Sistemas Operativos
Signals

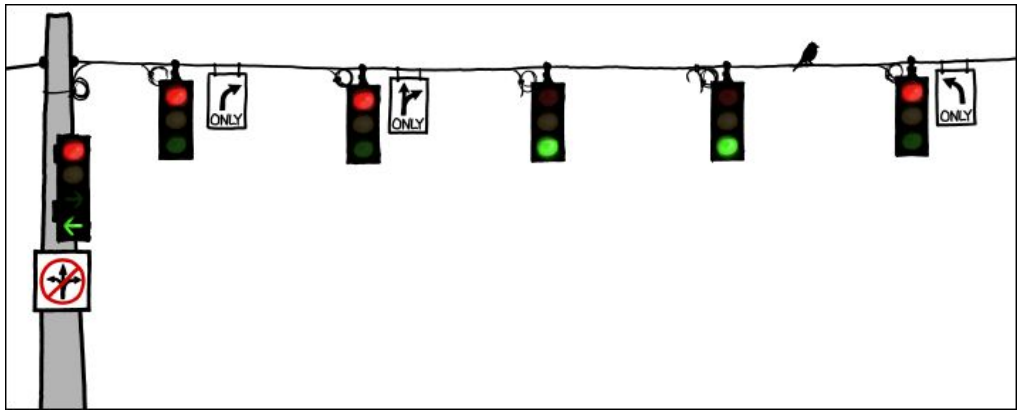
Signals

Signals o Señales

Una señal (signal) es una notificación que envía a un proceso cuando un determinado evento ocurre.

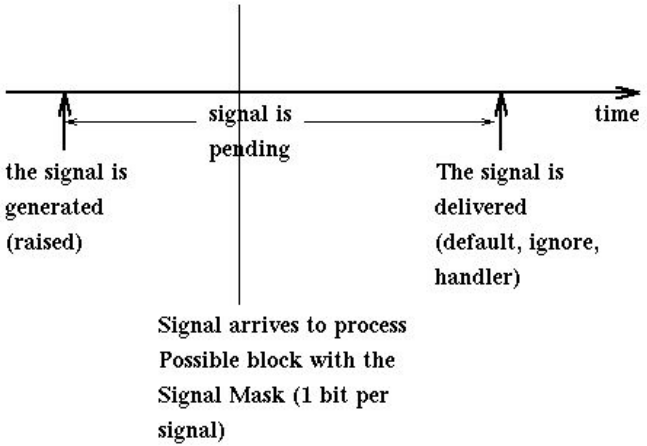
A las señales se les dice a veces *interrupciones por software*.

Son análogas a las interrupciones por hardware, interrumpen el flujo normal de la ejecución de un programa, pero en la mayoría de los casos no es posible predecir el arribo de una señal.



Signals o Señales

Son análogas a las interrupciones por hardware, interrumpen el flujo normal de la ejecución de un programa, pero en la mayoría de los casos no es posible predecir el arribo de una señal.



Signals o Señales

Un proceso puede enviar una señal a otro proceso [1].

Un proceso puede enviarse señales a él mismo.

Normalmente la fuente de envio de señales hacia un proceso es el Kernel.

DESCRIPTION

This **signal**() facility is a simplified interface to th

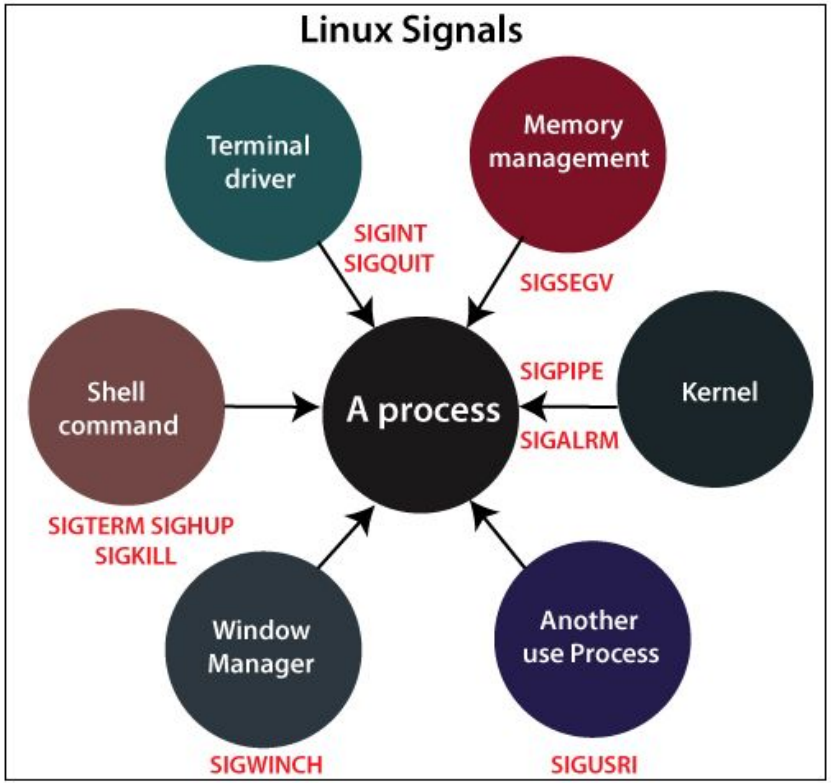
Signals allow the manipulation of a process from outsi
(children). There are two general types of signals: t
termination of a program might result from an irrecove
acter. Signals are used when a process is stopped bec
Signals are optionally generated when a process resume
ready at the control terminal. Most signals result ir
instead cause the process receiving them to be stopped
SIGKILL and SIGSTOP signals, the **signal**() function all
nals are defined in the file <signal.h>:

No	Name	Default Action	Description
1	SIGHUP	terminate process	terminal line hangup
2	SIGINT	terminate process	interrupt program
3	SIGQUIT	create core image	quit program
4	SIGILL	create core image	illegal instruction
5	SIGTRAP	create core image	trace trap
6	SIGABRT	create core image	abort program (formerly SIGIOT)
7	SIGEMT	create core image	emulate instruction executed
8	SIGFPE	create core image	floating-point exception
9	SIGKILL	terminate process	kill program
10	SIGBUS	create core image	bus error
11	SIGSEGV	create core image	segmentation violation
12	SIGSYS	create core image	non-existent system call invoked
13	SIGPIPE	terminate process	write on a pipe with no reader
14	SIGALRM	terminate process	real-time timer expired
15	SIGTERM	terminate process	software termination signal
16	SIGURG	discard signal	urgent condition present on socket
17	SIGSTOP	stop process	stop (cannot be caught or ignored)
18	SIGTSTP	stop process	stop signal generated from keyboard
19	SIGCONT	discard signal	continue after stop
20	SIGCHLD	discard signal	child status has changed
21	SIGTTIN	stop process	background read attempted from control terminal

Signals o Señales

Motivos por los los que el Kernel envía señales a un proceso:

- Una excepción de software (div por 0).
- El usuario presiona algún caracter especial en la terminal (CRTL-C).
- Un evento que tiene que ver con el software ocurrió, un proceso hijo terminó.



Signals o Señales

Cada señal está definida como un valor entero (corto) único en <signal..h> con un nombre simbólico que comienza con SIGxxxx.

Las señales se agrupan en dos categorías:

Tradicionales o estándares, cuyos valores van de 1 a 31.

De Tiempo Real.

SIGINT	2	Interrupt a process (used by **Ctrl-C**)
SIGHUP	1	Hang up or shut down and restart process
SIGKILL	9	Kill the process (cannot be ignored or caught elsewhere)
SIGTERM	15	Terminate signal, (can be ignored or caught)
SIGTSTP	20	Stop the terminal (used by **Ctrl-z**)
SIGSTOP	19	Stop execution (cannot be caught or ignored)

DESCRIPTION

This **signal**() facility is a simplified interface to the more general sigaction(2) facility.

Signals allow the manipulation of a process from outside its domain, as well as allowing the process to manipulate itse
(children). There are two general types of signals: those that cause termination of a process and those that do not.
termination of a program might result from an irrecoverable error or might be the result of a user at a terminal typing
acter. Signals are used when a process is stopped because it wishes to access its control terminal while in the backgr
Signals are optionally generated when a process resumes after being stopped, when the status of child processes changes
ready at the control terminal. Most signals result in the termination of the process receiving them, if no action is t
instead cause the process receiving them to be stopped, or are simply discarded if the process has not requested otherw
SIGKILL and SIGSTOP signals, the **signal**() function allows for a signal to be caught, to be ignored, or to generate an i
nals are defined in the file <signal.h>:

No	Name	Default Action	Description
1	SIGHUP	terminate process	terminal line hangup
2	SIGINT	terminate process	interrupt program
3	SIGQUIT	create core image	quit program
4	SIGILL	create core image	illegal instruction
5	SIGTRAP	create core image	trace trap
6	SIGABRT	create core image	abort program (formerly SIGIOT)
7	SIGEMT	create core image	emulate instruction executed
8	SIGFPE	create core image	floating-point exception
9	SIGKILL	terminate process	kill program
10	SIGBUS	create core image	bus error
11	SIGSEGV	create core image	segmentation violation
12	SIGSYS	create core image	non-existent system call invoked
13	SIGPIPE	terminate process	write on a pipe with no reader
14	SIGALRM	terminate process	real-time timer expired
15	SIGTERM	terminate process	software termination signal
16	SIGURG	discard signal	urgent condition present on socket
17	SIGSTOP	stop process	stop (cannot be caught or ignored)
18	SIGTSTP	stop process	stop signal generated from keyboard
19	SIGCONT	discard signal	continue after stop
20	SIGCHLD	discard signal	child status has changed
21	SIGTTIN	stop process	background read attempted from control terminal

Signals o Señales

Signals o Señales

SIGHUP	1	Term	Cuelgue detectado en la terminal de control o muerte del proceso de control
SIGINT	2	Term	Interrupción procedente del teclado
SIGQUIT	3	Core	Terminación procedente del teclado
SIGILL	4	Core	Instrucción ilegal
SIGABRT	6	Core	Señal de aborto procedente de abort(3)
SIGFPE	8	Core	Excepción de coma flotante

Signals o Señales

SIGKILL	9	Term	Señal de matar
SIGSEGV	11	Core	Referencia inválida a memoria
SIGPIPE	13	Term	Tubería rota: escritura sin lectores
SIGALRM	14	Term	Señal de alarma de alarm(2)
SIGTERM	15	Term	Señal de terminación
SIGUSR1	30,10,16	Term	Señal definida por usuario 1
SIGUSR2	31,12,17	Term	Señal definida por usuario 2

Signals o Señales

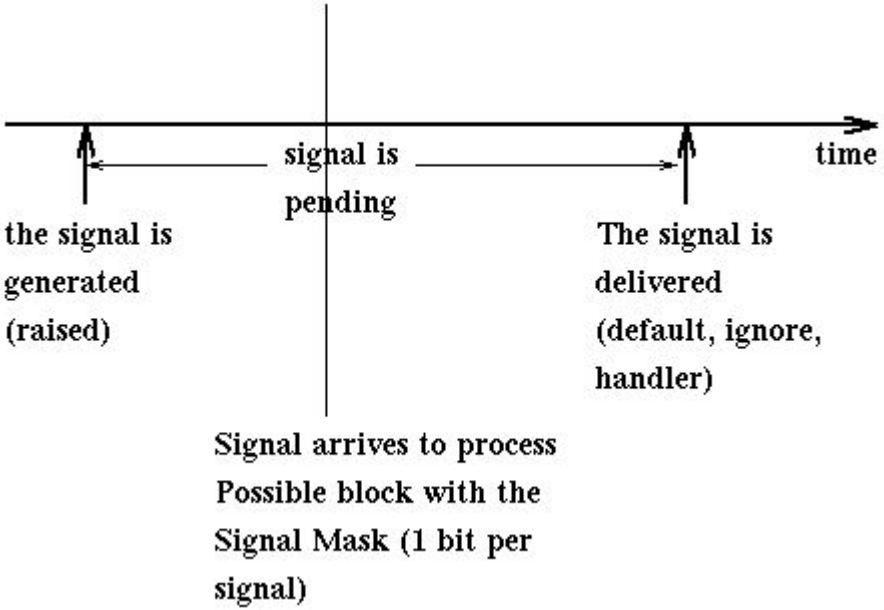
SIGCHLD	20,17,18	Ign	Proceso hijo terminado o parado
SIGCONT	19,18,25		Continuar si estaba parado
SIGSTOP	17,19,23	Stop	Parar proceso
SIGTSTP	18,20,24	Stop	Parada escrita en la tty
SIGTTIN	21,21,26	Stop	E. de la tty para un proc. de fondo
SIGTTOU	22,22,27	Stop	S. a la tty para un proc. de fondo

Signals: Ciclo de Vida

Estados de una señal:

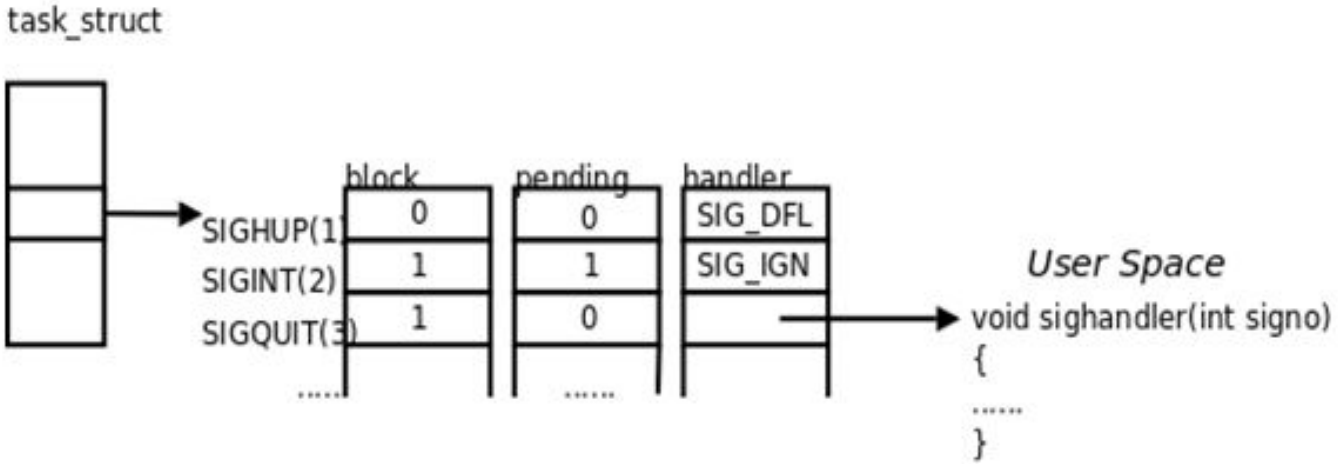
- Generada
- Entregada
- Pendiente

Una señal es ***generada*** por un evento, una vez generada, esta es ***entregada*** a un determinado proceso, que tomará en algún momento una acción. El tiempo entre la generación y la entrega la señal está ***pendiente***.



Signals: Ciclo de Vida

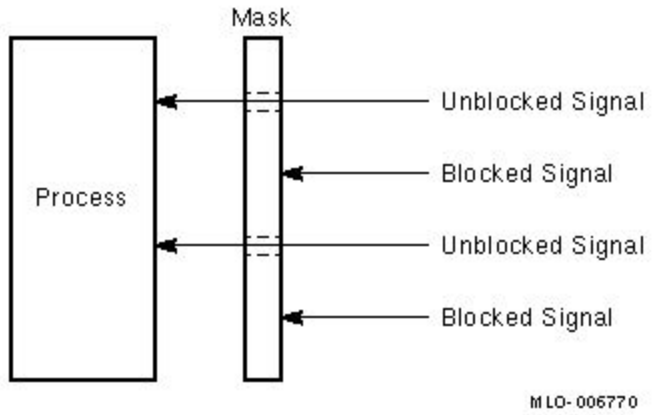
Una señal pendiente es entregada al proceso tan pronto como este esté planificado para ser el próximo en correr o inmediatamente si este se encuentra en ejecución.



Signals: Ciclo de Vida

A veces es necesario que un proceso no sea interrumpido por la entrega de una señal. Para ello se agrega una máscara de señal.

- **SIG_BLOCK** Adds the set of signals specified in the second argument to the process's signal mask
- **SIG_UNBLOCK** Subtracts the set of signals specified in the second argument from the process's signal mask
- **SIG_SETMASK** Replaces the process's signal mask with the set of signals specified in the third argument



Signals: Ciclo de Vida

Una vez que la señal se entrega, el proceso puede llevar a cabo una de las siguientes acciones por defecto:

1. La señal es **ignorada (ignored)**, es descartada por el kernel y no tiene efecto sobre el proceso.
2. El proceso es **terminado** (killed), muchas veces se denomina abnormal process termination.
3. Un archivo de **core dump** es generado y el proceso terminado. (imagen de la memoria virtual)
4. El proceso es **detenido** (stopped) - ejecución suspendida.
5. El proceso **continúa** (resumed) su ejecución.

Signals: Disposition

Un programa puede modificar la acción por defecto de una determinada señal cuando esta es entregada. Esto se denomina definir la ***disposition***:

1. Que ocurra la acción por defecto. Cuando es bueno esto?
2. La señal es ignorada. Cuando es bueno esto?.
3. Un manejador de la señal (signal handler) es ejecutado.

Signals: Disposition

Name	Signal number	Description	SUSv3	Default
SIGABRT	6	Abort process	•	core
SIGALRM	14	Real-time timer expired	•	term
SIGBUS	7 (SAMP=10)	Memory access error	•	core
SIGCHLD	17 (SA=20, MP=18)	Child terminated or stopped	•	ignore
SIGCONT	18 (SA=19, M=25, P=26)	Continue if stopped	•	cont
SIGEMT	undef (SAMP=7)	Hardware fault		term
SIGFPE	8	Arithmetic exception	•	core
SIGHUP	1	Hangup	•	term
SIGILL	4	Illegal instruction	•	core
SIGINT	2	Terminal interrupt	•	term
SIGIO / SIGPOLL	29 (SA=23, MP=22)	I/O possible	•	term
SIGKILL	9	Sure kill	•	term
SIGPIPE	13	Broken pipe	•	term
SIGPROF	27 (M=29, P=21)	Profiling timer expired	•	term
SIGPWR	30 (SA=29, MP=19)	Power about to fail		term
SIGQUIT	3	Terminal quit	•	core
SIGSEGV	11	Invalid memory reference	•	core
SIGSTKFLT	16 (SAM=undef, P=36)	Stack fault on coprocessor		term
SIGSTOP	19 (SA=17, M=23, P=24)	Sure stop	•	stop
SIGSYS	31 (SAMP=12)	Invalid system call	•	core
SIGTERM	15	Terminate process	•	term
SIGTRAP	5	Trace/breakpoint trap	•	core
SIGTSTP	20 (SA=18, M=24, P=25)	Terminal stop	•	stop
SIGTTIN	21 (M=26, P=27)	Terminal read from BG	•	stop
SIGTTOU	22 (M=27, P=28)	Terminal write from BG	•	stop
SIGURG	23 (SA=16, M=21, P=29)	Urgent data on socket	•	ignore
SIGUSR1	10 (SA=30, MP=16)	User-defined signal 1	•	term
SIGUSR2	12 (SA=31, MP=17)	User-defined signal 2	•	term
SIGVTALRM	26 (M=28, P=20)	Virtual timer expired	•	term
SIGWINCH	28 (M=20, P=23)	Terminal window size change		ignore
SIGXCPU	24 (M=30, P=33)	CPU time limit exceeded	•	core
SIGXFSZ	25 (M=31, P=34)	File size limit exceeded	•	core

Signals: Tipos de Señales -

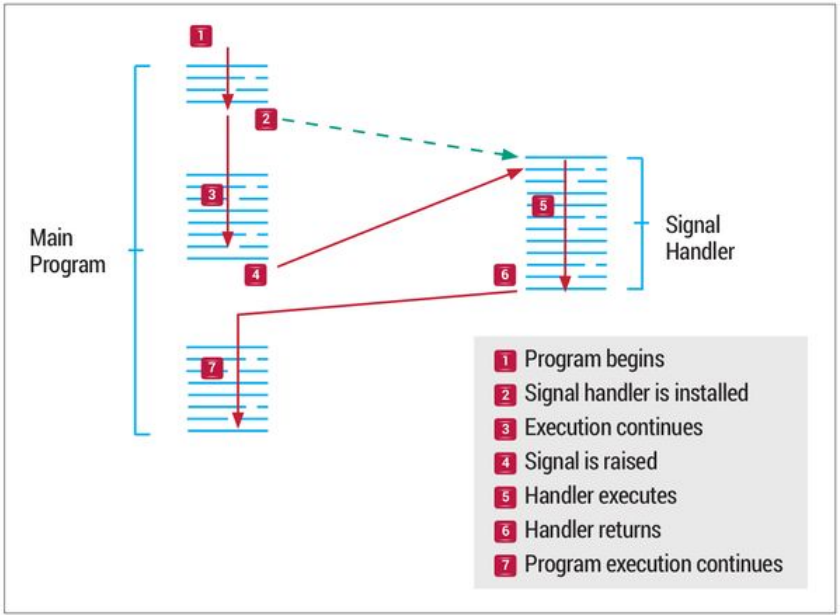
Un programa puede modificar la acción por defecto de una determinada señal cuando esta es entregada. Esto se denomina definir la ***disposition***:

1. Que ocurra la acción por defecto. Cuando es bueno esto?
2. La señal es ignorada. Cuando es bueno esto?.
3. Un manejador de la señal (signal handler) es ejecutado.

Signals: Signal Handler

- Un signal handler es una función, escrita por el programador, que realiza la tarea apropiada en respuesta a la entrega de una señal.

- Por Ejemplo, la Shell tiene un signal handler para la señal SIGINT (Crtl-C). Cual es?



Signals: Signal Handler

It is not possible to use signal() to retrieve the current disposition of a signal without at the same time changing that disposition.

```
#include <signal.h>

void ( *signal(int sig, void (*handler)(int)) ) (int);
                Returns previous signal disposition on success, or SIG_ERR on error
```

El primer argumento de signal() , identifica la señal a la cual se le quiere cambiar la disposition.

El segundo argumento handler es un puntero a una función que debe ser llamada cuando la señal se entrega.

signal() no tiene una implementación estándar.

El valor de retorno de Signal es un puntero a la antigua disposition

Signals: Signal Handler

It is not possible to use signal() to retrieve the current disposition of a signal without at the same time changing that disposition.

```
#include <signal.h>

void ( *signal(int sig, void (*handler)(int)) ) (int);
```

Returns previous signal disposition on success, or SIG_ERR on error

1. SIG_DFL: It is a pointer to system default function SIG_DFL(), declared in h header file. It is used for taking default action of the signal.
2. SIG_IGN: It is a pointer to system ignore function SIG_IGN(),declared in h header file.
3. User defined handler function pointer: The user defined handler function type is void(*)(int), means return type is void and one argument of type int.

Signals: Envío de señales - kill()

A traves de la system call kill() un proceso puede enviar señales a otro proceso .


```
#include <signal.h>

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

Returns 0 on success, or −1 on error

Signals: Envío de señales - raise()

La system call raise() permite a un proceso enviarse señales a sí mismo..

```
#include <signal.h>

int raise(int sig);
```

Returns 0 on success, or nonzero on error