DICo

Sistemi Operativi[1]

Mattia Monga

Dip. di Informatica e Comunicazione
Università degli Studi di Milano, Italia
mattia.monga@unimi.it

a.a. 2010/11

DICo

Lezione XVII: System e kernel call

Esercizio

1. Creare un disco virtuale

```
1 qemu-img create prova.img 10M
2
3 dd if=/dev/zero of=prova.img bs=1024 count=10240
4
5 qemu -hda minix -hdb prova.img
```

2. Partizionare il disco (con `part`)
   `c0d1`: primo controller IDE (`c0`), secondo disco (`d1`)
3. Creare il file system
   `mkfs /dev/c0d1p0`
4. Montare il file system
   `mount /dev/c0d1p0 /mnt`

Segnali

```
1  /* Regular signals. */
2  #define SIGHUP 1 /* hangup */
3  #define SIGINT 2 /* interrupt (DEL) */
4  #define SIGQUIT 3 /* quit (ASCII FS) */
5  #define SIGILL 4 /* illegal instruction */
6  #define SIGTRAP 5 /* trace trap (not reset when caught) */
7  #define SIGABRT 6 /* IOT instruction */
8  #define SIGBUS 7 /* bus error */
9  #define SIGFPE 8 /* floating point exception */
10 #define SIGKILL 9 /* kill (cannot be caught or ignored) */
11 #define SIGUSR1 10 /* user defined signal # 1 */
12 #define SIGSEGV 11 /* segmentation violation */
13 #define SIGUSR2 12 /* user defined signal # 2 */
14 #define SIGPIPE 13 /* write on a pipe with no one to read it */
15 #define SIGALRM 14 /* alarm clock */
16 #define SIGTERM 15 /* software termination signal from kill */
17 #define SIGEMT 16 /* EMT instruction */
18 #define SIGCHLD 17 /* child process terminated or stopped */
19 #define SIGWINCH 21 /* window size has changed */
```

Segnali


```
1 /* POSIX requires the following signals to be defined, even if they are
2  * not supported. Here are the definitions, but they are not supported.
3  */
4 #define SIGCONT 18 /* continue if stopped */
5 #define SIGSTOP 19 /* stop signal */
6 #define SIGTSTP 20 /* interactive stop signal */
7 #define SIGTTIN 22 /* background process wants to read */
8 #define SIGTTOU 23 /* background process wants to write */
```

Mandare segnali

- I segnali possono essere emessi col programma `kill`
- È possibile "intrappolare" i segnali ricevuti, ossia eseguire una routine di risposta

```
1  #!/bin/sh
2
3  trap "echo Ho ricevuto il segnale USR1" 10
4  trap "echo Ho ricevuto il segnale USR2" 12
5  trap "echo Ho ricevuto il segnale INT" 2
6
7  while true ; do
8      sleep 1
9      echo "ciao: sono $(id)"
10 done
```

- `SIGKILL` non può essere catturata.

Make

Stuart Feldman, 1977 at Bell Labs.
Permette di specificare dipendenze fra processi di generazione.
Dipendenze: se cambia questo file, allora il processo di generazione deve essere ripetuto.

```
1 helloworld: helloworld.o
2         cc -o $@ $<
3
4 helloworld.o: helloworld.c
5         cc -c -o $@ $<
6
7 .PHONY: clean
8 clean:
9         rm helloworld.o helloworld
```