Command line

Richiami dei principi di funzionamento

Marco Prandini
DISI
Università di Bologna

Introduzione

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
- alias
- funzioni

Tipi di comandi

- Una **keyword** è un comando che ha il compito di modificare l'esecuzione di altri comandi (es. misurare il tempo di esecuzione, innescare un ciclo)
- 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.
- Un **alias** è una stringa che viene sostituita da un'altra
- Una **funzione** è un'intera sequenza di comandi shell, con un nome, a cui possono essere passati parametri

(maggiori dettagli su alias e funzioni in seguito)

Documentazione dei comandi (e non solo)

- **man pages** – ogni applicazione installa "pagine di manuale" relative al suo utilizzo e configurazione.
  - Le man pages si leggono con il comando *man*
  - 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)`
- **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

Categorie di 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

Accesso alle 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.

Gli argomenti

- Ogni comando, sia built-in che esterno, può accedere ai caratteri che seguono la propria invocazione sulla command line.
  - La shell inserisce in memoria l'ARGV prima di generare il processo
- 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 '-' è chiamato solitamente *opzione*
  - normalmente non è un vero e proprio dato da elaborare
  - è un modo di specificare una variante al comportamento del comando
  - più opzioni possono essere solitamente raggruppate in un'unica stringa.

Esempi di argomenti e opzioni

ls /home	arg #1="/home" indica la directory di cui elencare il contenuto
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)

Digitare interattivamente la command line

- Iniziando a scrivere un comando (come primo elemento) 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`

Richiamare comandi passati

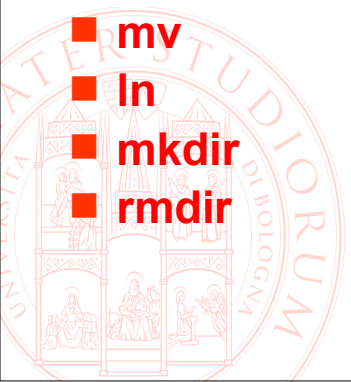
- `history` mostra l'elenco di tutti i comandi eseguiti in un terminale. Per richiamarli sulla command line, basta usare la freccia-su, appariranno (editabili) dal più recente al più vecchio
- La history è anche ricercabile interattivamente: al prompt basta digitare CTRL-r per far apparire un prompt (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 attività svolte sulla shell

- 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

Comandi di base relativi al fliesystem

ls	elenca i file (di default nella directory corrente)
cd	cambia directory (di default porta alla home)
pwd	mostra la directory corrente
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



Qualche opzione di ls

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

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

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

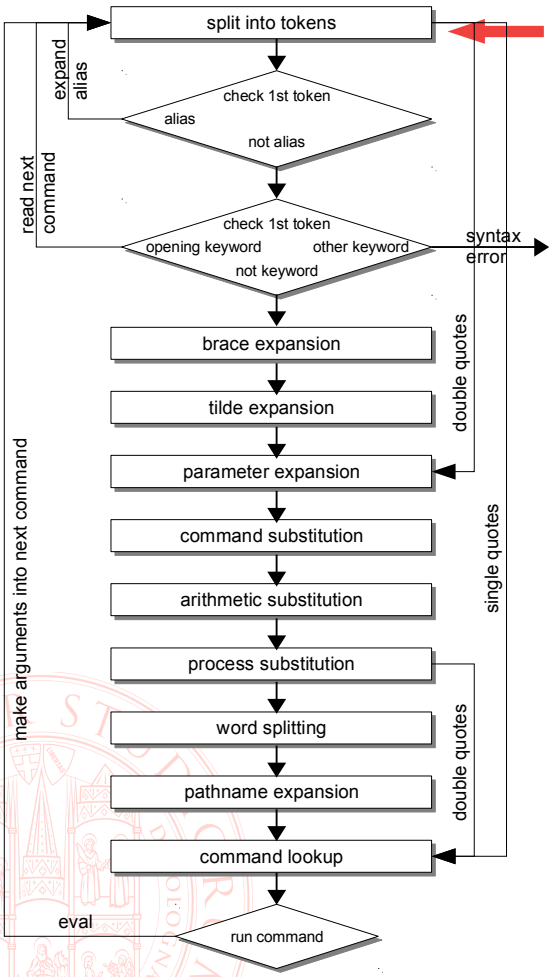
e che ogni percorso che inizia con la barra viene considerato assoluto, cioè relativo alla radice, mentre se inizia con qualsiasi altro carattere viene considerato relativo alla directory corrente.

Es. di spostamento **assoluto**
cd /home/alex
Es. di spostamento **relativo**
cd ../pippo/pluto
"cd -" (trattino/meno) torna alla dir. precedente l'ultimo "cd"

Shell expansion

La shell opera secondo un procedimento di *espansione*

- Individua sequenze speciali contrassegnate da *meta-caratteri*, che non vengono presi a valore nominale
- Interpreta il significato della sequenza speciale
- Al posto della sequenza mette il risultato dell'interpretazione, creando una riga di comando diversa da quella digitata
- Se un'espansione fallisce (ad esempio la sequenza speciale è mal formata, o dipende dalla presenza di dati che a tempo di esecuzione mancano) la sequenza è solitamenet lasciata inalterata sulla riga di comando
- Ci sono ben 12 passi che svolgono manipolazioni diverse della riga di comando, in una sequenza precisa
- Alcuni/tutti possono essere saltati per mezzo del *quoting*, cioè proteggendo i meta-caratteri da non interpretare, per mezzo di altri caratteri speciali: apici `'`, doppi apici `"`, backslash `\`



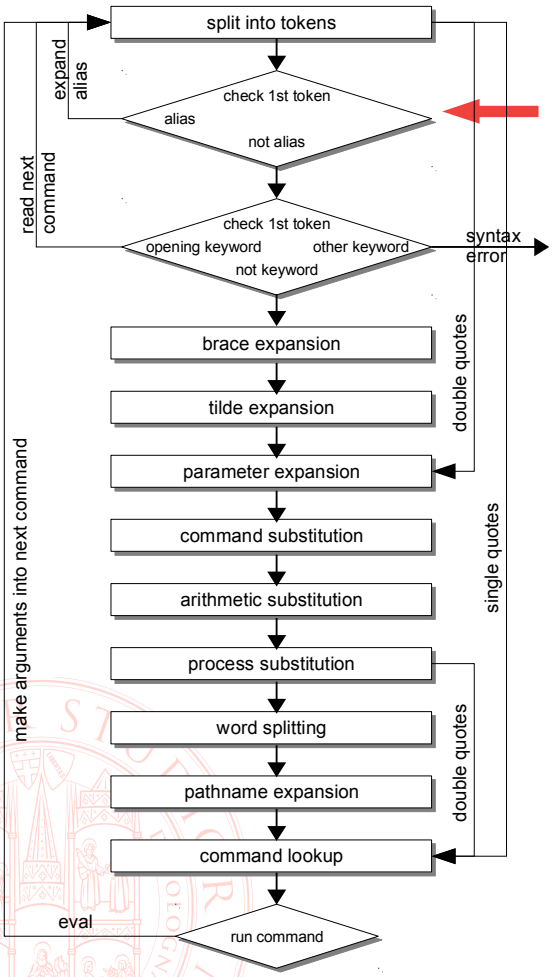
Shell expansion

1. Tokenizzazione

- La riga viene divisa in *token* usando come separatori un elenco fisso di metacaratteri:
  `SPACE TAB NEWLINE`
  `; ( ) < > | &`

- I token possono essere
  - stringhe
  - parole chiave
  - caratteri di ridirezione
  - carattere ":"

- Da qui in poi tutti i passi (2-10) sono saltati per le parti di riga racchiuse tra apici singoli



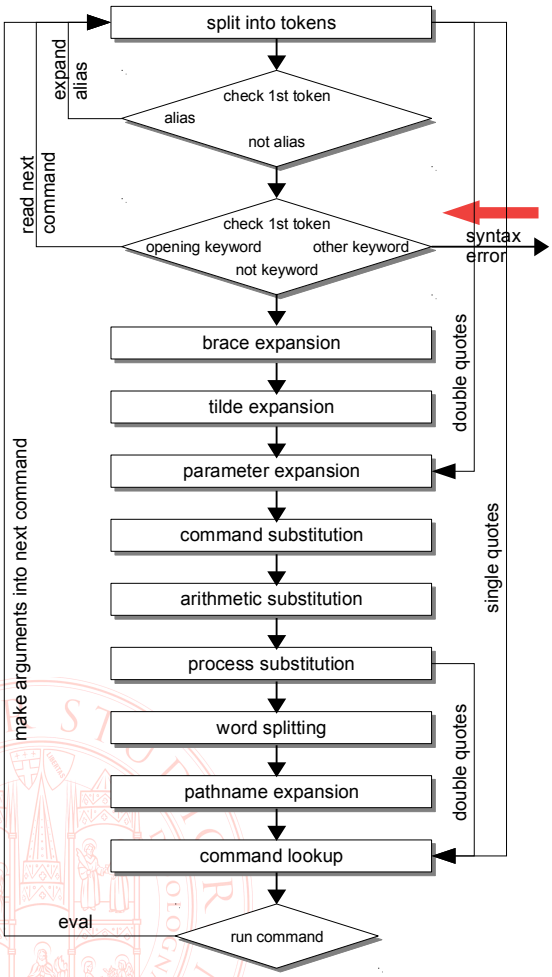
Shell expansion

2. primo token alias

- la shell cerca il primo token nella lista degli alias.
- Se lo trova, lo espande e riparte col processing dal punto 1.
  - Si noti che questo consente alias ricorsivi
  - un alias non verrà mai espanso due volte

    es. alias ls='ls -l' non crea loop

- Non eseguito sulle parti di riga racchiuse tra doppi apici



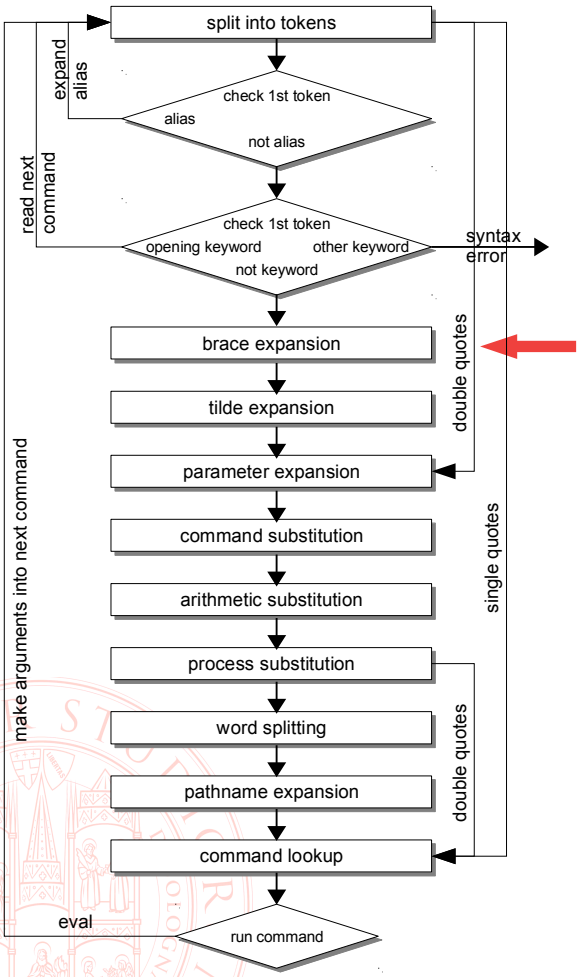
Shell expansion

3. primo token keyword

- se il primo token è una parola chiave che dà inizio a un comando composto, ad es.
  - `if`
  - `while`
  - `function`
  - `{`
  - `(`

  la shell predispone l'ambiente per il comando composto e ne va a leggere il primo token

- Non eseguito sulle parti di riga racchiuse tra doppi apici

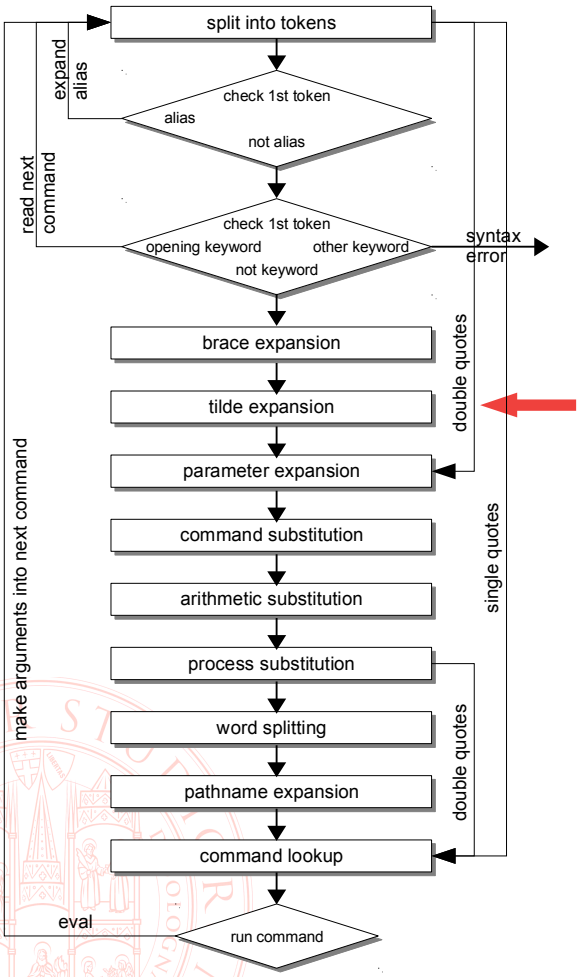


Shell expansion

4. Brace expansion

- `Pre{Lista}Post`
  → `PreItem1Post PreItem2Post`
- lista può essere estensiva
  - `{a,pippo,mamma}`
- o sequenza
  - `{min..max[..incr]}`

- Non eseguito sulle parti di riga racchiuse tra doppi apici

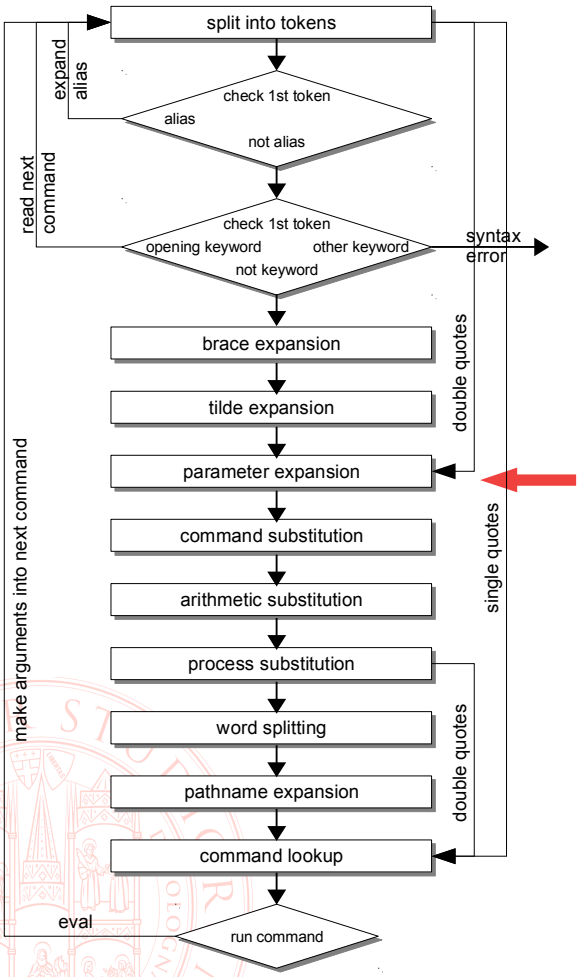


Shell expansion

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)

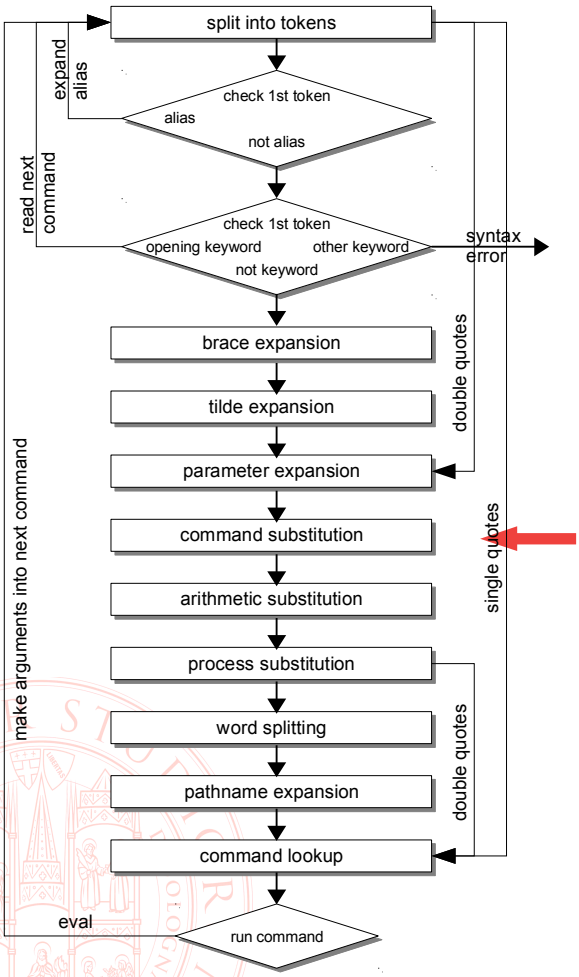
- Non eseguito sulle parti di riga racchiuse tra doppi apici



Shell expansion

6. Parameter expansion

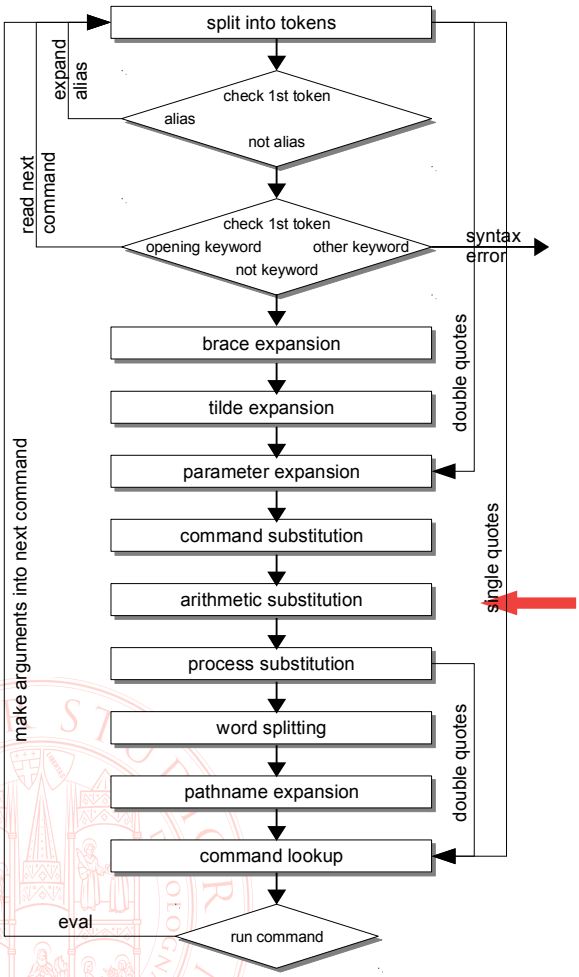
- Il carattere "$" può marcare l'inizio di diverse espansioni
  - parameter expansion
  - command substitution
  - arithmetic expansion
- L'esempio più semplice di PE è la sostituzione della stringa `$NAME` con il valore contenuto nella variabile `NAME`
- Questi quattro passaggi (6..9) sono eseguiti anche sulle parti di riga racchiuse tra doppi apici



Shell expansion

7. command substitution

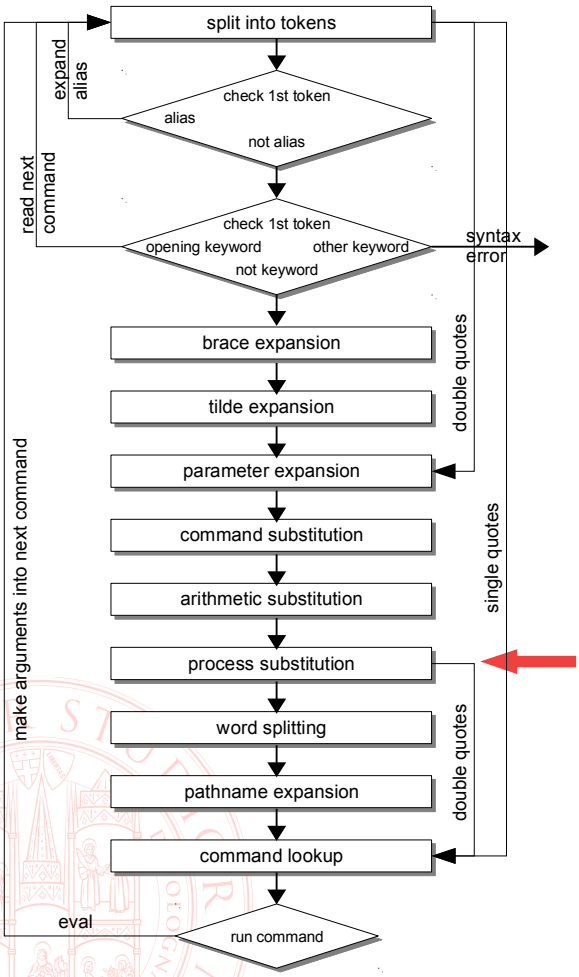
- il token `$(comando)` ha questo effetto:
  - viene creata una subshell
  - vi viene eseguito comando
  - sdtout di `comando` viene posto sulla riga di comando al posto del token originale, a parte eventuali righe vuote alla fine
- Questi quattro passaggi (6..9) sono eseguiti anche sulle parti di riga racchiuse tra doppi apici



Shell expansion

8. arithmetic expansion

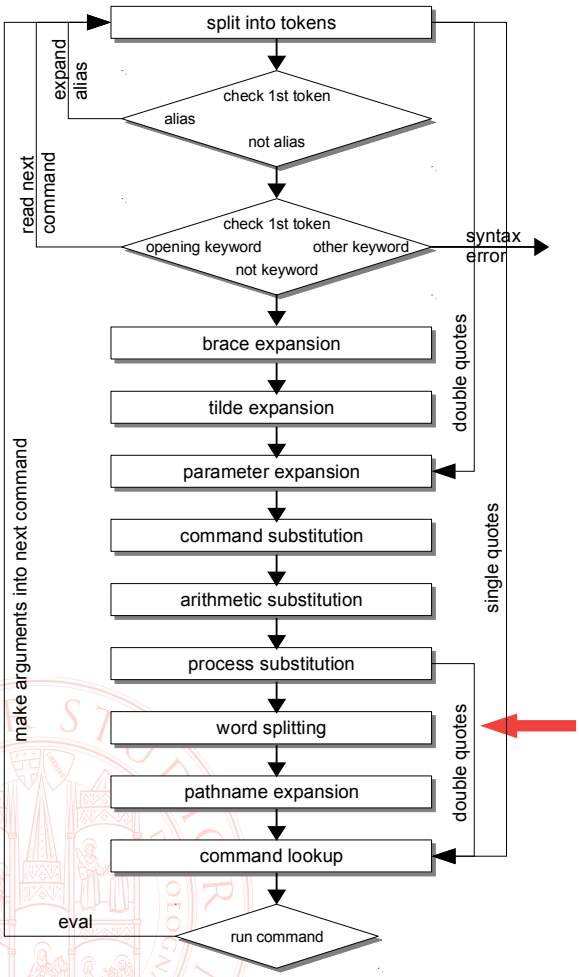
- il token `$((expr))` viene sostituito col risultato della valutazione di `expr`, un'espressione aritmetica
- `expr` viene trattata come se fosse racchiusa tra doppi apici (quindi subisce solo i passi 6 e 7)
- Questi quattro passaggi (6..9) sono eseguiti anche sulle parti di riga racchiuse tra doppi apici



Shell expansion

9. process substitution

- il token `<(comando)` o `>(comando)` ha questo effetto:
  - viene eseguito `comando` in modo concorrente e asincrono rispetto al comando all'inizio della riga
  - il suo input o output appare come un nome di file tra gli argomenti di tale comando
- Questi quattro passaggi (6..9) sono eseguiti anche sulle parti di riga racchiuse tra doppi apici

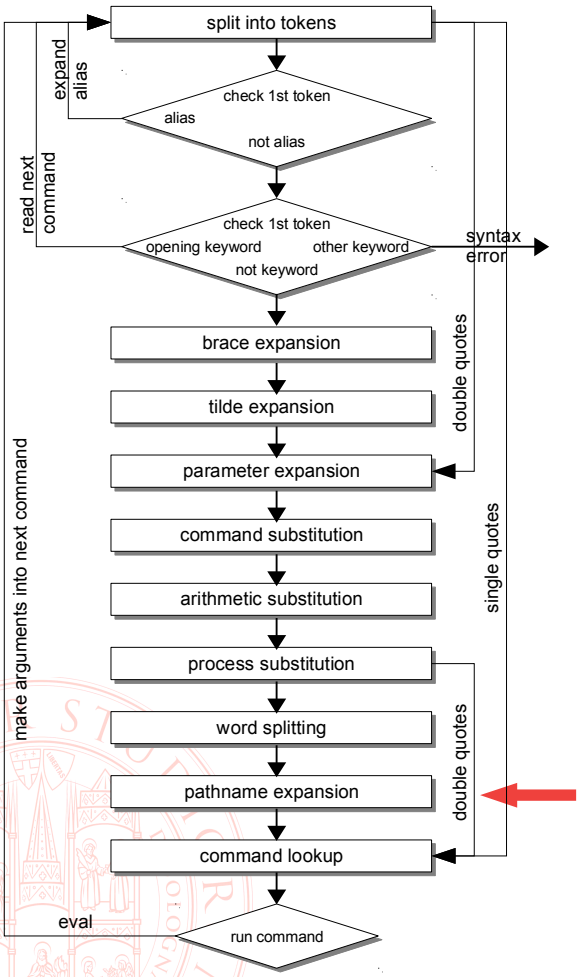


Shell expansion

10. word splitting

- I risultati dei passi 6..9 sono esaminati, e separati in *word* indipendenti
  - separatore = qualsiasi carattere presente nella variabile IFS
  - default IFS = <space><tab><newline>

- Non eseguito sulle parti di riga racchiuse tra doppi apici



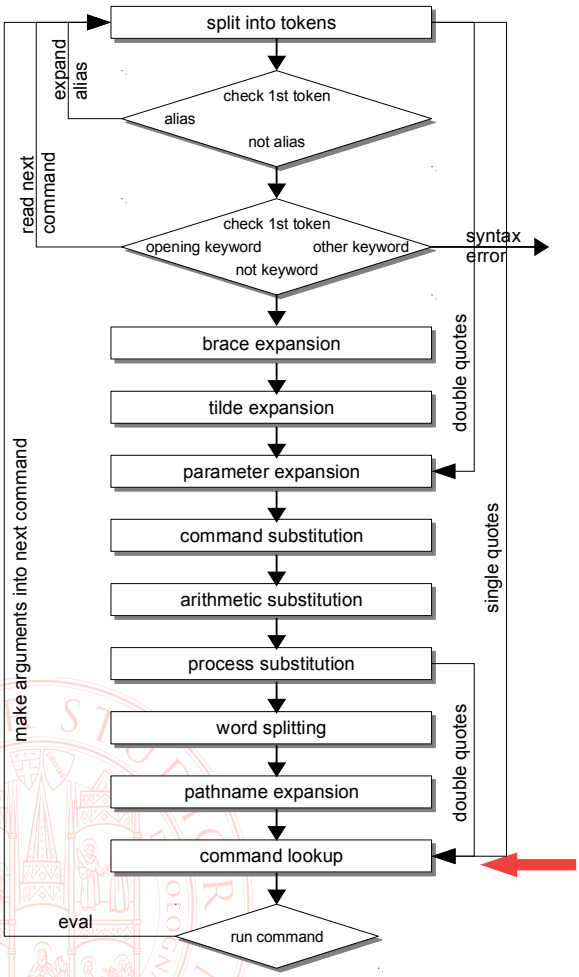
Shell expansion

11. pathname expansion

- Ogni word viene esaminata e se contiene uno dei caratteri
  - *
  - ?
  - [

  viene considerata un *pattern* e sostituita con tutti i nomi di file che concordano

- Non eseguito sulle parti di riga racchiuse tra doppi apici



Shell expansion

12. quote removal ed esecuzione

- Vengono rimosse tutte le occorrenze di caratteri di quoting "usate" effettivamente
  - non protette da altri quoting
  - non generate dai passi 6..9
- Vengono impostati gli stream in caso di ridirezione
- Viene cercato il comando in quest'ordine
  - funzioni
  - builtin
  - eseguibili in $PATH

Ricerca dei comandi tra i diversi tipi

- Come distinguere quale tipo di comando viene eseguito?
  - comando **type**

```
$ type -a echo
echo is a shell builtin
echo is /bin/echo
```

- Come alterare l'ordine di default?
  - backslash **** davanti al comando: previene solo l'espansione degli alias
  - keyword **builtin**: previene l'espansione degli alias e l'uso di funzioni e invoca l'esecuzione del builtin specificato, se esiste
  - keyword **command**: utilizza un comando esterno anche se esiste una funzione con lo stesso nome
  - comando **unalias**: cancella un alias definito in precedenza

Ricerca dei comandi esterni

- 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
```

- più eseguibili omonimi in directory diverse?
  - lista ordinata, il sistema usa la prima istanza che trova
  - **which** Permette di sapere quale versione si sta usando:

```
# which passwd
/usr/bin/passwd
```

- Per consentire automaticamente l'esecuzione di programmi presenti nella directory corrente la variabile PATH deve contenere la directory .
  - Non è una buona norma, è facile lanciare per distrazione comandi errati
  - Meglio usare il percorso esplicito, che permette sempre di specificare quale eseguibile lanciare anche se al di fuori di PATH:

```
/usr/local/bin/top
./mycommand
```


Impostazioni di default

- Per ottenere automaticamente all'avvio della shell
  - assegnazione di valori a variabili come PATH
  - definizione di alias di comandi
  - impostazione di valori di umask
  - … esecuzione di qualsiasi comando shell utile per inizializzare l'ambiente

  si ricorre ai file di configurazione di bash
- Si vedano (quasi all'inizio) la sezione INVOCATION e (quasi in fondo) la sezione FILES di *man bash*
- File globali

```
/etc/profile    /etc/bash.bashrc
```

- File personali (nella home)

```
.bash_profile    .bash_login    .profile    .bashrc
```