Quando una riga di comando viene passata alla *shell*, cerca di capire se sono presenti **coppie di parentesi graffe**; queste permetterebbero infatti di **generare delle stringhe a partire da cosa queste contengono**, di fatto cambiando il significato ultimo del comando. Questo tipo di espansione si chiama ***brace expansion***.
Quando viene richiesta una *brace expansion* è necessario passare:

- Una **stringa di testo** racchiusa tra separatori (spazi, tab o andate a capo) in cui sia presente una coppia di **parentesi graffe** `{}` non precedute da `$`.
- Tale stringa, la quale ordina una brace expansion, **non deve contenere spazi bianchi** e si divide in 3 parti; la prima e l'ultima (preambolo e postscritto) sono facoltative.
- La parte centrale è **quella racchiusa nelle parentesi graffe** `{}`.
- **Ciascuna stringa presente all'interno delle parentesi graffe rappresenta una possibile scelta** che possono essere aggiunte al preambolo e seguite dal postscritto per formare nuove stringhe.

```
echo C{ia,amin,ul,avol,ub}o
# Il comando viene espanso in echo Ciao Camino Culo Cavolo Cubo.
# Output: Ciao Camino Culo Cavolo Cubo
```

Se vogliamo inserire spazi bianchi o tabulazioni è necessario che questi siano protetti da un **metacarattere di *escape*** `\` oppure quotando la stringa tra **doppi apici**.

```
echo C{avolo cappucc}io
# Output: C{avolo cappucc}io; non viene effettuata la brace expansion
echo C{avolo\ cappucc,"iao sono "}io
# Output: Cavolo cappuccio Ciao sono io
```

E' possibile annidare le brace expansion, con la regola che per prime vengono calcolate **quelle più esterne**.

```
echo
/home/vulpi/{Desktop/{ciao.txt,ciao.zip},Documents/{salve.txt,salve.rar}}

# Output: /home/vulpi/Desktop/ciao.txt /home/vulpi/Desktop/ciao.zip
/home/vulpi/Documents/salve.txt /home/vulpi/Documents/salve.rar
```

È possibile specificare delle variabili da espandere anche usando la brace expansion; in questo caso la **brace expansion ha precedenza sulla variable expansion**.

```
A=ciao
B=coso
echo ${A}{${B},${A}${A}} lol.
# Output ciaocoso ciaociaociao lol.
```

Infine, esiste una particolare forma di brace expansion che permette di specificare **intervalli di caratteri**, separando i due estremi con `..`.

```
echo l{a..f}l
# Output: lal lbl lcl ldl lel lfl
```

Tilde expansion e Wildcards

La ***tilde expansion*** `~` viene sostituita con la *home directory* dell'utente che sta utilizzando la *shell* o di un altro utente, se specificato.
Supponiamo che l'utente `vulpi` sta attualmente utilizzando la *shell*:

```
echo ~
# Output: /home/vulpi
echo ~/
# Output: /home/vulpi/
echo ~/ciao
# Output: /home/vulpi/ciao
```

Supponiamo ora che l'utente `vulpi` stia usando la *shell*, ed esiste nella macchina un altro utente chiamato `draven`.

```
echo ~draven
# Output: /home/draven (o la home directory di draven)
echo ~draven/ciao
# Output /home/draven/ciao
echo ~jinx
# Output: ~jinx
# L'utente jinx non esiste quindi non viene effettuata la tilde expansion.
```

Le **Wildcards** sono *metacaratteri* usati per specificare **caratteri o stringhe generiche**.

- Il metacarattere `*` viene sostituito da **una qualunque sequenza di caratteri, anche vuota**
- Il metacarattere `?` viene sostituito da **esattamente uno e un solo carattere**.
- L'espansione `[elenco]` viene sostituito da **esattamente uno e un solo carattere presente nell'elenco** (vedremo dopo come).

È facile osservare l'uso delle wildcards tramite il comando `ls`:

```
ls ~/Desktop
# Visualizza tutti i file di /home/vulpi/Desktop
ls ~/Desktop/*.sh
# Visualizza tutti i file aventi come estensione .sh di /home/vulpi/Desktop
ls ~/Desktop/ciao*
# Visualizza tutti i file chiamati ciao di /home/vulpi/Desktop
ls ~/Desktop/c?ao.sh
# Visualizza tutti i file chiamati c?ao.sh in /home/vulpi/Desktop, con al
posto di ? un qualsiasi carattere.
```

La Wildcard `[]` permette al suo interno diversi parametri:

- Una lista, senza spazi, di caratteri: nel caso `[abc]` verrà sostituito con un solo carattere tra a,b,c.
- Specificando gli estremi di numeri o caratteri: nel caso `[1-7]` verrà sostituito con un solo carattere tra 1, 2, 3, 4, 5, 6, 7; nel caso `[c-f]` verrà sostituito un solo carattere tra c, d, e, f.
- Specificando il tipo di carattere: `[[:digit:]]` specifica una cifra numerica; `[[:upper:]]` e `[[:lower:]]` specificano rispettivamente una lettera minuscola e maiuscola. Queste **possono essere annidate**, ma solo tra loro stesse.