SC canale 2 - 19/5/2025

lunedì 19 maggio 2025 12:09

SEGNALI

Un segnale è una notifica asincrona inviata dal SO ad un processo.

	DESTINATARIO	MITTENTE	GESTORE
INTERRUPT	KERNEL	periferiche, CPU, programma	KERNEL (INTERRUPT HANDLER)
SEGNALI	PROCESSO	kernel, programma	PROCESSO (SIGNAL HANDLER)

Ogni segnale è definito da:

- NUMERO IDENTIFICATIVO : identificare il tipo
- PID : identifica il processo destinatario

Quando un processo viene notificato (dal kernel) dell'arrivo di un segnale si hanno tre possibili comportamenti:

TERM : il processo viene terminato

CORE : " " " " ed il kernel genera un "core dump"

Copia dell'immagine di memoria del processo e dei metadati contenuti nel PCB ai fini di debug.

IGN : il processo ignora il segnale

handler : il processo esegue il signal handler precedentemente configurato

I principali segnali sono:

CAT.	SEGNALE	NUM	DESCRIZIONE	IGN?	GEST?	DEFAULT
TERMINAZIONE	SIGTERM	15	soft kill	✓	✓	TERM
	SIGINT	2	generato da CTRL+C	✓	✓	TERM
	SIGKILL	9	hard kill	✗	✗	TERM
	SIGCHLD	17	Terminazione proc. figlio	✓	✓	IGN
MEM	SIGSEGV	11	accesso errato in mem	✓	✓	TERM
TIME	SIGALRM	14	timer di sistema	✓	✓	TERM

ATTENZIONE : un interrupt, attraverso il suo handler, può generare un segnale indirizzato ad un processo

Es.: CTRL+C → interrupt tastiera → int handler → SIGINT

utente periferica HW kernel va al processo

CTRL+Z → " " → " " → SIGSTOP

Timer HW → interrupt timer → " " → SIGALRM

GESTIONE DEI SEGNALI

Per inviare un segnale ad un processo "pid".

```
#include <signal.h>
int kill (pid_t pid, int sig);
```

sig: identificativo del segnale da inviare

pid: PID del processo destinatario

È possibile inviare un segnale solo ai processi avviati dallo stesso utente.

Anche da terminale!

```
kill -9 12345
```

12345: PID destinatario

-9: ID del segnale

Per gestire un segnale

```
int sigaction ( int sig,
                const struct sigaction * act,
                struct sigaction * oact)
```

int: [0; -1]

int sig → il segnale da gestire

const struct sigaction * act → indica come gestire il segnale

struct sigaction * oact → ottiene info su come viene gestito il segnale

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

sa_handler → come gestire il segnale:

- puntatore a funzione (handler)
- SIG_IGN
- SIG_DFL

sa_sigaction → usato per configurazioni più complesse

sa_mask → set di segnali che devono essere ignorati durante l'esecuzione del signal handler

sa_flags → flag che definiscono il comportamento del segnale

sa_restorer → non usato