

# Segnali ed allarmi

# Segnali

- Tecnica di comunicazione asincrona tra processi
- Si può solo inviare uno tra un insieme fissato di segnali
- Il destinatario viene interrotto, e si salta ad una procedura specifica (*handler*)
- Se il destinatario non si predispone a ricevere un segnale, tipicamente esso viene terminato dal segnale

# Uso dei segnali

- Segnalare situazioni d'errore (segmentation fault, divisione per zero)
- Segnalare la terminazione di un processo figlio
- Terminare un processo
- Altri usi dipendenti dall'applicazione

# Alcuni segnali

<i>nome</i>	<i>significato</i>	<i>azione</i>
SIGINT	interruzione da tastiera (Ctrl-c)	terminare
SIGQUIT	quit da tastiera (Ctrl-y)	terminare
SIGTERM	terminazione da tastiera (Ctrl-\)	terminare
SIGKILL*	terminazione forzata	terminare
SIGSTOP*	fermare il processo	<i>fermare</i>
SIGCONT	il processo può continuare	<i>riprendere</i>

(\*) non può essere catturato né ignorato

# Alcuni segnali

<i>nome</i>	<i>significato</i>	<i>azione</i>
SIGCHLD	figlio terminato o fermato	<i>ignorare</i>
SIGALRM	suona la sveglia!	terminare
SIGSEGV	segmentation fault (inviato dal kernel)	terminare
SIGUSR1	a disposizione dell'utente	terminare
SIGUSR2	a disposizione dell'utente	terminare

# Inviare segnali

- I segnali si inviano ai processi
- Un processo si individua usando il suo *process id* (pid), che è un intero non negativo
  - i pid 0 ed 1 sono riservati al sistema
- Per vedere i pid dei vostri processi correnti, usare “ps”

# Inviare segnali

- Per inviare un segnale, bisogna averne il **permesso**
- L'uid (*real o effective*) del **mittente** deve essere uguale all'uid (*real o effective*) del **destinatario**
- root può inviare segnali a tutti i processi

# Inviare segnali dalla shell

comando: `kill [-<segnale>] <pid>`  
oppure: `kill -l`

- `kill -INT 127` invia il segnale SIGINT al processo il cui pid è 127
- di default, viene inviato il segnale SIGTERM
  - `kill 127` equivale a `kill -TERM 127`
- `kill -l` elenca tutti i segnali ed i loro valori numerici

# Inviare segnali in C

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

- `kill(127, SIGINT)` invia il segnale SIGINT al processo il cui pid e' 127
- con `pid==-1`, invia il segnale a tutti i processi per i quali si ha il permesso
- restituisce 0 in caso di successo e -1 in caso di errore

# Reazione ai segnali

- Un processo può scegliere come reagire a ciascun segnale
- Le possibilità sono:
  - ignorare il segnale
  - catturare il segnale ed eseguire un *handler*
  - affidarsi al comportamento di default (che varia in base al segnale)

# Catturare i segnali

- Un handler (gestore) è una funzione del tipo:

```
void funzione(int num_segnaile) {  
    printf("%d", num_segnaile);  
    ...  
}
```

Una volta che l'handler termina, si torna al punto in cui il programma era stato interrotto.

# Catturare un segnale

```
typedef void (*sighandler_t)(int);
```

```
sighandler_t signal(int signum, sighandler_t handler);
```

- `signal(SIGINT, foo)` imposta la funzione `foo` come handler del segnale `SIGINT`
- si può anche richiedere di ignorare il segnale
  - `signal(SIGINT, SIG_IGN)`
- oppure ritornare alla reazione di default
  - `signal(SIGINT, SIG_DFL)`

# Catturare un segnale

```
typedef void (*sighandler_t)(int);  
sighandler_t signal(int signum, sighandler_t handler);
```

- restituisce l'impostazione precedente, cioè uno dei seguenti valori:
  - l'indirizzo dell'handler precedente
  - SIG\_DFL: reazione di default
  - SIG\_IGN: ignorare il segnale
  - SIG\_ERR: errore durante `signal`

# Impostare una sveglia

```
unsigned int alarm(unsigned int seconds);
```

- `alarm(30)` prenota un segnale `SIGALRM`, che sarà inviato tra 30 secondi
- restituisce 0 se non c'era nessuna sveglia già prenotata
- altrimenti, restituisce il tempo rimanente perché la vecchia sveglia suonasse

# Aspettare un segnale

```
int pause(void);
```

- Aspetta l'arrivo di un segnale
- Restituisce -1, solo se:
  - arriva un segnale,
  - il segnale viene catturato, e
  - l'handler corrispondente esegue “return”

# Esercizio 1

- Usando il comando kill della shell, tentare di inviare il segnale SIGINT ai processi con pid 1 e 2

# Esercizio 2

- Eseguire una lunga operazione in background e poi terminarla inviando il segnale SIGTERM
  - ad es.,

```
> ls -lR / > pippo.txt 2>&1 &  
[4] 23008  
> ps  
PID TTY      TIME CMD  
15918 pts/6    00:00:03 bash  
20143 pts/6    00:00:05 emacs  
23008 pts/6    00:00:06 ls  
23009 pts/6    00:00:00 ps  
> /bin/kill 23008
```

# Esercizio 3

- Scrivere un programma C “aspetta.c”, che scrive un messaggio su standard output ogni volta che riceve i segnali SIGINT o SIGUSR1.
- Il programma non deve mai terminare spontaneamente.
- Lanciare il programma. Usando il comando kill della shell, provare ad inviargli quei due segnali, ed infine terminarlo con un altro segnale.