Linux e la shell Bash

- Espressioni regolari
  - Comandi: *grep, find, sed*
- Filtri
- Esercizi
- Controllo dei processi
  - Comandi: *yes, bg, fg, sleep, wait, sh, jobs, ps, top, kill, killall*

Testi di riferimento:

- Linux e la shell Bash http://sisinflab.poliba.it/ruta/linux/Linux e la shell Bash.pdf
- La shell Bash http://www-ictserv.poliba.it/piscitelli/doc/lab_esercitazioni_SO_no/bash_shell.pdf
- Comandi shell Bash http://sisinflab.poliba.it/giannini/SO/Comandi_shell_bash.pdf

Espressioni regolari

- Un'espressione regolare è un pattern che rappresenta un insieme di stringhe.

SINTASSI

- **{a}** il caso più elementare di espressione regolare è un singolo carattere, che rappresenta l'insieme delle stringhe costituite dal solo carattere.
- **[xwyz]** qualsiasi carattere contenuto nelle parentesi quadre (OR)
- **[^xwyz]** qualsiasi carattere non contenuto nella lista

  Il carattere **^**, se specificato dopo il primo carattere, assume significato letterale.
- Due espressioni regolari, **regex1** e **regex2**, possono essere concatenate
  - in una nuova espressione regolare **regex1regex2** → qualsiasi stringa formata da due stringhe concatenate corrispondenti alle singole espressioni di partenza.
  - utilizzando l'operatore **pipe |** → qualsiasi stringa che corrisponda alla prima o alla seconda. **regex1|regex2** è l'unione degli insiemi delle stringhe rappresentate da **regex1** e **regex2**
- I caratteri **^** e **$** identificano all'interno di un'espressione regolare rispettivamente inizio e fine riga.
- I simboli **\<** e **\>** individuano rispettivamente l'inizio e la fine di una stringa.

Esempi espressioni regolari

- L'espressione regolare '[^abcd]' mostra le linee del file che non contengono i caratteri a,b, c e d.
- L'espresione regolare '[a^b]' mostra le linee del file che contengono i caratteri a,^ e b e non caratteri a seguiti da caratteri diversi da b.
- L'espresione regolare '[a-g]' mostra tutte le righe del file che contengono una lettera compresa tra a e g.
- L'espressione regolare 'ab' restituisce tutte le righe che contengono la stringa ab.
- L'espressione regolare 'a|b' visualizza le righe che contengono le stringhe che contengono la lettera a e quelle che contengono la lettera b.
- L'espressione regolare 'a[1-5]' restituisce tutte le righe contenenti una delle seguenti stringhe: {a1,a2,a3,a4,a5}.
- L'espressione regolare 'stringa$' restituisce tutte le righe che terminano con stringa.
- L'espressione regolare '^stringa' restituisce tutte le righe che iniziano con stringa.
- L'espressione regolare '^stringa$' restituisce le righe costituite solo dalla parola stringa.
- L'espressione regolare '\<stringa' restituisce tutte le righe che contengono una parola che comincia con stringa (ad es.: stringa, stringato, ma non costringa).
- L'espressione regolare 'stringa\>' restituisce tutte le righe che contengono una parola che termina con stringa (ad es.: stringa, costringa, ma non stringato).

Regole sintattiche – classi di caratteri

[:**upper**:]
lettere maiuscole

[:**lower**:]
lettere minuscole

[:**alpha**:]
lettere alfabetiche

[:**digit**:]
cifre numeriche

[:**alnum**:]
caratteri alfanumerici

[:**punct**:] Caratteri di punteggiatura

[:**space**:] Caratteri definiti come «spazi bianchi»

[:**blank**:] Comprende solo <space> e <tab>

ESEMPIO [[:lower:]] Seleziona tutte le righe che contengono (almeno) una minuscola.

Regole sintattiche

- **. (punto)** carattere qualsiasi
- **[xwyz]** Set di caratteri qualsiasi all'interno delle parentesi quadre, I caratteri sono in OR tra di loro
- **[a-f]** Un carattere qualsiasi tra a ed f, estremi inclusi
- **[^a-f]** Tutti i caratteri tranne quelli specificati
- **^[a-f]** Cerca il modello specificato all'inizio di ogni riga
- **[a-f]$** Cerca il modello specificato alla fine di ogni riga
- **\<[a-f]** Cerca il modello specificato all'inizio di ogni parola
- **[a-f]\>** Cerca il modello specificato alla fine di ogni parola
- **y{n,m}** Ripetizione del carattere da n a m volte. n ed m possono anche non essere definiti
- **y*** Ripetizione del carattere da 0 a più volte (Eq. a y{0,})
- **y?** Al massimo una ripetizione (Eq. a y{0,1})
- **y+** Almeno una ripetizione (Eq. a y{1,})

[^a-f] è diverso da ^[a-f]

grep

- **grep**: (global regular expression print) Mostra le linee che corrispondono al modello ricercato. Si possono specificare uno o più file oppure è possibile usare lo standard input

SINTASSI

grep [opzioni] modello [file_1 file_2]

ESPRESSIONI REGOLARI

1) Un qualsiasi carattere è già un'espressione regolare
2) Un carattere speciale può essere usato come carattere regolare anticipandolo con backslash (\) (es. * viene considerato asterisco con *)
3) L'espressione regolare va racchiusa tra apici (es. grep 'expr').

grep (flag)

- **-G** Utilizza l'espressione regolare elementare (comportamento predefinito)
- **-E** Utilizza l'espressione regolare estesa. Aggiunge ulteriori metacaratteri alla serie di base
- **-F** Utilizza un modello fatto di stringhe fisse
- **-e** Specifica il modello di ricerca
- **-c** Mostra il numero di righe in cui è presente il modello
- **-n** Aggiunge ad ogni riga contenente il modello il numero di tale riga nel file
- **-v** Selezione inversa: mostra tutte le righe tranne quelle contenenti il modello
- **-l** Mostra il percorso dei file in cui è presente il modello
- **-L** Selezione inversa: mostra il percorso di tutti i file in cui non è presente il modello
- **-i** Non distingue tra caratteri maiuscoli e minuscoli di modello

I caratteri **?**, **+**, **{**, **}**, **|**, **(**, e **)**, se utilizzati con **grep** che interpreta espressioni regolari **base** (in assenza dell'opzione **-E**), vanno preceduti dal carattere di escape ****.

grep - Esempi

- Mostra tutte le righe di file_esempio

```
grep '.*' file_esempio
```

```
15 fifteen
14 fourteen
13 thirteen
12 twelve
11 eleven
10 ten
9 nine
8 eight
7 seven
6 six
5 five
4 four
3 three
2 two
1 one
```

grep - Esempi

1) grep '[24]' file_esempio

14 fourteen
12 twelve
4 four
2 two

Mostra tutte le righe di file_esempio contenenti i caratteri 2 oppure 4, indipendentemente dalla loro collocazione nella stringa.

2) grep 'ee.' file_esempio

grep 'e\{2\}.' file_esempio

grep -E 'e{2}.' file esempio

15 fifteen
14 fourteen
13 thirteen

Mostra le righe che contengono due "e" di seguito seguite da un carattere qualsiasi.

grep - esempi

3) grep '^1.*e$' file_esempio.txt

12 twelve
1 one

Mostra le righe che:

^1 Iniziano con il carattere "1"
e$ Terminano con il carattere "e"
.* Presentano al loro interno zero o più caratteri (qualsiasi)

4) grep '\<f.*n\>' esempio_grep.txt

15 fifteen
14 fourteen

Mostra le righe contenenti parole che:

\<f Iniziano con il carattere "f"
n\> Terminano con il carattere "n"
.* Presentano al loro interno zero o più caratteri

find

find: cerca dei file nella directory specificata e relative subdirectory.

SINTASSI

find [-HLP] [percorso] [-name 'modello'] [-user utente_1]
[-group gruppo_a] [-perm +-permessi] [-max(min)depth X]
[-mount] [-[mac]time n] [-[mac]min n] [-[nac]ewer file]
[-empty] [-links n] [-inum n] [-size n[bckMG]]

-name modello	Restituisce i file aventi nel nome la stringa modello.
-user utente_1	Cerca i file in percorso aventi come utente proprietario utente_1
-group gruppo_a	Cerca i file in percorso aventi come gruppo proprietario gruppo_a
-maxdepth X	Prosegue la ricerca massimo fino al livello X, con X intero non negativo
-mindepth X	Prosegue la ricerca a partire dal livello X
-[mac]time n	Trova i file o directory la cui data di modifica (mtime), accesso (atime) o creazione (ctime) corrisponde a n giorni fa
-[mac]min n	Come l'opzione precedente ma n corrisponde a minuti
-empty	Solo file e directory non vuoti
-size n[bckMG]	File che usano al massimo n unità di spazio

find - flag

(644) -rw-r--r-- esempio_sort_num.txt

(646) -rw-r--rw- esempio_sort.txt

(644) -rw-r--r-- esempio_uniq.txt

(664) -rw-rw-r- file_esempio.txt

- find -perm 644

  ./esempio_uniq.txt

  ./esempio_sort_num.txt

  Restituisce i file che hanno esattamente i permessi specificati, in modo ottale o simbolico

- find -perm -242

  ./esempio_sort.txt

  Restituisce i file che hanno almeno i permessi specificati, in modo ottale o simbolico

- find -perm +022

  ./esempio_sort.txt

  ./file_esempio.txt

  Restituisce i file che hanno alcuni i permessi specificati, in modo ottale o simbolico. Almeno uno tra quelli indicati.

sed

sed: Esegue modifiche sui dati ricevuti dallo standard input, lavorando con stringhe di qualsiasi lunghezza.

SINTASSI

sed [-n] [-e script] [file]

FLAG

- -n Il normale output non viene mostrato
- -e Viene usata l'espressione script per elaborare il testo

ESEMPI

sed -e '/regexp/d' file	Elimina (d) tutte le righe corrispondenti all'espressione regolare regexp
sed -e 'n,md' file	Elimina (d) le righe dalla n alla m incluse.
sed –n -e '/regexp/p' file	Visualizza (p) solo le righe corrispondenti all'espressione regolare regexp (-n).
sed -e 's/stringa1/stringa2/g' file	Sostituisce (s) ogni occorrenza di stringa1 con stringa2. Se g non è presente viene sostituita sola la prima occorrenza di ogni riga.

Esercizi

1) Supponendo di trovarsi in una working directory qualsiasi, scrivere il comando per poter listare il contenuto della home directory (compresi i file nascosti), redirezionando l'output nel file *output.out*

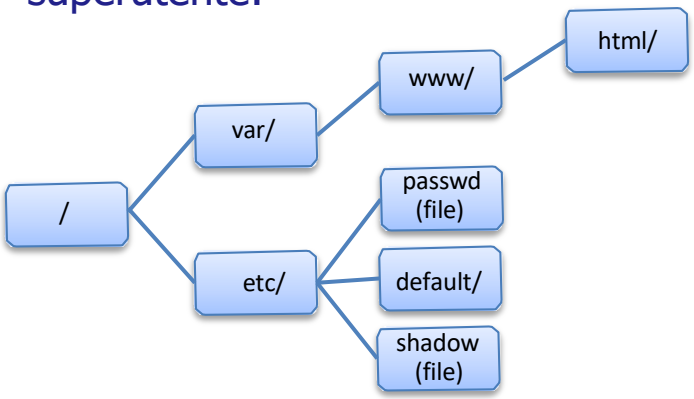
Soluzione
- ls -a ~/ > output.out

2) Si supponga che la current working directory sia ~/dir1. Descrivere come copiare il file ~/dir1/myfile nella home directory, evitando di sovrascrivere un file esistente.

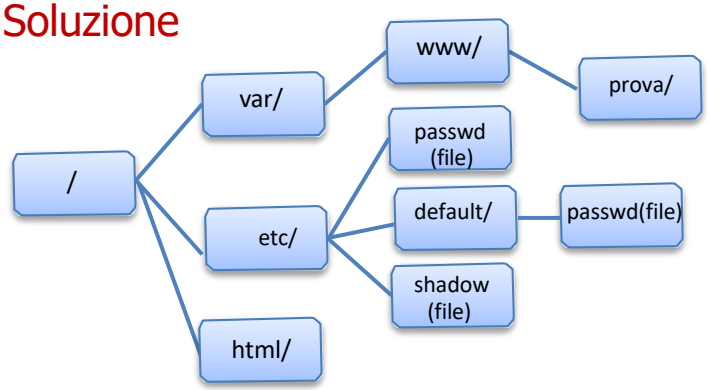
Soluzione
- cp -i ./myfile ../

Equivalente a:
- cp -i ./myfile ~/
- cp -i ~/dir1/myfile ~/

3) Sia assegnato il seguente albero di directory e si supponga di avere i privilegi di superutente:



Soluzione



Dire come l'albero risulta modificato dopo la seguente sequenza di comandi:

- cd /etc/default: ci posizioniamo nella cartella /etc/default
- cp ../pa* ./: copia il file passwd nella directory corrente /etc/default
- cd /var/www: ci posizioniamo nella cartella /var/www
- mv ./html /: sposta la cartella html in root /
- touch ./html: aggiorna l'orario di accesso e modifica con l'orario corrente
- mkdir prova: crea la cartella prova nella directory corrente /var/www
- cd –: ci posizioniamo nella cartella precedente /etc/default
- pwd: visualizza la working directory /etc/default

Esercizi

4) Scrivere il comando UNIX per visualizzare in ordine alfabetico i file contenuti nella directory corrente e produrre il risultato nel file di nome *fileA* nella root directory.

Soluzione
- ls | sort > /fileA

5) Si dica quale genere di output produce il comando seguente:
- $grep – E '^.*\<7[3-9]' ./prova?

Soluzione
- La grep viene eseguita sul contenuto dei file aventi come nome *prova* seguito da un carattere qualsiasi (ES: prova1, prova2, provaA,…).
- -E indica che il comando grep è utilizza espressioni regolari estese
- ^.* indica che le righe da selezionare possono iniziare con una qualsiasi sequenza di caratteri
- \<7[3-9] indica che nella riga deve essere presente una parola che inizi con 7 seguito da un carattere tra 3 e 9 (ES: 73, 74, …79)

Esercizi

6) Spiegare quale è l'effetto del seguente comando:
   - ls -aF1 | grep '\<.`

   Soluzione
   - ls -aF1 visualizza i file presenti nella directory corrente compresi i file nascosti (-a), aggiungendo a ciascun file un carattere che ne indica il tipo (-F) mostrando il tutto in una singola colonna (-1)
   - grep '\<.` seleziona i file che presentino delle parole che inizino con un carattere qualsiasi. Seleziona quindi tutti i file ottenuti con il comando precedente

7) Scrivere una pipe di comandi che consenta di estrarre da un file di testo di nome fileA le prime X linee e di ordinarle in ordine alfabetico decrescente.

   Soluzione
   - head –n X fileA | sort –r

8) Scrivere una sequenza di comandi che consenta di ordinare in ordine alfabetico il contenuto di un file di nome gianni e di estrarre dal file ordinato le prime X linee scrivendole in appendice sul file gianni2 nella home directory dell'utente loggato.

   Soluzione
   - sort gianni | head –n X >> ~/gianni2

9) Il file *text* contiene le seguenti parole:

```
casa
cane
gatto
volpe
```

tail -n +2 →

```
cane
gatto
volpe
```

grep 'a' →

```
cane
gatto
```

Indicare l'output del seguente comando
- tail -n +2 text | grep 'a'| sort –r

Soluzione
- gatto
  cane

10) Dati due file, uno di nome *fileA* costituito da X linee di testo e uno di nome *fileB* costituito da Y linee di testo, indicare l'output del comando
- (cat fileA fileB) | wc –l

Soluzione
- cat concatena i due file mentre wc visualizza il n° di linee complessive, quindi l'output sarà rappresentato dal valore X+Y

11) Dato il file *elenco* così fatto:

```
mario rossi 12/07/1982 25 Ammesso
paolo paoli 15/09/1984 16 Non Ammesso
rocco verdi 12/02/1980 22 Ammesso
marco rossi 24/06/1984 19 Ammesso
sergio bianchi 02/07/1985 28 Ammesso
fabio giallo 03/05/1988 19 Ammesso
rosa barbieri 17/03/1981 20 Ammesso
```

11.a) Dire qual è l'effetto del comando:

- cat elenco | tail -5 | grep '^.*\<[m-z].*$' 2> risultati

Soluzione

- Non essendoci errori durante l'esecuzione del comando non saranno inviati messaggi sullo standard error e quindi il file *risultati* sarà vuoto.
- Lo standard output visualizzerà:

```
rocco verdi 12/02/1980 22 Ammesso
marco rossi 24/06/1984 19 Ammesso
sergio bianchi 02/07/1985 28 Ammesso
rosa barbieri 17/03/1981 20 Ammesso
```

11.b) Dire qual è l'effetto del comando:

- cat elenco | head -6 | grep '\<2[29].*$' -v > output

Soluzione

- Il contenuto del file *output* sarà:

```
mario rossi 12/07/1982 25 Ammesso
paolo paoli 15/09/1984 16 Non Ammesso
marco rossi 24/06/1984 19 Ammesso
sergio bianchi 02/07/1985 28 Ammesso
fabio giallo 03/05/1988 19 Ammesso
```

Esercizi

12) Scrivere un comando per listare il contenuto delle ultime 5 directory nascoste presenti all'interno della home directory dell'utente corrente

Soluzione

- ls -a ~ | grep '^\..*\/$' | tail -5

13) Il file *elenco* ha il seguente contenuto

```
Rossi Mario
Rossi Paolo
Verdi Giuseppe
Bollo Franco
Bianchi Alessandra
Cerri Elena
Dodi francesco
```

Si costruisca il file *risultato.prova* che contenga in ordine alfabetico inverso tutti gli elementi con cognome che inizi per C o per B

Soluzione

- sort -r elenco | grep '^[BC]' > risultato.prova

- Un processo eseguito in **foreground** interagisce in maniera diretta con l'utente (browser, editor di testo).
- Un processo in **background** è invece un processo che non interagisce direttamente con l'utente e tipicamente fornisce un servizio non direttamente visibile all'utente (servizio di sistema, demone).
- Per eseguire un comando (o una pipeline) in background da shell si aggiunge il carattere e-commerciale **&** alla fine della linea comando.
- Quando viene lanciata una pipeline in background vengono mostrati il **numero di job** in background sulla macchina e il **PID** dell'ultimo processo della pipeline.

```
sleep 5&
        [1] 15902
ps
    PID TTY          TIME CMD
   6690 pts/0    00:00:00 bash
  15902 pts/0    00:00:00 sleep
  15904 pts/0    00:00:00 ps
```

yes – fg - bg

- **yes**: Stampa il carattere "y", o la stringa specificata indefinitamente sullo standard finché non viene terminato.

  SINTASSI

  yes [stringa]

  La sequenza CTRL-Z ferma l'esecuzione del comando o di tutti i processi della pipeline restituendo il controllo alla shell, il job in questione viene messo in background. Premendo invece CTRL-C viene terminato il comando o la pipeline attualmente in esecuzione in foreground.

- **fg**: Porta in foreground un job che prima era in background. Se non viene specificato il job su cui agire si intende quello attuale, ossia l'ultimo referenziato.

  SINTASSI

  fg [num_job]

- **bg**: Permette di far riprendere in background l'esecuzione di un job sospeso, se il job in questione non è in attesa di un input o di poter emettere l'output.

  SINTASSI

  bg [num_job]

sleep – wait - sh

- **sleep**: Sospende la shell per i secondi specificati.

  SINTASSI

  sleep num_secondi

- **wait**: Sospende l'esecuzione del processo padre di quello passato in argomento, finché quello in argomento non termina. Senza argomento, sospende il padre finché non terminano tutti i figli.

  SINTASSI

  wait [PID]

- **sh**: L'interprete standard di comandi, permette quindi di eseguire comandi letti da una stringa a riga di comando (-c), dallo standard input (-s) o da un file specificato.

  SINTASSI

  sh -c stringa_comando

  sh -s

  sh file_script

jobs

- **jobs**: Mostra lo stato dei job avviati nella sessione corrente. In maniera informale un jobs è definibile come un'attività derivata da un comando dato attraverso una shell.

SINTASSI

jobs [-lp][num_job]

```
user@ubuntu:~$ sleep 15&
[1] 17245
user@ubuntu:~$ jobs
[1]+  Running sleep 15 &
```

FLAG

- -l Visualizzazione estesa delle informazioni di ogni processo di stato

```
user@ubuntu:~$ jobs -l
[1]+ 17245 Running sleep 15 &
```

- -p Mostra solo il PID del processo leader del job selezionato

```
user@ubuntu:~$ jobs -p
17245
```

ps

- **ps**: (process status) Mostra i processi attivi al momento (snapshot statica)

SINTASSI

ps [-rxeaf] [f] [-u lista_user] [-p lista_PID] [-t lista_tty]

FLAG

- -u [lista_utenti] Solo i processi relativi agli utenti in lista_utenti, se non specificato all'utente corrente.
- -f Visualizzazione estesa
- -r Restringe la selezione solo ai processi in esecuzione
- -p [lista_PID] Solo le informazioni relative ai PID specificati

ps ax: Visualizza i processi di tutti gli utenti e di sistema (con terminale criptato) in BSD syntax

ESEMPIO

user@ubuntu:~$ *ps ax | head -n 10*

PID	TTY	STAT	TIME	COMMAND
1	?	Ss	0:01	/sbin/init
2	?	S	0:00	[kthreadd]
3	?	S	0:00	[ksoftirqd/0]
4	?	S	0:01	[kworker/0:0]
5	?	S<	0:00	[kworker/0:0H]
7	?	S	0:00	[rcu_sched]
8	?	S	0:00	[rcu_bh]
9	?	S	0:00	[migration/0]
10	?	S	0:00	[watchdog/0]

STAT : Codici dello stato dei processi
R : Running
S : Sleeping interrompibile
D : Sleeping non interrompibile
T : Stopped
< : Alta Priorità
…

top

- **top**: Mostra dinamicamente (in foreground), ad intervalli regolari, una serie di informazioni sui processi e sul sistema.

SINTASSI

top

(q per uscire)

ESEMPIO

```
user@ubuntu:~$ top
top - 12:31:19 up  7:12,  2 users,  load average: 1.64, 1.45, 1.36
Tasks: 116 total,   4 running, 112 sleeping,   0 stopped,   0 zombie
Cpu(s):  2.7%us,  1.7%sy,  0.0%ni, 33.2%id, 62.5%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:    514296k total,   493048k used,    21248k free,   136620k buffers
Swap:        0k total,        0k used,        0k free,   153360k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 1983 root      15  -5     0    0    0 D  3.7  0.0  15:26.33 kjournald
 6059 root      20   0 42320  14m 7376 S  3.7  3.0   1:29.96 Xorg
   48 root      15  -5     0    0    0 S  1.3  0.0   5:00.52 kblockd/0
 5938 mythtv    20   0 69804 7844 5044 D  1.3  1.5   5:13.02 mythbackend
 6541 user      20   0 45252  14m 8428 S  0.7  2.9   0:01.26 gnome-settings-
 6633 user      20   0 30156  15m  10m S  0.7  3.0   0:39.56 vmware-user-loa
16217 user      20   0 98.8m  25m  12m R  0.7  5.0   0:07.76 gnome-terminal
 4974 root      20   0 13808 2820 2288 R  0.3  0.5   0:28.42 vmtoolsd
 5492 mysql     20   0  125m  17m 5188 S  0.3  3.5   1:00.04 mysqld
    1 root      20   0  3056 1888  564 S  0.0  0.4   0:02.00 init
    2 root      15  -5     0    0    0 S  0.0  0.0   0:00.00 kthreadd
    3 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 migration/0
    4 root      15  -5     0    0    0 S  0.0  0.0   0:03.76 ksoftirqd/0
    5 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 watchdog/0
    6 root      15  -5     0    0    0 R  0.0  0.0   0:00.62 events/0
```

PARTE SUPERIORE
Informazioni generali sul sistema

TABELLA CENTRALE
Mostra i processi che usano maggiormente la CPU

kill

- **kill**: permette di inviare un segnale a ciascun processo di cui è indicato il PID come argomento. Qualora non specificato il segnale di default è SIGTERM (15).
- segnale: interrupt software → comunicazione asincrona tra processi

SINTASSI

kill [-s nome_segnale] [-nome_segn] [-num_segn] num_PID

1	**SIGHUP**	Il collegamento con il terminale è stato interrotto.
2	**SIGINT**	Interruzione attraverso un comando dalla tastiera (CTRL+C).
3	**SIGQUIT**	Conclusione attraverso un comando dalla tastiera (CTRL+Y) con core dump → creazione di un'immagine di memoria del processo in esecuzione al momento della ricezione del segnale.
9	**SIGKILL**	Terminazione forzata (non può essere nè intercettato, nè ignorato).
15	**SIGTERM**	Terminazione del software, inviato di default da kill e killall.
30	**SIGPWR**	Mancanza di corrente.

killall

- **killall**: Si comporta esattamente come il comando kill con l'unica differenza che non richiede come argomento uno o più PID, ma il **nome** dei processi. Invia il segnale a tutti i processi che eseguono il comando specificato. Se non è specificato nessun segnale, di default esso è SIGTERM (15).

- SINTASSI

killall [-ui] [-nome_segn] [-num_segn] nome_proc

- FLAG
  - -u[utente] Manda il segnale di terminazione a tutti i processi relativi all'utente specificato.
  - -i Chiede conferma prima di inviare il segnale, processo per processo.

- ESEMPIO
  - sleep 160 &
  - sleep 120 &
  - yes > /dev/null &
  - killall –i sleep

Richiede interattivamente conferma per la terminazione (SIGTERM) dei comandi sleep (in background)