L'interfaccia testo

Login

Riga di comando

Shell: fondamenti

L'interfaccia testo

Lo strumento più potente,flessibile e soprattutto più standard con cui amministrare un sistema Unix è l'interfaccia testuale a riga di comando.

Componente essenziale è la *shell* o interprete dei comandi, che permette di svolgere compiti quali job control, mette a disposizione variabili e strutture di controllo per scrivere semplici programmi, e permette di invocare gli eseguibili installati nel sistema.

Una delle shell più usate è la *bash*, un'evoluzione della classica Bourne Shell di Unix.

L'utente 'root'

La maggior parte delle operazioni normalmente eseguite sulla macchina non richiede particolari privilegi. Un utente standard puo' tranquillamente utilizzare la macchina per programmare, editare testi, collegarsi ad internet…

Le operazioni di manutenzione, di installazione e di configurazione dei pacchetti devono invece essere eseguite da un utente particolare (**root**) che possieda opportuni diritti di scrittura e modifica delle directory e dei file 'critici'.

E' sempre buona norma, anche per gli utenti che conoscono la password di root, utilizzare un account standard per le operazioni di routine.

Shutdown

Anche lo spegnimento di una macchina Linux (e unix in generale) richiede alcune operazioni.

Si può abbandonare semplicemente la sessione di lavoro con exit (dalla shell di login) o con logout. Il sistema rimane comunue attivo e pronto ad altri accessi.

Lo spegnimento richiede invece, di norma, l'accesso di root. Alcune installazioni permettono a chiunque si trovi in possesso della console (tastiera) di eseguire lo spegnimento (shutdown) premendo ALT-CTRL-CANC.

Shutdown

root puo' invocare direttamente lo spegnimento, tipicamente con il comando:

shutdown [-h|-r] now

-h indica la richiesta di arrestare il sistema (esiste anche il comando **halt**),
-r indica la richiesta di riavviare il sistema (esiste anche il comando **reboot**).

now indica quando eseguire l'operazione. E' possibile, ad esempio, lasciare agli utenti collegati alcuni minuti per terminare il proprio lavoro.

Chi sono, chi c'e'

Poiché è del tutto comune disporre di differenti identificativi utente con cui lavorare, è utile disporre di un comando per sapere quale e' l'identificativo con il quale si sta operando:

whoami	indica il proprio username
id	dà informazioni sull'identità e sul gruppo di appartenenza
who	indica chi e' attualmente collegato alla macchina

Cambiare identità

L'utilità di questi comandi diviene ancora più evidente se si considera che un utente puo' modificare durante la sessione di lavoro la propria “identità” con il comando 'su':

su tipicamente (senza argomenti) consente l'accesso come root ma può anche consentire l'accesso come altri utenti:

su alex

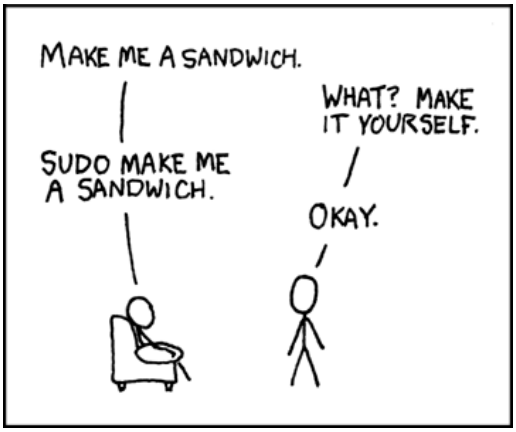
naturalmente l’operazione richiede di conoscere la password dell’utente nel quale ci si vuole “trasformare”, a meno che a lanciare il comando non sia root.

L'utente 'root'

su non è il modo ideale di passare all'identità di root quando necessario

- chiede la password di root
- consente accesso illimitato

→ inadatto alla delega di task a più “vice-amministratori”



sudo è la soluzione migliore:

- usa la password dell'utente di origine (solo per evitare attacchi a terminali incustoditi)
- autorizzazioni configurabili
  - `/etc/sudoers`

La command line

La shell indica il proprio stato di 'pronto' con una stringa di caratteri visualizzati nella parte iniziale della prima linea vuota. Questa stringa e' detta 'prompt'.

I caratteri digitati dall'utente dopo il prompt e terminati dal fine linea (ritorno a capo) costituiscono la command line.

Questa, tipicamente, contiene comandi e argomenti.
I comandi digitati sulla command line possono essere:
- built-in
- comandi esterni

La command line

Un built-in e' un comando direttamente eseguito dal codice interno della shell che lo interpreta e lo 'converte' in azioni sul sistema operativo. Un esempio tipico è costituito dai comandi per la navigazione del filesystem.

Un comando esterno e' un file eseguibile che viene localizzato e messo in esecuzione dalla shell (tipicamente in un processo-figlio). La shell può attenderne o meno la conclusione prima di accettare nuovi comandi. Il risultato di un comando esterno è un codice di ritorno ed un file (standard output).

"Scorciatoie" utili della shell bash - autocompletamento

Iniziando a scrivere un comando o un nome di file come argomento, e battendo TAB, bash completa automaticamente la stringa se non ci sono ambiguità, o suggerisce come completarla correttamente.

Es.: al prompt digito **`pass`**<TAB> → compare **`passwd`** seguito da spazio ad indicare che passwd è l'unico completamento possibile

Se ci sono ambiguità, una seconda pressione di <TAB> mostra i completamenti possibili.

Es. digito **`ls /etc/pam`**<TAB> → compare **`/etc/pam.`**
digito <TAB> → vengono elencati **`pam.conf`** e **`pam.d/`**
aggiungo "**`c`**" e digito <TAB> → compare **`/etc/pam.conf`**

"Scorciatoie" utili della shell bash - history

`history` mostra l'elenco di tutti i comandi eseguiti in un terminale. Per richiamarli sulla command line, basta usare la freccia-su, appariranno (editabili) dal più recente al più vecchio

La history è anche ricercabile interattivamente: al prompt basta digitare CTRL-r per far apparire un prompt (reverse-i-search); digitando una stringa, verrà mostrato il comando più recente che la contiene.
Per navigare verso comandi impartiti precedentemente basta digitare nuovamente CTRL-r.
Individuato il comando desiderato, si può lanciare direttamente premendo invio, o renderlo editabile sulla command line con freccia-destra o freccia-sinistra

Documentare le proprie attività

Il comando **`script`** permette di catturare in un file una sessione di lavoro al terminale, esattamente come compare a video, quindi sia i comandi impartiti che il loro risultato.

Per terminare l'attività, digitare **`exit`** o premere CTRL-d

Gli argomenti

Ogni comando, sia built-in che esterno, può accedere ai caratteri che seguono la propria invocazione sulla command line. I gruppi di caratteri, separati da spazi, rappresentano gli argomenti, cioè i dati su cui si vuole che il comando operi.

Un argomento che inizia con il carattere '-' e' indicato come opzione, e normalmente non è un vero e proprio dato da elaborare ma un modo di chiedere al comando di comportarsi secondo una certa modalità. Più opzioni possono essere solitamente raggruppate in un’unica stringa.

Gli argomenti

Esempi:

ls /home	arg #1="/home" indica la dir di partenza per la lista
ls -l /home	arg #1=opzione **l** indica che si desidera un listato in forma lunga
ls -l -a /home/alex	arg #1 e #2 = opzioni l (lunga) ed a (all)
ls -la /home/alex	arg #1 = opzioni concatenate l+a (identico a prima)

Documentazione

Le fonti di documentazione per Linux sono molteplici.

- man pages – ogni applicazione installa “pagine di manuale” relative al suo utilizzo e configurazione. Le man pages si leggono con il comando *man*
- info files – a metà strada tra la man page e l’ipertesto, si leggono con il comando *info*, che invoca l’editor emacs appositamente esteso per gestire questi file
- HOWTO – documenti specifici per la risoluzione dei più svariati problemi pratici, sono raccolti in un pacchetto e vengono tutti installati in */usr/[share/]doc/HOWTO* Inoltre, sotto */usr/doc/HOWTO/translations/it* si possono trovare la maggior parte degli HOWTO tradotti in italiano.
- on-line – troppe fonti per citarle... un punto di partenza può essere http://tldp.org/ The Linux Documentation Project

Man pages

Una installazione standard di unix mette a disposizione innumerevoli pagine di manuale raggruppate in sezioni:

(1) User commands
(2) Chiamate al sistema operativo
(3) Funzioni di libreria, ed in particolare
(4) File speciali (/dev/*)
(5) Formati dei file, dei protocolli, e delle relative strutture C
(6) Giochi
(7) Varie: macro, header, filesystem, concetti generali
(8) Comandi di amministrazione riservati a *root*
(n) Comandi predefiniti del linguaggio Tcl/Tk

Man pages

L'accesso alle pagine si ottiene con il comando
man <nome della pagina>

Spesso il nome della pagina coincide con il comando o il nome del file di configurazione che essa documenta.

Alcune opzioni utili sono qui riassunte:

man –a <comando>	cercherà in tutte le sezioni
man <sez.> <comando>	cercherà nella sezione specificata
man –k <keyword>	cercherà tutte le pagine attinenti alla parola chiave specificata

***Per avere altre informazioni sul comando* man *è ovviamente sufficiente usare* man man.**

bash help

I builtin, non essendo programmi installati indipendentemente, non hanno man page.

Un sommario del loro funzionamento può essere visualizzato con
`help <builtin>`

Inoltre, naturalmente, sono documentati nella man page
`bash(1)`

Elencare i file

ls elenca i file o il contenuto della directory specificati come argomento; senza argomenti elenca il contenuto della directory corrente. Le opzioni più comuni sono:

-l abbina al nome le informazioni associate al file
-a non nasconde i nomi dei file che iniziano con .
-A come -a ma esclude i file particolari '.' e '..'
-F postpone il carattere '*' agli eseguibili e '/' ai direttori
-d lista il nome delle directory senza listarne il contenuto
-R percorre ricorsivamente la gerarchia
-i indica gli i-number dei file oltre al loro nome
-r inverte l'ordine dell'elenco
-t lista i file in ordine di data/ora di modifica (dal più recente)

Navigazione nel filesystem

Ogni processo, shell compresa, possiede un'informazione di stato che indica il nome della directory corrente nel file system. Normalmente, all'atto del login, ogni utente si trova nella propria home directory.

In ogni istante si può conoscere la directory corrente con il comando **pwd** (print working directory); è possibile modificare la directory corrente spostandosi cosi' nel file system utilizzando il comando built-in **cd**.

- cd invocato senza argomenti posiziona l'utente nella propria home
- cd seguito da un argomento posiziona l'utente nella directory con tale nome.

Navigazione nel filesystem

Si ricordi che ogni directory di unix contiene due directory speciali:

. rappresenta la directory stessa
.. rappresenta la directory superiore (tranne nella radice */*)

e che ogni percorso che inizia con la barra viene considerato assoluto, cioè relativo alla radice, mentre se inizia con qualsiasi altro carattere viene considerato relativo alla directory corrente.

Es. di spostamento assoluto
cd /home/alex
Es. di spostamento relativo
cd ../pippo/pluto

Altri comandi di uso generale sul filesystem

df	visualizza lo spazio utilizzato e disponibile su ogni filesystem
du	visualizza l'uso di spazio di un file o una directory
rm	cancella un file o, meglio, rimuove il link
cp	copia un file o piu' file in una directory
mv	sposta un file o piu' file in una directory
ln	crea un link ad un file
mkdir	crea una directory
rmdir	cancella una directory
chown	cambia il proprietario di un file
chgrp	cambia il gruppo proprietario di un file

Shell expansion (versione semplice)

La bash permette di specificare nomi di file per mezzo di *pattern* (schemi). Quando si scrive una linea di comando contenente uno schema, la bash la interpreta in 2 passi:

1. Sostituisce lo schema con l'elenco di tutti i nomi di file che si adattano a tale schema
2. Esegue la riga di comando risultante

Shell (pathname) expansion

Gli schemi vengono composti usando caratteri speciali:

***** rappresenta una qualunque stringa di zero o più caratteri

? rappresenta un qualunque carattere singolo

[$c_1c_2c_3$] rappresenta un qualunque carattere purchè appartenente all'insieme. Anche **range** di valori: **[c_0-c_n]**
Es. ***ls [q-s]**** lista i file con nomi che iniziano con almeno un carattere compreso tra q e s

[!$c_1c_2c_3$] rappresenta un qualunque carattere purchè non appartenente all'insieme
Es. ***ls *[!0-9]*** lista i nomi dei file che terminano con caratteri non numerici

Esempi d'uso delle wildcard

ls * lista i file del direttorio corrente (nel caso vi siano direttori cosa succede?)

ls [a-p,1-7]*[cfd]?
lista i file i cui nomi hanno come iniziale un carattere compreso tra 'a e 'p' e tra 1 e 7
Il penultimo carattere deve essere c, f, oppure d.

echo * esegue l'echo del carattere '*', privato del suo significato

ls *[!*\?]* lista tutti i file del direttorio corrente che non contengono una wildcard * o ?

Esempi d'uso delle wildcard

ls /*/*/* lista tutti i file dei direttori di secondo livello a partire dalla root

ls -d /*/*/* lista tutti i file dei direttori di secondo livello a partire dalla root (i direttori sono trattati come file)

ls [a-z]*[0-9]*[A-Z] lista i file i cui nomi iniziano con una minuscola, terminano con una maiuscola e contengono almeno un carattere numerico

ls *[!0-9]* lista i file del direttorio corrente i cui nomi non contengono alcun carattere numerico

Esempi d'uso delle wildcard

ls [a-z,A-Z,0-9][a-zA-Z0-9] lista i file con nomi di due caratteri alfanumerici (entrando nei direttori)

echo [a-z,A-Z,0-9][a-zA-Z0-9] lista i file con nomi di due caratteri alfanumerici

Shell expansion

(tutta la verità)

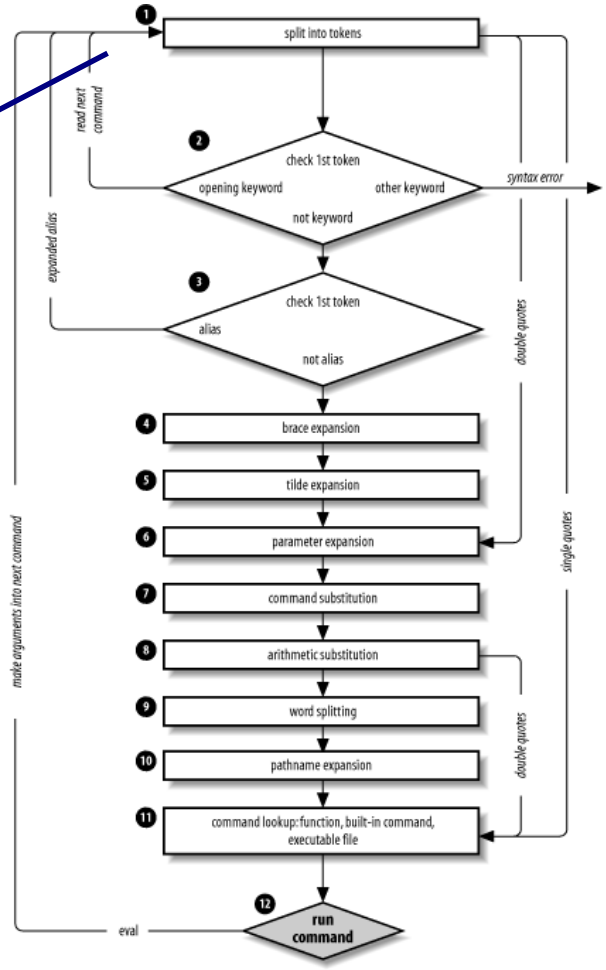
L'espansione dei caratteri speciali riguardanti i nomi di file è l'ultima di molte che la shell compie prima di lanciare il comando scritto sulla riga.

Esemplifichiamo i passi, che saranno chiari via via che vengono introdotti concetti più avanzati.

Processing della riga di comando

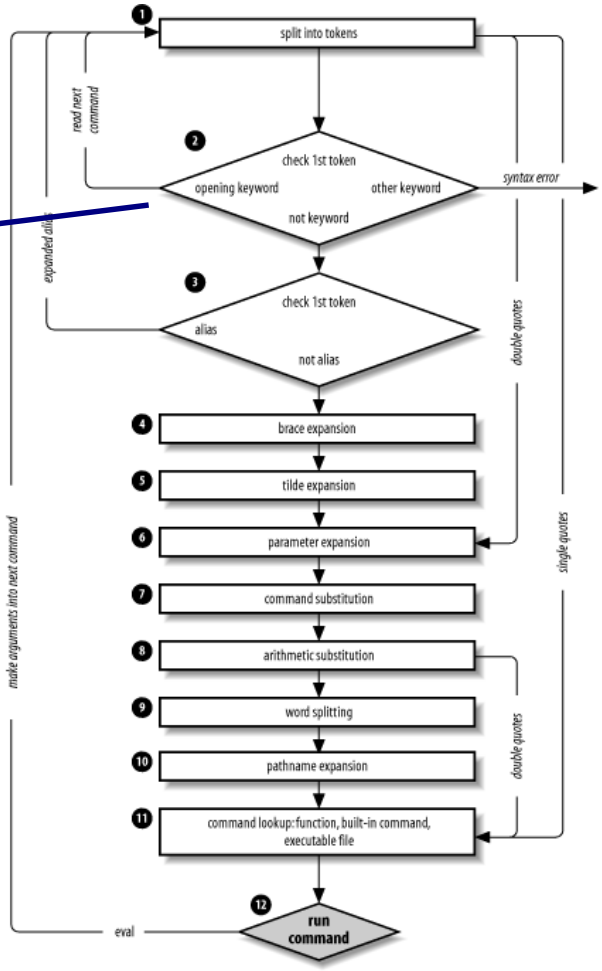
1. Individua i *token* usando come separatori un elenco fisso di metacaratteri: **`SPACE TAB NEWLINE ; ( ) < > | &`**

I token sono stringhe, parole chiave, ridirettori, ":"



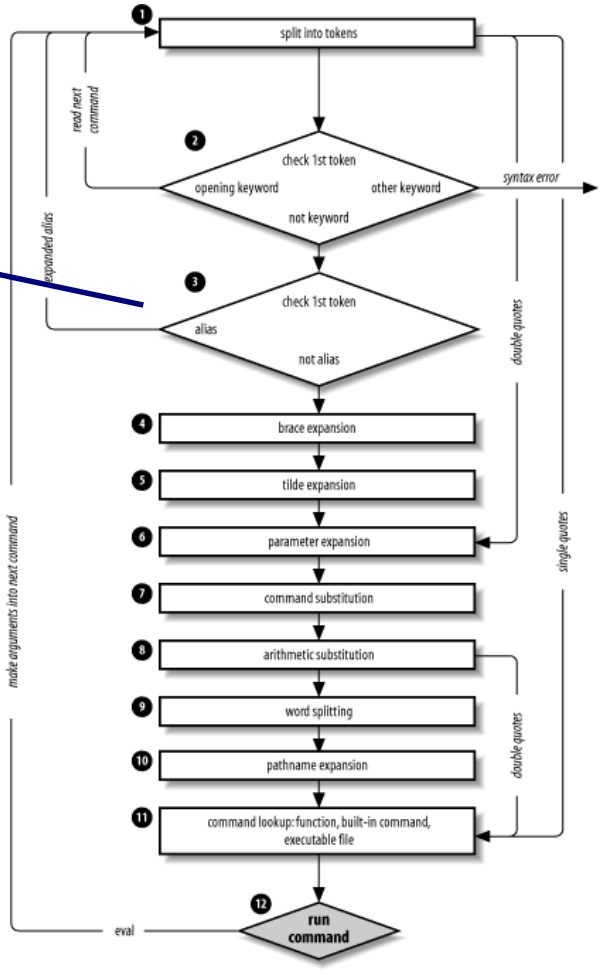
Processing della riga di comando

2. se il primo token individua l'inizio di un comando composto (es. **`if, while, function, { , (`**), la shell predispone l'ambiente per il comando composto e ne va a leggere il primo token



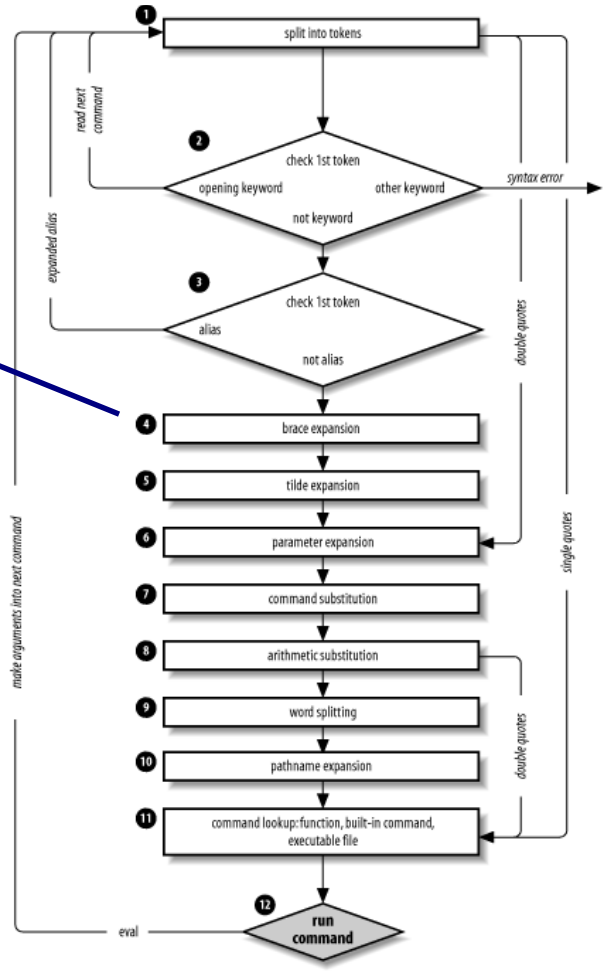
Processing della riga di comando

3. Ricerca la prima parola di ogni comando nella lista degli alias. Se trova un alias, lo espande e riparte col processing dal punto 1.
(Si noti che questo consente alias ricorsivi e definizione di alias anche per parole chiave)



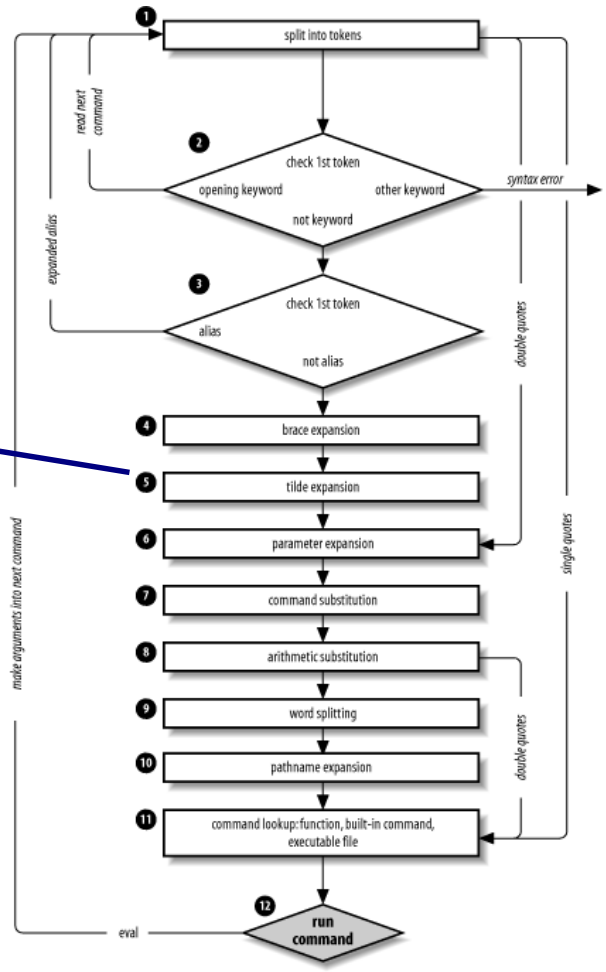
Processing della riga di comando

4. Brace expansion: es.
`a{b,c} → ab ac`



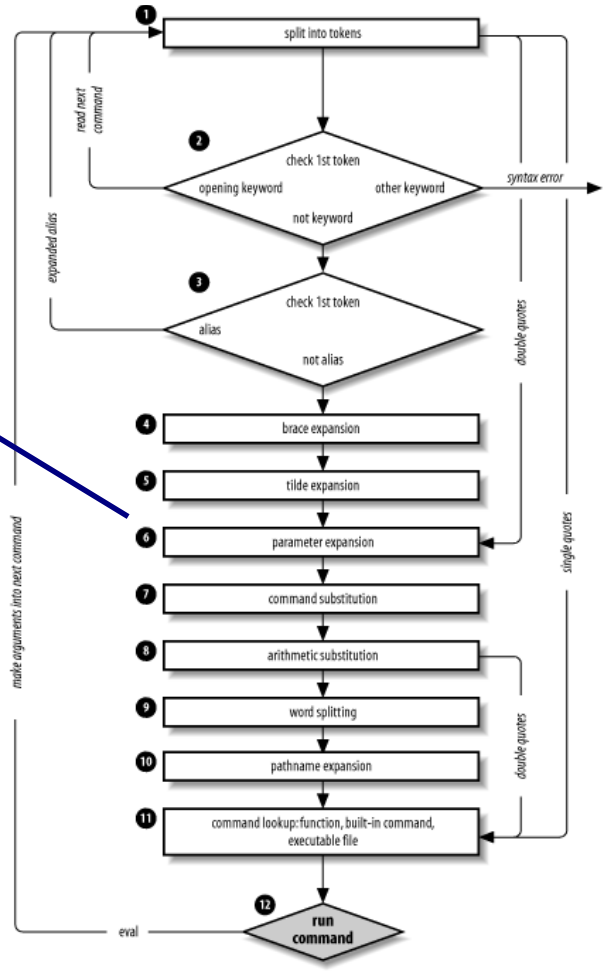
Processing della riga di comando

5. Tilde expansion
Se c'è un token nella forma **`~username`**, viene sostituito con la home directory dell'utente username (se username è vuoto, si utilizza l'utente corrente)



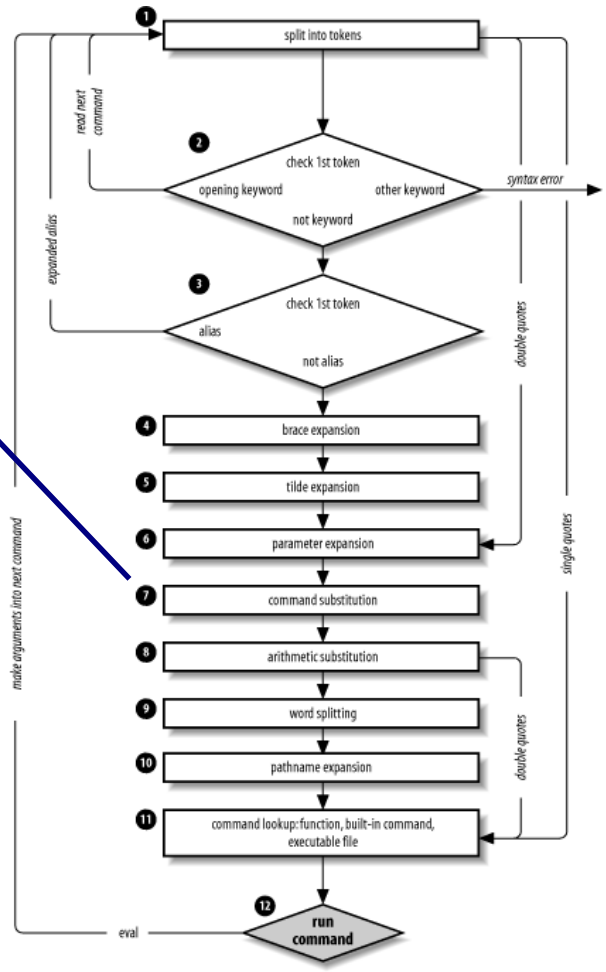
Processing della riga di comando

6. Sostituzione delle variabili (token che iniziano con $) col loro valore



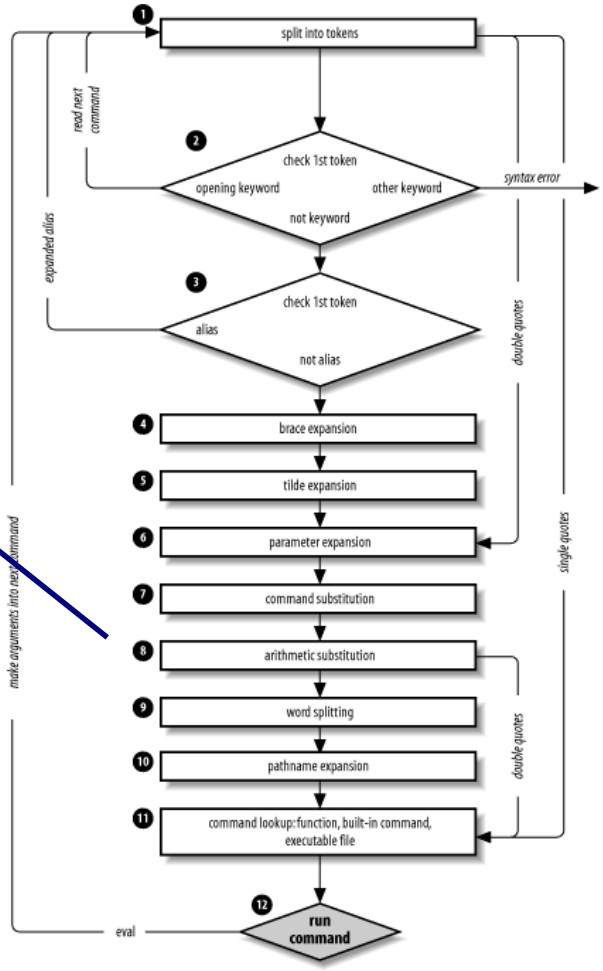
Processing della riga di comando

7. Sostituzione dei comandi racchiusi tra **$(** e **)** con la stringa risultante dalla serializzazione del loro output



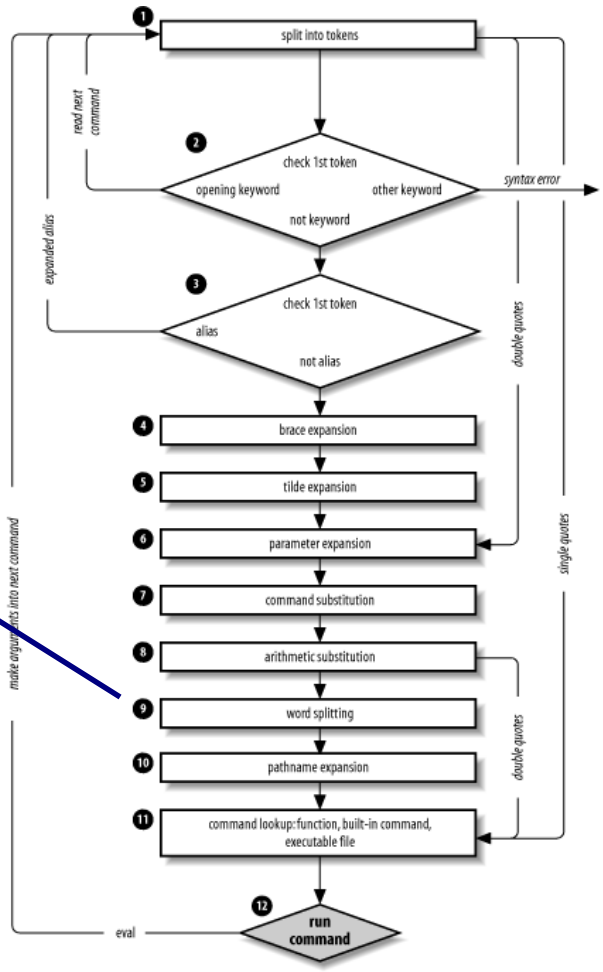
Processing della riga di comando

8. Valutazione delle espressioni aritmetiche nella forma **`$((expr))`**



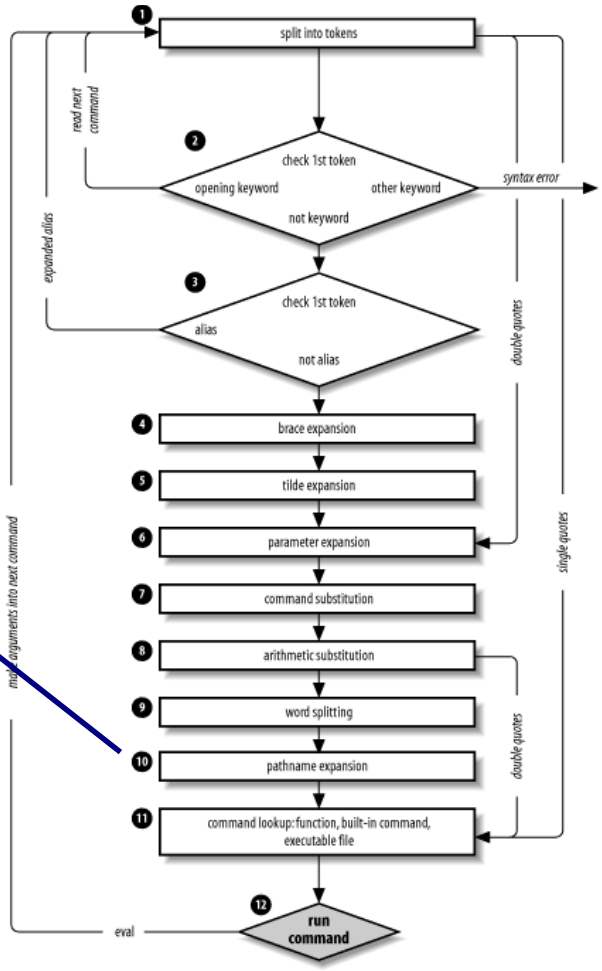
Processing della riga di comando

9. La riga di comando risultante dalle sostituzioni di variabili, comandi ed espressioni aritmetiche è ulteriormente separata in token, stavolta utilizzando $IFS come separatore



Processing della riga di comando

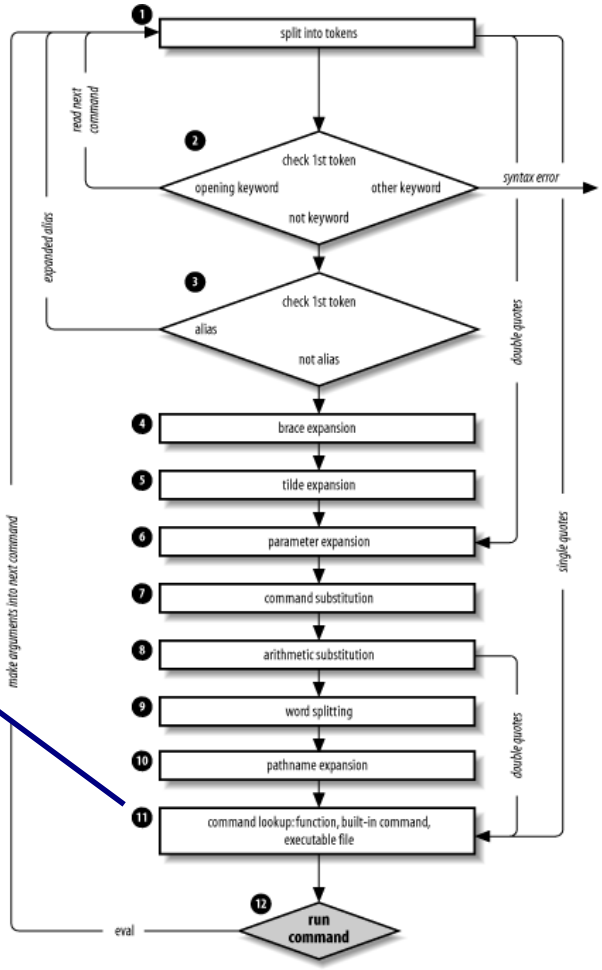
10. Espansione del pathname (o wildcard expansion): i caratteri
 *** ? [ ]**
danno origine agli elenchi di file che corrispondono al pattern



Processing della riga di comando

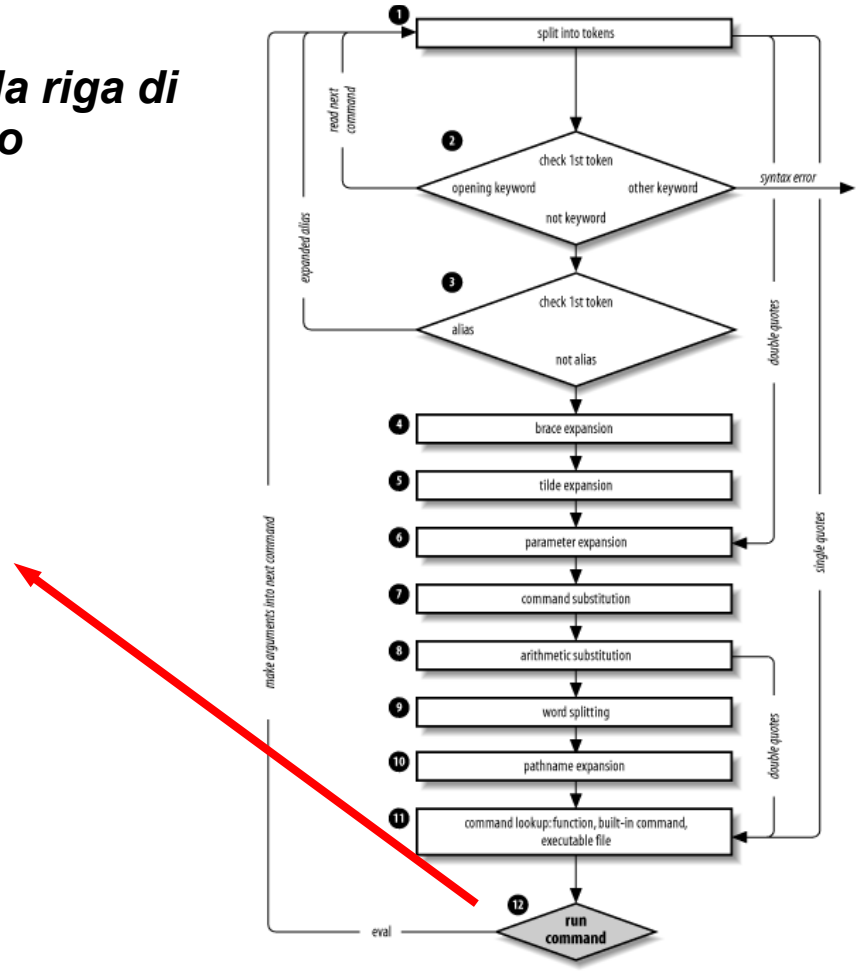
11. La prima parola della riga viene utilizzata come comando, cercando nell'ordine
- una funzione
- un built-in
- un comando in $PATH

che abbia tale nome



Processing della riga di comando

Whew!



Variabili

Le variabili sono un modo offerto dalla shell per memorizzare delle stringhe di testo sotto un dato nome. La modifica o la creazione di una variabile si ottengono semplicemente indicando sulla command line il nome della variabile seguito da **=** e dal valore che le si vuole attribuire.
Nota: non inserire spazi prima e dopo il simbolo =

pippo=valore assegna "valore" alla variabile pippo

Il meccanismo di shell expansion si applica anche agli elementi della riga di comando che iniziano con **$**: $pippo viene sostituito con il valore assegnato alla variabile pippo

Comando echo

Per visualizzare un messaggio, unix mette a diposizione il comando echo che stampa i caratteri che lo seguono. Questo è immediatamente utilizzabile per visualizzare il valore di una variabile:

echo PATH visualizza "PATH"
echo $PATH visualizza il contenuto della variabile PATH
echo $pippo visualizza "valore"

PATH=.:$PATH modifica la var PATH inserendo la dir corrente

Variabili di ambiente

Vi sono alcuni dati, solitamente riguardanti il sistema o le preferenze di un utente, che sono utili a tutti i comandi (es. la versione del sistema operativo, la lingua dell'utente, ecc…). Sarebbe faticoso e ripetitivo doverli passare come argomento ad ogni comando che si esegue: per questo unix dispone del meccanismo delle variabili di ambiente.
La shell distingue le variabili d'ambiente dalle variabili standard (che invece rimangono confinate alla shell stessa) per mezzo dell'*esportazione*.

export pippo

Variabili di ambiente

La variabile esportata è *copiata* ai comandi discendenti dalla shell, non *condivisa*: non è quindi possibile per un processo discendente modificare le variabili di ambiente del padre.

Le operazioni di export e di assegnamento possono essere contemporanee:
export pluto=valore

esempio:

man ls=> output in inglese
export LANG=it
man ls => output in italiano

Quoting

Il meccanismo di espansione di wildcard e variabili è potente ma interferisce con l'interpretazione letterale di alcuni simboli: i già visti **[ ] ! * ? $** ed anche **{ } ( ) " ' ` \ | > < ;**

Quando si debbano passare come parametro ad un comando delle stringhe contenenti tali simboli, è necessario proteggerli dall'espansione.

**** non interpretare il carattere successivo come speciale

Es. ***ls ***** lista i nomi dei file che contengono il carattere * in qualunque posizione

Quoting

' (virgolette singole) ogni carattere di una stringa racchiusa tra virgolette singole viene protetto dall'espansione e trattato letteralmente, senza eccezioni.

" (virgolette doppie) ogni carattere di una stringa racchiusa tra virgolette doppie viene protetto dall'espansione, con l'eccezione del **$**, del backtick (`), di ****, ed altri casi particolari

Ricerca dei comandi

Tra le variabili d'ambiente comuni, la shell utilizza PATH per eseguire la ricerca dei comandi nel file system. La sua struttura è quella di un elenco di directory separate da **:**

PATH=/bin:/usr/bin:/sbin

Si noti che se si vuole mettere in esecuzione un comando (o programma) che si trova nella directory corrente (ad esempio, un compilato) la variabile PATH deve contenere la directory **.** Questa non è una buona norma, perché è facile lanciare per distrazione comandi errati.

Ricerca dei comandi

È comunque sempre possibile lanciare un comando specificandone il percorso esplicito in modo relativo o assoluto, anche se al di fuori di PATH:
/usr/local/bin/top
./mycommand

In un sistema possono essere presenti diverse versioni di uno stesso comando in diverse directory. Il sistema quando non si usa un percorso esplicito lancia quella che trova nella directory che appare per prima in PATH.
which Permette di sapere quale versione si sta usando:
which passwd
/usr/bin/passwd

Alias

La shell mette a disposizione alcuni sistemi per memorizzare linee di comando complesse in comandi più semplici da invocare. Uno di questi è l'**alias** che permette di attribuire un nome ad una command line:

alias miols='ls -l'

Le associazioni definite con alias, così come la definizione di qualunque variabile, vanno perse al termine della sessione.

Alias vs. builtin e comandi

Una volta definiti, gli alias (o come descritto in seguito, le funzioni) questi hanno la priorità rispetto ai builtin ed i comandi omonimi. Tornano utili, per analizzare e pilotare la configurazione, i comandi type, builtin, command e unalias

```
$ alias echo='echo ~~~'
$ echo test
~~~ test
$ \echo test
test
$ builtin echo test
test
$ type echo
echo is aliased to `echo ~~~'
$ unalias echo
$ type -a echo
echo is a shell builtin
echo is /bin/echo
```

\ previene solo l'espansione degli alias, se echo fosse stato rifedinito come funzione sarebbe stata usata quest'ultima

builtin invece fa override sia delle definizioni di alias che di funzioni; vale solo per builtin: se è stato ridefinito un comando esterno, si usi `command` per invocare la versione originale

File di configurazione della shell

Se si desidera ottenere un alias o una variabile persistenti, è possibile inserirne le definizioni in uno dei file di configurazione della shell, che vengono riletti ad ogni esecuzione della shell stessa. Questi file sono molteplici:

1) se la bash è attivata in seguito ad un login:
   prima **/etc/profile**
   poi il primo accessibile tra **~/.bash_profile**, **~/.bash_login**, **~/.profile**
   ed all'uscita **~/.bash_logout**
2) se la bash è invocata in altro modo:
   ~/.bashrc

nota: il carattere **~** è speciale e viene sostituito dalla bash con la homedir dell'utente.