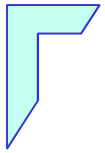
Lab. di Sistemi Operativi
- Lezione in aula -
a.a. 2012/2013

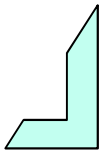
"Processi bash"

Sommario

- I processi
- Comando ps (process status)
- Terminazione di un processo
  - CTRL-C
  - Kill
- Controllo dei processi
  - Processi in background
  - Jobs e Processi
- Monitoraggio e Memoria



- I processi -



I processi

- Sono **programmi** in esecuzione (parte dinamica di un programma)
- Lo stesso programma può corrispondere a diversi processi (lanciato da utenti diversi)

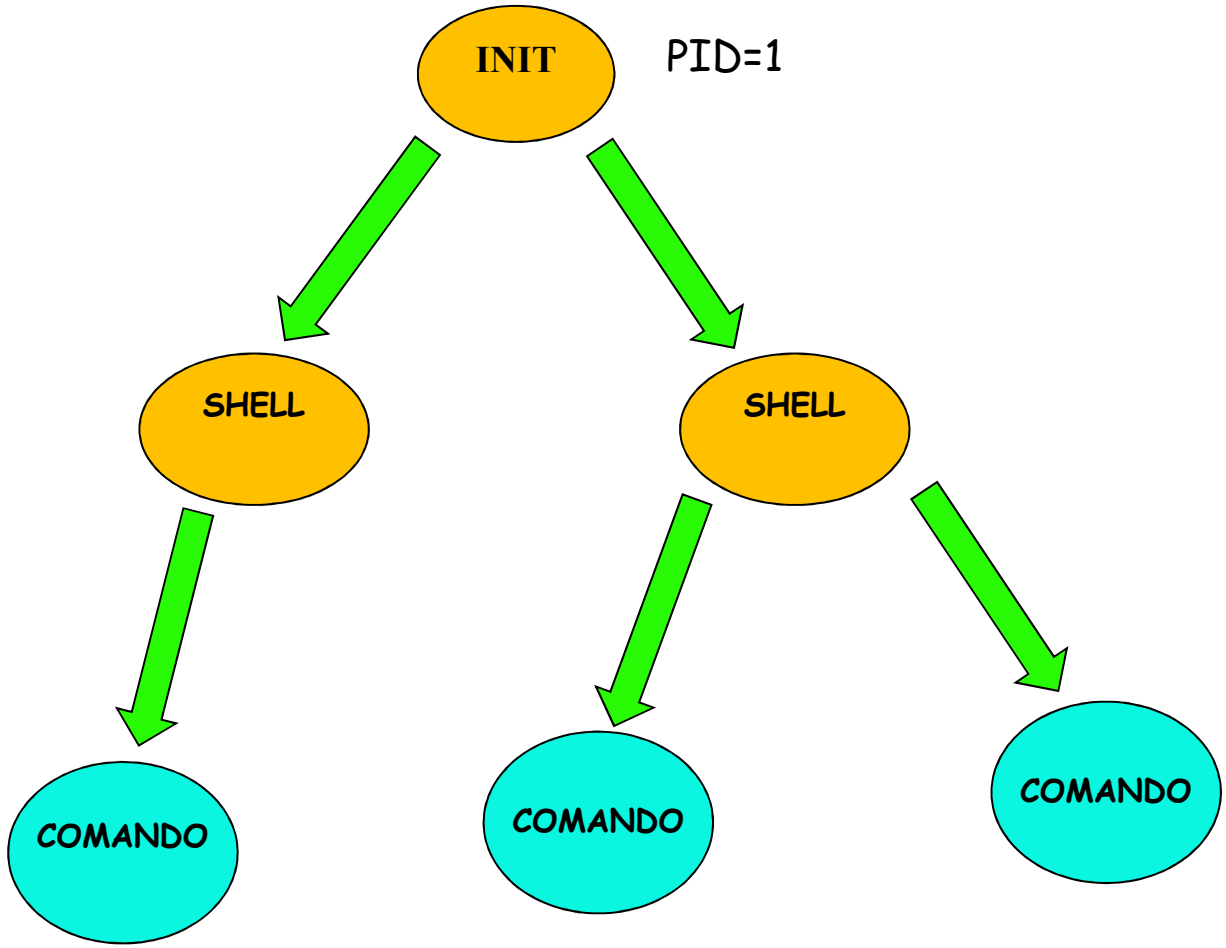
- Ogni processo può generare nuovi processi (figli) ed è caratterizzato da:
  - **PID** (identificativo del processo)
  - **PPID** (identificativo del figlio generato dal processo)

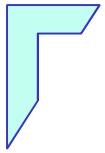
- I processi si rappresentano con uno schema gerarchico dove al vertice c'è il processo INIT che è l'unico processo ad avere un PID=1 e nessun PPID

- INIT è il processo che parte al boot del sistema creato dal processo principale chiamato root

I processi

- Modello gerarchico dei processi organizzata ad albero:





- Tabella dei processi ed Attributi -

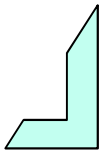


Tabella dei processi

- Il kernel gestisce una tabella dei processi che serve a tenere traccia del loro stato. In particolare sono registrati i valori seguenti:
  1. il nome dell'eseguibile in funzione che ha generato il processo;
  2. gli eventuali argomenti passati all'eseguibile al momento dell'avvio attraverso la riga di comando
  3. il numero di identificazione del processo (PPID)
  4. il numero di identificazione del processo che ha generato quello a cui si fa riferimento (PID)
  5. il nome del dispositivo di comunicazione se il processo controllato da un terminale
  6. il numero di identificazione dell'utente; (uid)
  7. il numero di identificazione del gruppo; (gid)

Attributi dei processi

- Ogni processo possiede vari ID
  - **real user id:** utente che ha lanciato il processo (uid)
  - **real group id:** gruppo a cui appartiene l'utente che ha lanciato il processo (gid)
  - **effective user id**: utente che determina i diritti del processo ad accedere al file system
  - **effective group id**
  - **set-user-id e set-group-id:** contengono una copia di **effective user id e di effective group id**

- Di solito, i due utenti coincidono cioè
  - **real user id = effective user id**
  - **real group id = effective group id**
- ovvero, i diritti di accesso sono determinati dall' utente che esegue il programma

Attributi dei processi

- In alcune situazioni, è desiderabile un comportamento diverso:
  - potrebbe essere necessario che un processo abbia (in un dato momento) diritti maggiori di chi esegue il programma

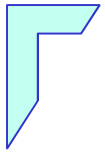
Esempio:

comando passwd (cut /etc/passwd)

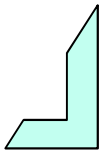
- Chiu **swd**
- Si agisce su flag set-user-id e set-group-id :
  - se il flag **set-user-id** è attivo, quando il file viene eseguito: effective user id = st_uid
  - se il flag **set-group-id** è attivo, quando il file viene eseguito: effective group id = st_gid

Attributi dei processi

- Quest i flag risolvono il problema di **passwd**
- l'owner del "comando passwd" è root:
  - quando passwd viene eseguito, il suo **effective user id** è uguale a **root**
  - il comando può accedere in scrittura al file **/etc/passwd**



- Comando ps (process status) -



Comando ps

- Il comando ps fornisce i processi presenti nel sistema

Sintassi:

```
ps [selezione] [formato]
```

- Selezione:
  - niente processi lanciati dalla shell corrente
  - -u pippo i processi dell'utente pippo
  - -a (All) tutti i processi

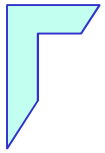
- Formato:
  - niente PID, terminale, ora di esecuzione, comando
  - -f (full) anche UID, PPID, argomenti
  - -F (Full) anche altro

Esempio: comando ps

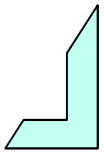
- Esempio di utilizzo:

```
user> ps   # fornisce i processi dell'utente associati al terminale corrente

   PID          TTY          TIME     CMD
 23228          pts/15       0:00     xdvi.bin
  9796          pts/15       0:01     bash
 23216          pts/15       0:04     xemacs-2
  9547          pts/15       0:00     csh
```

Legenda: `PID` = PID; `TTY` = terminale (virtuale); `TIME` = tempo di CPU utilizzato; `CMD` = comando che ha generato il processo.



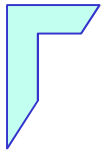
- Terminazione di un processo: Ctrl-c kill -



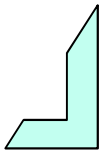
Terminazione di un processo

- Per arrestare un processo in esecuzione si può utilizzare:
  - la sequenza **Ctrl-c** dal terminale stesso su cui il processo è in esecuzione
  - il comando **kill** seguito dal PID del processo (da qualsiasi terminale):

```
user> ps

  PID    TTY      TIME CMD
      .......
  28015  pts/14   0:01   man
      .......

user> kill 28015    Uccide il processo
```



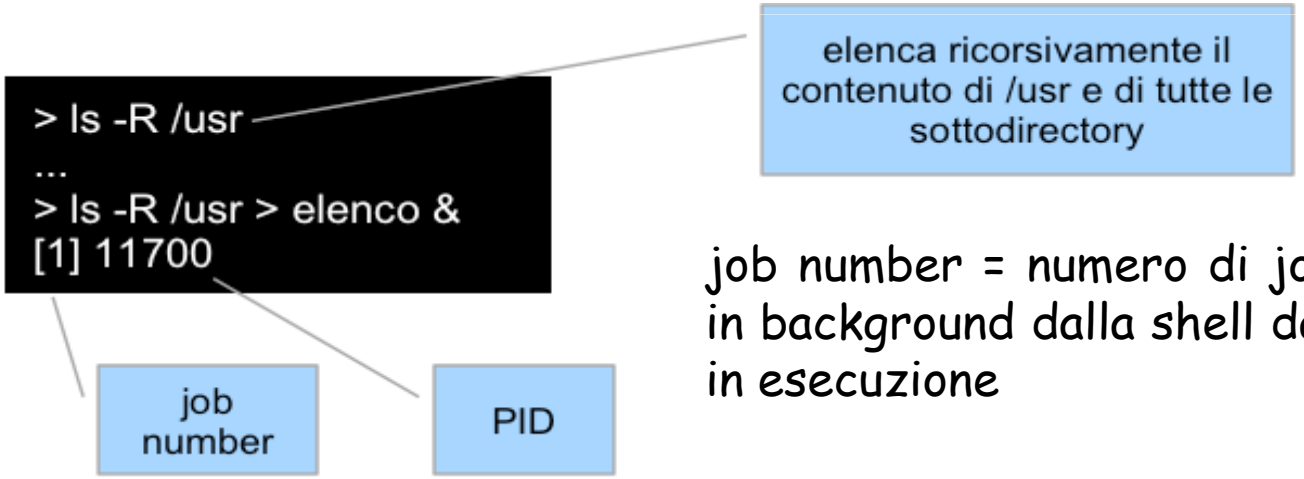
- Controllo dei processi -



Controllo dei processi

- Normalmente, la shell aspetta che ogni comando termini (comando in foreground)
- Con "comando&", (e commerciale) la shell non aspetta (comando in background)
  - Il comando può comunque scrivere su standard output

- Esempio



job number = numero di job lanciato in background dalla shell da quando è in esecuzione

Processi in background

- I processi in background sono eseguiti in una sottoshell, in parallelo al processo padre (la shell) e non sono controllati da tastiera.
- I processi in background sono quindi utili per eseguire task (Unix multi-tasking) in parallelo che non richiedono controllo da tastiera.

```
user>   man  &      lancio di un comando in background
  [1] 24760
```

- "1" il numero di job (i job sono processi gestiti dalla shell corrente),
- mentre "24760" il numero di processo (i processi sono gestiti dal sistema operativo).
- Per terminare questo job/processo si può utilizzare sia **kill %1** che **kill 24760**.

jobs e processi

- L'avvio da linea di comando di un singolo processo e' un caso particolare, quello in cui l'insieme contiene un solo elemento
- Per avviare un job in background occorre terminare la linea di comando con il carattere & (e commerciale):

```
> cmd1|cmd2|……|cmdN &
```

- Questa procedura produce come output immediato la stampa a video di un numero fra parentesi quadre che identifica il job avviato in background, dopodichè ricompare il prompt: la shell è pronta a eseguire in parallelo altri comandi.
- Su ogni terminale è possibile avviare molti job in background ma uno solo può eseguire in foreground

jobs e processi

- Non si deve **confondere** un job di shell con un processo
- Un comando impartito attraverso una shell può generare più di un processo, per esempio quando viene avviato un programma o uno script che avvia a sua volta diversi programmi.

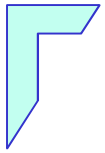
- Un job di shell rappresenta tutti i processi che vengono generati da un comando impartito tramite la shell stessa.
- Cioè corrisponde ad un gruppo di processi connessi da una pipe:

- Il job identifica → cmd1|cmd2|.....|cmdN l'insieme degli N processi avviati (corrispondenti ciascuno ad un comando), ciascun processo avra' comunque un proprio PID

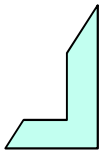
Controllo dei processi

Un job si può **sospendere** e poi **rimandare in esecuzione**

```
user> cat >temp    # job in foreground

   Ctrl-z   # sospende il job

  [1]+ Stopped

user> jobs
  [1]+ Stopped     cat >temp

user> fg    # fa il resume del job in foreground

   Ctrl-z   # sospende il job

user> bg    # fa il resume del job in background

user> kill %1   # termina il job 1
  [1]+ Terminated
```



- Monitoraggio e Memoria -

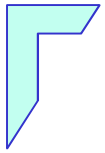


Monitoraggio e Memoria: comando top

Il comando `top` fornisce informazioni sulla memoria utilizzata dai processi, che vengono aggiornate ad intervalli di qualche secondo. I processi sono elencati secondo la quantità di tempo di CPU utilizzata.

```
user> top
load averages: 0.68,  0.39, 0.27 14:34:55
245 processes: 235  sleeping, 9 zombie, 1 on cpu
CPU states: 91.9% idle, 5.8% user, 2.4% kernel, 0.0% iowait, 0.0% swap
Memory: 768M real, 17M free, 937M swap in use, 759M swap free

  PID USERNAME THR PRI NICE  SIZE   RES STATE    TIME    CPU COMMAND
12887 root       1  59    0   65M   56M sleep  105:00  3.71% Xsun
 4210 pippo      1  48    0 2856K 2312K cpu      0:00  1.50% top
 9241 root       1  59    0   35M   26M sleep   15:58  1.47% Xsun
24389 pluto      4  47    0   28M   25M sleep   16:30  0.74% opera
  ......
```

Legenda: la prima riga indica il carico del sistema nell'ultimo minuto, negli ultimi 5 minuti, negli ultimi 15 minuti, rispett.; il carico è espresso come numero di processori necessari per far girare tutti i processi a velocità massima; alla fine della prima riga c'è l'ora; la seconda contiene numero e stato dei processi nel sistema; la terza l'utilizzo della CPU; la quarta informazioni sulla memoria; le restanti righe contengono informazioni sui processi (THR=thread, RES=resident)



- Fine Lezione -