# Gestione Processi

Mattia Rizzolo
mattia@mapreri.org


Corso Linux Avanzato 2015

# Processes?

# Process

attività controllata da un programma che si svolge su un processore in genere sotto la gestione o supervisione del rispettivo sistema operativo

# metadata

Ogni processo ha dei metadata:

- PID
- PPID
- UID/GID
- EUID/EGID
- Parent
- Times
- State
- Priority
- Resources

# birth

- `fork()` or `clone()` system calls
  - ritorna `0` al figlio
  - ritorna il pid del figlio al padre
  - il figlio con un `exec()` esegue qualcosa di diverso da se stesso

# death

- `exit()`
  - il padre raccoglie il codice d'uscita e si comporta di conseguenza
  - IFF il padre è ancora vivo...

# something near to you...

- Ogni riga che si scrive al terminale causa un nuovo processo
- la variabile ? contiene il codice d'uscita dell'ultimo comando
  - `echo $?`
- gli operatori booleani || e && controllano proprio quel valore quando confrontano dei comandi
  - `apt-get update && apt-get upgrade`

# init

è il primo processo avviato dal sistema, padre di tutti gli altri processi

# init

- PID == **1**
- adotta tutti i figli orfani
- è l'unico processo orfano ammesso

# states

- **R** running/runnable
- **D** uninterruptible sleep
- **S** interruptible sleep
- **T** stopped
- **Z** zombie

# backgrounds jobs

La maggior parte dei processi è eseguita in background, cioà senza che la shell venga occupata da quel processo.

In una shell testuale:

- `<command> &`: spawn del programma e passaggio in background
- CTRL+Z per mandare un SIGTSTP[1] e tornare alla shell. `bg` per far continuare il programma in background
- `jobs` per mostrare i processi in background
- `fg` porta un processo in foreground

[1] more on this later...

# signals

È un tipo di IPC usato per far modificare ai processi (più o meno forzatamente) il loro stato.

- identificati con un numero da 1 a 64
- sono usati all'interno dei programmi per modificare lo stato dei propri thread
- la maggior parte dei segnali vengono gestiti dal processo (se hanno una routine per farlo), alcuni vengono controllati esclusivamente dal kernel (e.g. SIGKILL)
- sono usati dal kernel
- possono essere usati dall'utente

# signals

| value | signal | description |
|---|---|---|
| 1 | **SIGHUP** | Hangup detected |
| 2 | **SIGINT** | C Interrupt from keyboard |
| 3 | **SIGQUIT** | Quit from keyboard |
| 9 | **SIGKILL** | Kill signal |
| 15 | **SIGTERM** | Termination signal |
| 18 | **SIGCONT** | Continue if stopped |
| 19 | **SIGSTOP** | Stop process |
| 20 | **SIGTSTP** | Z Stop typed at terminal |

Fonte:
signals(7)

# sending signals

via terminale:

- `kill [-<signal>] <PID>`
- `kill -9 12345`
- `kill -SIGKILL 1234`
- di default `kill` manda un SIGTERM

# sending signals

via codice:

- **C**: `int kill(pid_t pid, int sig);`
- **python**: `os.kill(pid, sig)`

# killall

Comando che manda un segnale (SIGTERM di default) a tutti processi con un dato nome.

# nice

Modifica la priorità di un processo (quindi lo scheduling)

- varia da -20 a +19
- niceness alta == bassa priority
- niceness bassa == alta priorità
- viene ereditato dal processo padre
- l'owner del processo può alzare il valore
- solo root può abbassare il valore

# nice(1) e renice(1)

Sono i due programmi che permettono di operarare sulla niceness di un processo:

- nice 5 process-name
- renice -10 another-process
- renice 15 -u username

# /proc

File system virtuale con informazioni sui processi (e non solo)

C'è una directory per ogni processo, chiamata dopo il PID.

- **cmdiline** comando con cui il processo è stato invocato
- **cwd** symlink alla directory di lavoro del processo
- **fd/** directory contente i file descriptor attualmente aperti
- molto molto altro, vedi `man 5 proc`

# ps

Mostra le informazioni sui processi, generalmente usato insieme a grep.

Stampa UID, PID, ora/data di avvio del processo, comando, stato, tempo di esecuzione, e qualsiasi altra cosa gli si chieda.

Un uso comune consiste nel chiamarlo con le opzioni `aux` o `-ef` per avere un output con qualche dato.

```
man 1 ps
```

# top

- monitori i processi in tempo reale con un'interfaccia ncurses
- refresh ogni 3 secondi
- si possono mandare segnali e modificare il nice
- molto personalizzabile: `man 1 top`

# top

```
last pid: 80032;  load averages:  0.84,  1.06,  1.05   up 199+03:28:06 22:26:35
107 processes: 1 running, 106 sleeping
CPU:  0.2% user,  5.0% nice,  1.0% system,  0.0% interrupt, 93.7% idle
Mem: 7938M Active, 3027M Inact, 2359M Wired, 683M Cache, 1643M Buf, 1842M Free
Swap: 8192M Total, 821M Used, 7371M Free, 10% Inuse

  PID USERNAME    THR PRI NICE   SIZE    RES STATE   C   TIME   WCPU COMMAND
79957 www           1  52    8   356M 44704K accept  6   0:02 13.87% php-cgi
79477 www           1  52    8   360M 62256K accept  3   0:30  9.18% php-cgi
79471 www           1  52    8   356M 47792K accept  7   0:03  9.18% php-cgi
79476 www           1  34    8   356M 56140K accept  1   0:14  5.96% php-cgi
79933 www           1  40    8   356M 47340K accept  2   0:02  5.96% php-cgi
79958 pgsql         1  21    0  6399M   164M sbwait  0   0:00  2.10% postgres
79490 pgsql         1  21    0  6413M   652M sbwait  4   0:14  1.76% postgres
79474 www           1  28    8   360M 58620K sbwait  0   0:21  1.17% php-cgi
79475 www           1  30    8   360M 49396K accept  4   0:04  1.07% php-cgi
79934 pgsql         1  20    0  6407M   187M sbwait  0   0:00  0.98% postgres
79480 pgsql         1  20    0  6407M   397M sbwait  2   0:02  0.88% postgres
79482 pgsql         1  22    0  6403M   215M sbwait  3   0:01  0.88% postgres
79483 pgsql         1  20    0  6405M   495M sbwait  7   0:05  0.78% postgres
79470 www           1  30    8   356M 47476K accept  0   0:04  0.29% php-cgi
 1321 nobody        4  52    0   152M 85708K uwait   4  55.3H  0.10% memcached
79472 www           1  30    8   356M 46868K accept  6   0:02  0.10% php-cgi
 1308 pgsql         1  20    0  6391M  5328M select  1 287:40  0.00% postgres
```

Image by Ivan Voras - Licensed under Public Domain via Wikimedia Commons - https://commons.wikimedia.org/wiki/File:BSD-unix-top-plain.png#/media/File:BSD-unix-top-plain.png

# htop

- versione figa e colorata di top
- e più facile da usare, ovviamente

# htop

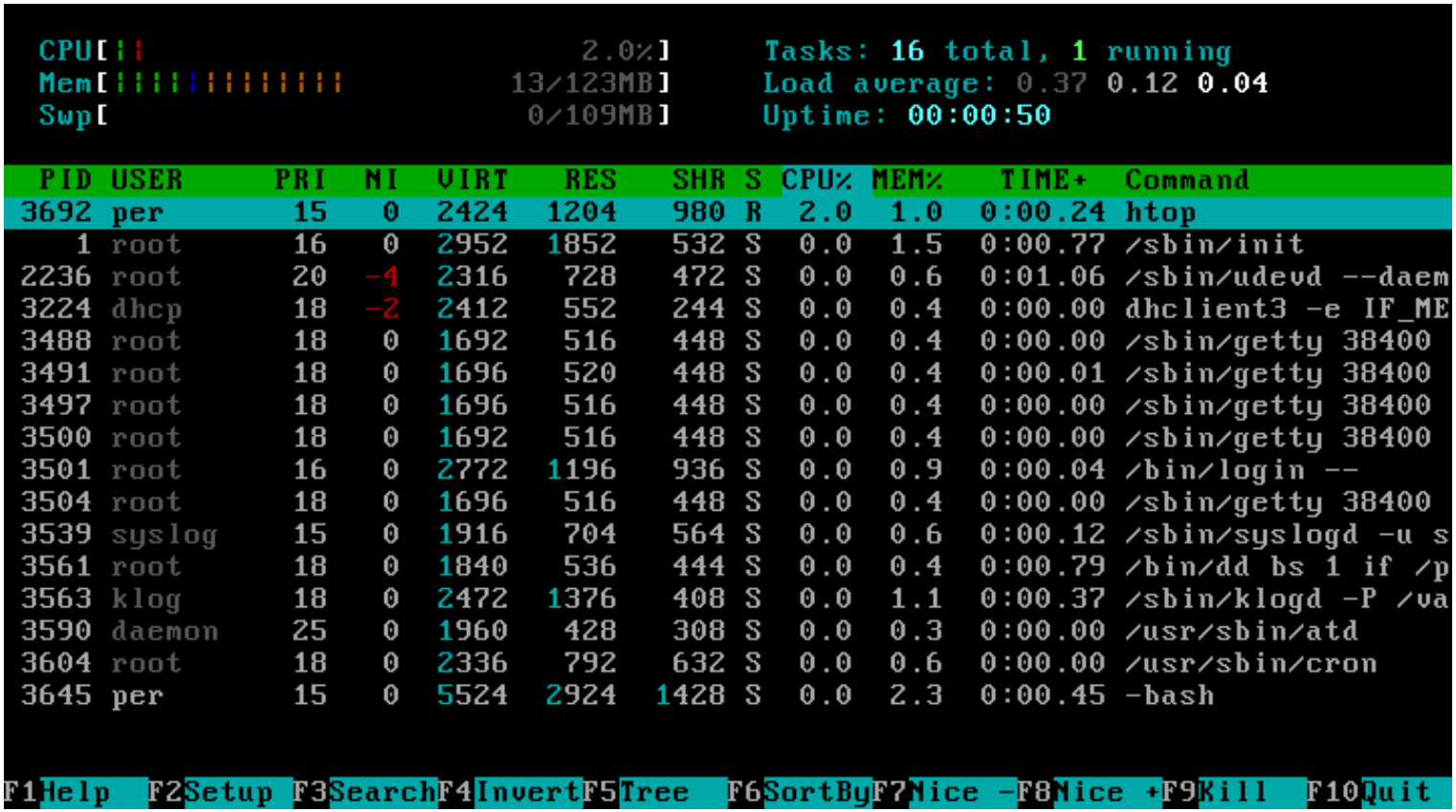



Image by Erik Strandberg - Licensed under CC BY-SA 3.0 via Wikimedia Commons - https://commons.wikimedia.org/wiki/File:Htop.png#/media/File:Htop.png

# lsof

- mostra i file aperti da un processo
  - `lsof -p <PID>`
- mostra i processi che hanno file aperti in una directory
  - `lsof +D <directory>`

# Processes

Questions?!?

# Do you want more cookies?

~~RTFM and stop ranting~~

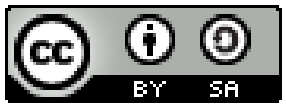
1. Chiedi al tuo computer: `man chmod`, per esempio.
   - se non riesci a usare man: `man man`
2. GIYF[2]
3. wiki della tua distribuzione
   - Hint: anche se non è della distribuzione che stai usando va bene lo stesso.
4. chiedi nei canali di supporto della tua distribuzione
5. chiedi a noi ;)

[2] Google Is Your Friend

# Thanks

- POuL, for the organization
- **YOU**, for attending
- Holger Levsen <holger@debian.org> and Jérémy Bobbio <lunar@debian.org> for the template used for these slides
- Slides degli anni precedenti. Thanks to:
  - Daniele Iamartino (2011)
  - Pietro Virgilio (2012)
  - Radu Andries (2013)
  - Stefano Bouchs (2014)
- chiunque si sia sbattutto per scrivere le pagine di manuale!!!!
- Linux for being so funny

# Bye o/






| | |
|---|---|
| email | mattia@mapreri.org |
| GPG key | 66AE 2B4A FCCF 3F52 DA18 4D18 4B04 3FCD B944 4540 |