Laboratorio di Sistemi Operativi

primavera 2009

Segnali: Interrupt software per la gestione di eventi asincroni

Concetto di segnale

- Un segnale è un interrupt software
- Un segnale può essere generato da un processo utente o dal kernel a seguito di un errore software o hardware
- Ogni segnale ha un nome che comincia con SIG (ex. SIGABRT, SIGALARM) a cui viene associato una costante intera ($\neq 0$) positiva definita in **signal.h**
- Il segnale e' un evento asincrono; esso puo' arrivare in un momento qualunque ad un processo ed il processo puo' limitarsi a verificare, per esempio, il valore di una variabile, o può fare cose piu' specifiche

Azione associata ad un segnale

Le azioni associate ad un segnale sono le seguenti:

- **Ignorare il segnale** (tranne che per SIGKILL e SIGSTOP)
- **Catturare il segnale** (equivale ad associare una funzione utente quando il segnale occorre; ex. se il segnale SIGTERM e' catturato possiamo voler ripulire tutti i file temporanei generati dal processo)
- **Eseguire l'azione di default associata** (terminazione del processo per la maggior parte dei segnali)

Segnali in un sistema Unix

Name	Description	ANSI C	POSIX.1	SVR4	4.3+BSD	Default action
SIGABRT	abnormal termination (abort)	•	•	•	•	terminate w/core
SIGALRM	time out (alarm)		•	•	•	terminate
SIGBUS	hardware fault			•	•	terminate w/core
SIGCHLD	change in status of child		job	•	•	ignore
SIGCONT	continue stopped process		job	•	•	continue/ignore
SIGEMT	hardware fault			•	•	terminate w/core
SIGFPE	arithmetic exception	•	•	•	•	terminate w/core
SIGHUP	hangup		•	•	•	terminate
SIGILL	illegal hardware instruction	•	•	•	•	terminate w/core
SIGINFO	status request from keyboard				•	ignore
SIGINT	terminal interrupt character	•	•	•	•	terminate
SIGIO	asynchronous I/O			•	•	terminate/ignore
SIGIOT	hardware fault			•	•	terminate w/core
SIGKILL	termination		•	•	•	terminate
SIGPIPE	write to pipe with no readers		•	•	•	terminate
SIGPOLL	pollable event (poll)			•		terminate
SIGPROF	profiling time alarm (setitimer)			•	•	terminate
SIGPWR	power fail/restart			•		ignore
SIGQUIT	terminal quit character		•	•	•	terminate w/core
SIGSEGV	invalid memory reference	•	•	•	•	terminate w/core
SIGSTOP	stop		job	•	•	stop process
SIGSYS	invalid system call			•	•	terminate w/core
SIGTERM	termination	•	•	•	•	terminate
SIGTRAP	hardware fault			•	•	terminate w/core
SIGTSTP	terminal stop character		job	•	•	stop process
SIGTTIN	background read from control tty		job	•	•	stop process
SIGTTOU	background write to control tty		job	•	•	stop process
SIGURG	urgent condition			•	•	ignore
SIGUSR1	user-defined signal		•	•	•	terminate
SIGUSR2	user-defined signal		•	•	•	terminate
SIGVTALRM	virtual time alarm (setitimer)			•	•	terminate
SIGWINCH	terminal window size change			•	•	ignore
SIGXCPU	CPU limit exceeded (setrlimit)			•	•	terminate w/core
SIGXFSZ	file size limit exceeded (setrlimit)			•	•	terminate w/core

Funzione signal

```
#include <signal.h>

void (*signal(int signo, void (*func)(int)))(int);
```

Restituisce: SIG_ERR in caso di errore e
il puntatore al precedente gestore del segnale se OK

Funzione signal

- prende due argomenti: il nome del segnale *signo* ed il puntatore alla funzione *func* da eseguire come azione da associare all'arrivo di *signo* (*signal handler*) e
- restituisce il puntatore ad una funzione che prende un intero e non restituisce niente che rappresenta il puntatore al precedente *signal handler*

Funzione signal

Il valore di *func* può essere:

- SIG_IGN per ignorare il segnale (tranne che per SIGKILL e SIGSTOP)
- SIG_DFL per settare l'azione associata al suo default
- L'indirizzo di una funzione che sarà eseguita quando il segnale occorre

Funzioni kill e raise

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

int kill (pid_t pid, int signo);
int raise (int signo);
```

Descrizione: mandano il segnale *signo* specificato come argomento

Restituiscono: 0 se OK,
-1 in caso di errore

Funzioni kill e raise

kill manda un segnale ad un processo o ad un gruppo di processi specificato da *pid*

raise consente ad un processo di mandare un segnale a se stesso

- *pid* > 0 invia al processo *pid*
- *pid* == 0 invia ai processi con lo stesso *gid* del processo sender
- *pid* < *0* invia ai processi con *gid* uguale a |*pid*|

Funzione pause

#include <unistd.h>

int pause(void);

Descrizione: sospende il processo finché non arriva un segnale ed il corrispondente signal handler è eseguito ed esce

Restituisce: –1

Funzione sleep

#include <unistd.h>

unsigned int sleep (unsigned int *secs*);


```
#include <signal.h>
void sig_usr(int);
int main (void) {
  signal(SIGUSR1, sig_usr);
  signal(SIGUSR2, sig_usr);
  for(; ;)  pause();
 }
void sig_usr(int signo) {
  if(signo == SIGUSR1) printf("SIGUSR1\n");
  else     if (signo == SIGUSR2)
  printf("SIGUSR2\n");
      else printf("Segnale %d\n",signo);
}
```

uso dell'esempio

- lanciare in background (&) il programma precedente e vedere con che pid gira
- scrivere un programma che manda il segnale a questo processo con

```
kill(pid, SIGUSR1) ;
kill(pid, SIGUSR2) ;
```

- oppure usare kill(1) da linea di comando

```
kill –USR1 pid    //si otterra' SIGUSR1
kill –USR2 pid    //si otterra' SIGUSR2
kill pid          //si sta mandando SIGTERM e con esso il
                    processo termina perche' tale segnale non
                    e' catturato e di default termina
```


esercizio

- scrivere un programma che intercetta il ctrl-C da tastiera e gli fa stampare un messaggio.

Start-up di un programma

L'esecuzione di un programma tramite fork+exec ha le seguenti caratteristiche

- Se un segnale è **ignorato** nel processo padre viene ignorato anche nel processo figlio
- Se un segnale è **catturato** nel processo padre viene assegnata l'azione di default nel processo figlio