Señales

En UNIX una señal es una notificación software (evento asíncrono) del sistema operativo a un proceso. Limitadas en número e identificadas individualmente, se les asocia a una acción que se ejecuta en el momento que la señal es entregada.

El manual de sistema (`man 7 signal`) ofrece un listado y una breve descripción de las señales existentes en LINUX. Las funciones `psignal()` y `strsignal()` facilitan una descripción de una señal determinada.

Las señales pueden ser generadas por diversas causas:

- Pulsación de caracteres especiales desde el terminal de control. Ej: *SIGINT*, *SIGQUIT*, ..
- Excepción. Ej.: *SIGSEGV*, *SIGTRAP*, ..
- Condiciones software. Ej.: *SIGALRM*, *SIGPIPE*, …
- Envío expreso mediante la llamada `kill()`:

```
SINOPSIS
#include <sys/types.h>
#include <signal.h>

int kill(pid_t pid, int sig);
```

El parámetro `pid` indica el proceso al que se desea enviar la señal `sig`.

La recepción de una señal por parte un proceso implica una de las siguientes acciones:

- Ignorar la señal recibida (*SIG_IGN*) –¡Ni SIGKILL, ni SIGSTOP, pueden ser ignoradas!–.
- Ejecutar una *acción por defecto* (*SIG_DFL*). Dependiendo de la señal ésta puede ser:
  - Ignorar la señal.
  - Detener el proceso.
  - Terminar el proceso.
  - Terminar el proceso y producir un volcado (*core*).
- Ejecutar una función definida por el usuario. En este caso se dice que la función captura la señal –¡Ni SIGKILL, ni SIGSTOP, pueden ser capturadas!– y su prototipo debe ser: `void funcion(int)`.

La *captura* de señales se puede llevar a cabo mediante dos llamadas distintas, `signal()` y `sigaction()`, aunque es preferible el uso de esta última.

```
SINOPSIS
#include <signal.h>

int sigaction(int signum, const struct sigaction *act,
      struct sigaction *oldact);
```

El parámetro `signum` indica la señal que se desea capturar.

Los punteros `oldact` y `act` apuntan a las estructuras que definen la acción actualmente asociada a la señal y la nueva acción que se desea asociar respectivamente. Si `oldact` es nulo no se guarda la información de la acción anterior.

La estructura `sigaction` se define como:

```
struct sigaction {
      void (*sa_handler)(int);
      void (*sa_sigaction)(int, siginfo_t *, void *);
      sigset_t sa_mask;
      int sa_flags;
      void (*sa_restorer)(void);
}
```

cuyos principales campos son:

- `sa_handler` – puntero a la función que *captura* la señal.
- `sa_flags` – opciones de manejo de la señal
- `sa_mask` – máscara de señales: conjunto de señales que se bloquean durante la ejecución de la función (por defecto siempre se bloquea la señal que desencadena la acción).

Las funciones `sigemptyset()`, `sigfillset()`, `sigaddset()`, `sigdelset()`,`sigismember()` facilitan la gestión de conjuntos de señales de tipo `sigset_t`.

Las llamadas `sigprocmask()` y `sigsuspend()` permiten manipular la máscara de señales del proceso y la llamada `sigpending()`, permite gestionar las señales bloqueadas.

Al crear un nuevo proceso –mediante `fork()`– éste hereda el tratamiento de las señales establecido en el proceso padre. No obstante, si posteriormente este proceso hijo invoca la llamada `exec()`, se asociará la *acción por defecto* a todas aquellas señales que no estuvieran siendo *ignoradas*.

Salvo las *"lentas"* (aquellas que pueden ser afectadas por un bloqueo indefinido: `pause()`, `wait()`, `read()` de terminal o pipe, etc), las llamadas al sistema no pueden interrumpirse mediante una señal.

Otras funciones y llamadas al sistema relacionadas con el empleo de señales son:
- `pause()` – duerme el proceso en curso hasta que reciba una señal.
- `alarm()` – envía una señal SIGALRM al proceso en curso transcurrido un cierto intervalo de tiempo (en segundos).
- `abort()` – manda la señal SIGABORT, provocando la finalización *anormal* del proceso.

Ejercicio: Crear un programa (Mensaje) que muestra un texto en pantalla de manera periódica. El periodo, en segundos, se especifica como argumento de entrada del programa. Para finalizarlo es necesario mandarle la señal *SIGTERM* mediante el comando `kill` del sistema, y antes de finalizar el programa mostrará el número de segundos que han transcurrido desde su inicio. El programa ignorará la tecla de interrupción (CTRL+C).

Sintaxis: ***mensaje*** *segundos [mensaje_opcional]*