -onstack -s10 -mUSR1 -mHUP INT SEGV
-> senal -ign 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 cuando 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. La segunda línea pone SIGSEGV como ignorada

- **senal -dfl [S1] [S2] ...** Establece las señales S1, S2 ... a su acción por defecto. Si no se especifican señales nos informa de las que están a su acción por defecto
- **senal -ign [S1] [S2] ...** Establece las señales S1, S2 ... a ignoradas. Si no se especifican señales nos informa de las que están ignoradas
- **senal [S1] [S2] ...** Es equivalente a *seninfo [S1] [S2] ...*

maxreenvios [N] Establece el valor de *maxReenvios* (definido en el apartado anterior): el número máximo de veces que un manejador que se ha instalado con la opción *-r* reenvía una señal. Si no se especifica N nos informa de cuál es su valor

seninfo [-cont] [-cero] [S1] [S2] ...

- **seninfo [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. Ejemplo

```
#seninfo INT HUP SEGV
INT Enmacarada manejador 0x30045a00 SA_RESTART SA_NODEFER
      mascara asociada SIHGUP SIGUSR1
HUP No enmascarada Accion por defecto
SEGV No emmascarada Ignorada
```

- **seninfo -cont [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.
- **seninfo -cero [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.

senpila [*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. (/***/ es posible que, especificando alguna dirección de memoria concreta como pila alternativa, se pueda producir un fallo de segmentación al recibir una señal cuyo manejador se ejecuta en dicha

pila). Si no se especifica *tam* nos muestra la dirección y el tamaño de la pila alternativa

senmasc **[-n|-d] S1 S2 ...** Enmascara o desenmascara (mediante *sigprocmask*) las señales S1, S2 .... Si no se especifican señales nos informa de las que están enmascaradas.

- **senmasc [S1] [S2] ...** Enmascara (mediante *sigprocmask*) las señales S1, S2 ....
- **senmasc -n [S1] [S2] ...** Hace que el conjunto de señales enmascaradas del proceso sea (mediante *sigprocmask*) S1, S2 ....
- **senmasc -d [S1] [S2] ...** Desenmascara (mediante *sigprocmask*) las señales S1, S2 ....

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). (/***/ evidentemente produce un error en tiempo de ejecución)

Información detallada de las llamadas al sistema y las funciones de la librería debe obternerse con man (sigprocmask, sigaction, sigaltstack, ...)

Para comprobar el funcionamiento de flag SA_RESTART en *sigaction* es conveniente que contemplemos el caso de que la función que lee de la entrada puede devolver un error (**NO DEBE TERMINAR EL PROCESO EN CASO DE QUE ESO OCURRA**). Por ejemplo

```
if (fgets(linea, stdin, MAXLINEA)==NULL)
   perror ("Error al leer de teclado");
```

FORMA DE ENTREGA Como en prácticas anteriores

FECHA DE ENTREGA VIERNES 23 DICIEMBRE DE 2012