Corso «Sistemi Operativi» AA 2020-2021

Marco Tessarotto

signals

signals

- Un segnale (signal) è la notifica a un processo che un evento si è verificato.
- I segnali sono talvolta descritti come interruzioni software (software interrupt)
- I segnali sono analoghi alle interruzioni hardware (hardware interrupt) in quanto interrompono il normale flusso di esecuzione di un programma
- non è possibile prevedere esattamente quando arriverà un segnale

signals

- Un processo può mandare (se ha i permessi sufficienti) un segnale ad un altro processo: diventa una tecnica di sincronizzazione tra processi e/o una tecnica di IPC
- Un processo può mandare un segnale a se stesso
- La sorgente usuale di molti segnali è il kernel

Tipi di evento che generano segnali

- Hardware exception: l'hardware riconosce una condizione di errore che viene notificata al kernel, che a sua volta invia un corrispondente segnale al processo interessato (es: accesso ad una parte di memoria non accessibile, divisione per 0)
- L'utente digita a terminale uno dei caratteri speciali che generano segnali: 'interrupt character' (ctrl-C), 'suspend' (ctrl-Z)
- Evento software: es. terminazione di un processo figlio provoca un segnale verso il processo padre; un processo manda un segnale ad un altro processo

segnali

- Ogni segnale è definito da un numero intero (piccolo), a partire da 1
- I segnali sono definiti in <signal.h>, nomi del tipo SIGxxxx
- Segnali «**standard**» o tradizionali: da 1 a 31 (man 7 signal)
- Gli altri sono i segnali «realtime»
- Un segnale è «**generato**» da un evento
- Una volta generato, un segnale è «**consegnato**» ad un processo…
- …Il processo risponde al segnale effettuando una **azione**
- Nell'intervallo di tempo tra la generazione del segnale la consegna del segnale al processo destinatario, il segnale è «**pendente**»

programma «kill» (man 1 kill)

- Invia un segnale ad un processo (o più di uno); il segnale che manda di default è TERM

kill [options] <pid> […]

-<signal>

-s <signal>

--signal <signal>

- Esempi:

kill -9 234234

Kill –l 9

programma «kill» (man 1 kill)

kill -l

1) SIGHUP 2) SIGINT 3) SIGQUIT 4) SIGILL 5) SIGTRAP

6) SIGABRT 7) SIGBUS 8) SIGFPE 9) SIGKILL 10) SIGUSR1

11) SIGSEGV 12) SIGUSR2 13) SIGPIPE 14) SIGALRM 15) SIGTERM

16) SIGSTKFLT 17) SIGCHLD 18) SIGCONT 19) SIGSTOP 20) SIGTSTP

21) SIGTTIN 22) SIGTTOU 23) SIGURG 24) SIGXCPU 25) SIGXFSZ

26) SIGVTALRM 27) SIGPROF 28) SIGWINCH 29) SIGIO 30) SIGPWR

31) SIGSYS …

Caratteri speciali

- Ctrl-C : invia SIGINT al processo in esecuzione

- Ctrl-Z: invia SIGSTOP al processo in esecuzione
- Comando fg: fa continuare il processo sospeso e lo riporta in «foreground»

- kill –STOP <pid> : stoppa il processo
- kill –CONT <pid> : fa ripartire il processo

Elenco segnali standard

https://replit.com/@MarcoTessarotto/enum-signals

signal 1 **SIGHUP** : Hangup | Hangup

signal 2 **SIGINT** : Interactive attention signal | Interrupt — ctrl-C

signal 3 SIGQUIT : Quit | Quit — come SIGINT ma con core dump

signal 4 SIGILL : Illegal instruction | Illegal instruction

signal 5 SIGTRAP : Trace/breakpoint trap | Trace/breakpoint trap

signal 6 SIGABRT : Abnormal termination | Aborted

signal 7 SIGBUS : Bus error (bad memory access) | Bus error

signal 8 SIGFPE : Erroneous arithmetic operation | Floating point exception

signal 9 **SIGKILL** : Killed | Killed

signal 10 **SIGUSR1** : User-defined signal 1 | User defined signal 1

signal 11 SIGSEGV : Invalid access to storage | Segmentation fault

signal 12 **SIGUSR2** : User-defined signal | User defined signal 2

signal 13 SIGPIPE : Broken pipe | Broken pipe

signal 14 SIGALRM : Alarm clock | Alarm clock

signal 15 **SIGTERM** : Termination request | Terminated

signal 16 SIGSTKFLT : Stack fault | Stack fault

Elenco segnali standard

signal 17 **SIGCHLD** : Child terminated or stopped | Child exited
signal 18 SIGCONT : Continue | Continued
signal 19 SIGSTOP : Stop, unblockable | Stopped (signal)
signal 20 SIGTSTP : Keyboard stop | Stopped
signal 21 SIGTTIN : Background read from control terminal | Stopped (tty input)
signal 22 SIGTTOU : Background write to control terminal | Stopped (tty output)
signal 23 SIGURG : Keyboard stop | Urgent I/O condition
signal 24 SIGXCPU : CPU time limit exceeded | CPU time limit exceeded
signal 25 SIGXFSZ : File size limit exceeded | File size limit exceeded
signal 26 SIGVTALRM : Virtual timer expired | Virtual timer expired
signal 27 SIGPROF : Profiling timer expired | Profiling timer expired
signal 28 SIGWINCH : Window size change (4.3 BSD, Sun) | Window changed
signal 29 SIGPOLL : Pollable event occurred (System V) | I/O possible
signal 30 SIGPWR : Power failure imminent | Power failure
signal 31 SIGSYS : Bad system call | Bad system call

Maschera di segnale (signal mask)

- Un segnale generato viene consegnato al processo cui è destinato:

❑non appena questo è schedulato per eseguire (dallo schedulatore dei processi)

❑Immediatamente se il processo sta eseguendo

- Il processo può specificare che una certa sua parte di codice non va interrotta dalla consegna di un segnale:
- Il segnale va aggiunto alla «**signal mask**»: insieme di segnali la cui consegna è bloccata
- Se un segnale viene generato per un processo mentre è nella sua «signal mask», allora il segnale rimane pendente fino a quando viene sbloccato
- Un processo figlio creato tramite fork eredita una copia della maschera di segnale del genitore

Azioni del processo in seguito a consegna di un segnale

- Alla consegna di un segnale al processo, questo può effettuare una delle seguenti azioni:

❑Il segnale è ignorato

❑Il processo viene terminato

❑Viene prodotto un file contenente il «core dump» per debugging ed il processo viene terminato (il file di core dump contiene una immagine della memoria virtuale del processo)

❑Il processo viene stoppato

❑L'esecuzione del processo riprende (dopo uno stop)

azioni

- Ogni segnale standard ha una **azione di default**
- Il programma può cambiare l'azione da svolgere alla consegna di un segnale (la «**disposizione**» del segnale):

❑Azione di default

❑Ignorare il segnale

❑Eseguire un «signal handler» (gestore di segnale)

Azioni di default (man 7 signal)

Signal	Value	Action	Comment
SIGHUP	1	Term	Hangup detected on controlling terminal or death of controlling process
SIGINT	2	Term	Interrupt from keyboard
SIGQUIT	3	Core	Quit from keyboard
SIGILL	4	Core	Illegal Instruction
SIGABRT	6	Core	Abort signal from abort(3)
SIGFPE	8	Core	Floating-point exception
SIGKILL	9	Term	Kill signal
SIGSEGV	11	Core	Invalid memory reference
SIGPIPE	13	Term	Broken pipe: write to pipe with no readers; see pipe(7)
SIGALRM	14	Term	Timer signal from alarm(2)
SIGTERM	15	Term	Termination signal

Azioni di default (man 7 signal)

Signal	Value	Action	Comment
SIGUSR1	30,10,16	Term	User-defined signal 1
SIGUSR2	31,12,17	Term	User-defined signal 2
SIGCHLD	20,17,18	Ign	Child stopped or terminated
SIGCONT	19,18,25	Cont	Continue if stopped
SIGSTOP	17,19,23	Stop	Stop process
SIGTSTP	18,20,24	Stop	Stop typed at terminal
SIGTTIN	21,21,26	Stop	Terminal input for background process
SIGTTOU	22,22,27	Stop	Terminal output for background process

man 7 signal

- Linux supports the standard signals listed below. **Several signal numbers are architecture-dependent**, as indicated in the "Value" column. (Where **three values** are given, the first one is usually valid for alpha and sparc, the middle one for x86, arm, and most other architectures, and the last one for mips. (Values for parisc are not shown; see the Linux kernel source for signal numbering on that architecture.) A dash (-) denotes that a signal is absent on the corresponding architecture.

Gestore di segnali - signal handler

- Il gestore di segnale è una funzione che svolge dei task in risposta al segnale consegnato
- Il programma notifica il kernel di avere installato un gestore di segnale
- Quando un gestore di segnale viene invocato in risposta alla consegna di un segnale, si dice che il segnale è stato «gestito» o «catturato»

```
void sigHandler(int sig) { … }

if (signal(SIGINT, sigHandler) == SIG_ERR) {
        perror("signal");
…
```

man 2 signal

- SIG_IGN

if (signal(SIGINT, SIG_IGN) == SIG_ERR) { // IGNORE signal

- SIG_DFL
- if (signal(SIGINT, SIG_DFL) == SIG_ERR) { // set to DEFAULT action

System call pause

- pause(2) fa dormire il processo chiamante (o thread) fino a quando non viene consegnato un segnale che termina il processo o provoca l'invocazione di un gestore di segnale (signal handler)
- https://replit.com/@MarcoTessarotto/signal-tests
- pause ritorna solo quando un segnale è stato consegnato (catturato) ed il gestore di segnale è ritornata.
- In questo caso, pause () restituisce -1 e errno è impostato su EINTR.