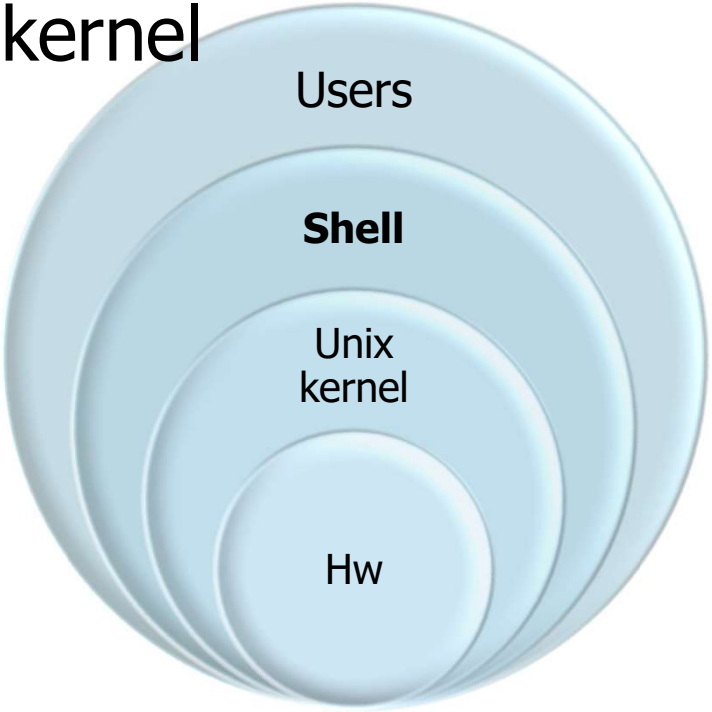
Il sistema operativo UNIX/Linux

Le shell

Stefano Quer
Dipartimento di Automatica e Informatica
Politecnico di Torino

Introduzione alle shell

- Strato più esterno del sistema operativo
  - Fornisce l'interfaccia utente, ovvero interpreta i comandi degli utenti
  - Unica interfaccia prima dell'introduzione dei server grafici
- In Unix la shell non è parte del kernel
  - È un normale processo utente
  - Simile a DOS ma più potente
  - Ambiente di programmazione "nativo del SO"



Introduzione alle shell

- Ogni shell permette
  - La gestione di comandi in linea
    - La shell comprende automaticamente quando il costrutto termina e lo esegue immediatamente
  - La scrittura di programmi (**script**)
    - Memorizzazione dei comandi desiderati in un file
    - Esecuzione dei comandi richiamando il file stesso
- La scrittura di uno script evita di
  - Digitare ripetutamente complesse sequenze di comandi
  - Automatizzare operazioni tediose, ripetitive e prone a errori banali

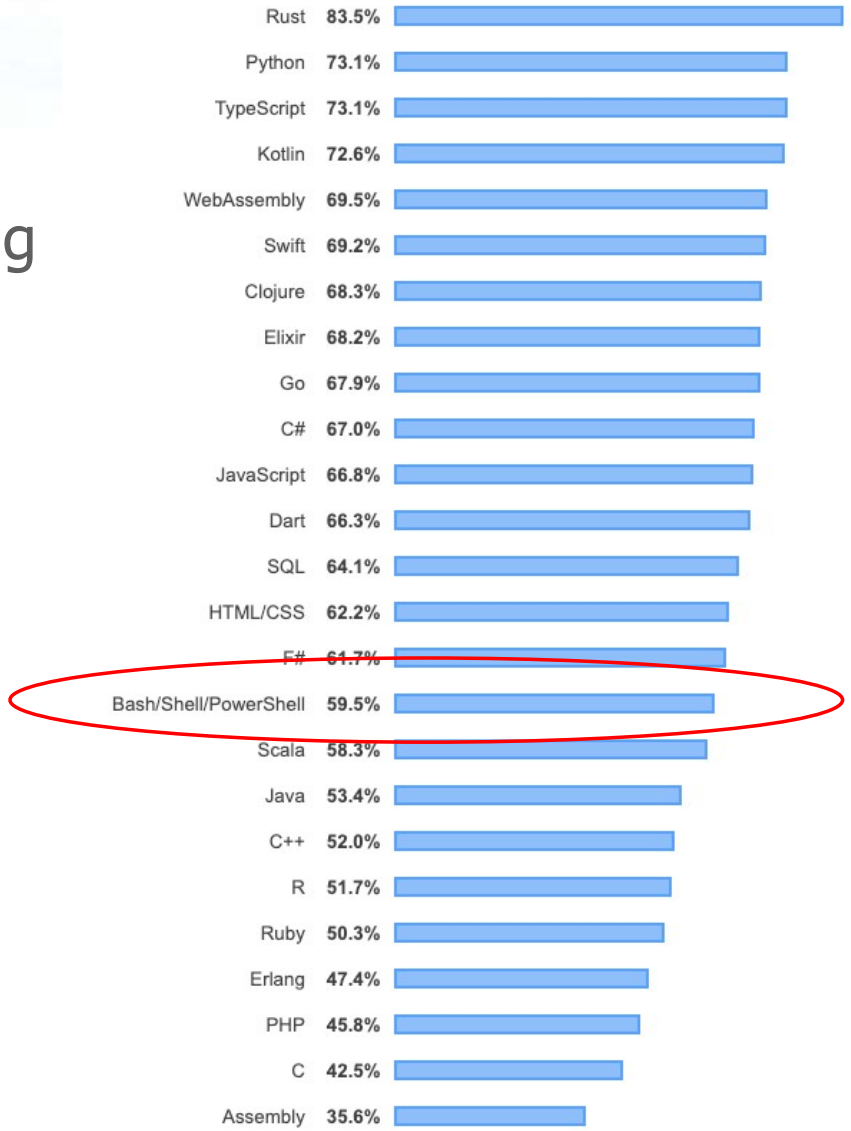
Shell principali

Elenco shell disponibili
cat /etc/shells

Shell	Caratteristiche
Bourne shell (sh)	Shell originaria, molto usata nella programmazione sistemistica in Unix
C-shell (csh)	Shell di Berkeley, ottima per l'uso interattivo e per gli script non di sistema. Utilizza la sintassi del linguaggio C
Korn shell (ksh)	Bourne shell riscritta dall'AT&T per assomigliare di più alla C-shell
Tahoe C-shell (tcsh)	Progetto Tahoe, una C-shell migliorata (sovra-insieme)
Bourne again shell (bash)	È compatibile ma estende csh e ksh. Shell GNU standard; conforme allo standard POSIX; potente ma adatta ai principianti. La maggior parte degli script sh sono eseguibili in bash senza modifiche

Shell principali

- Script di shell? Hug???
  - Da "Stack Overflow"
    - Most Loved Programming Languages In 2019



Bash & co. (59.9%)

Introduzione alle shell

- Shell diverse prevedono comandi diversi
- Spesso **/bin/sh** è un link alla shell in uso
  - La shell di default può essere modificata
    - chsh (change login shell)
  - Versione in uso
    - /bin/bash –version
    - echo $BASH_VERSION

tcsh	bash
set myVar = "ciao"	myVar="ciao"
setenv MY_DIR /home/usr/	export MY_VAR=/home/usr/
if ($str1==$str2) then … else … endif	if test $str1=$str2 then … else … fi if [$str1=$str2]; then … else … fi

Esecuzione di una shell

- Una shell può essere attivata
  - Automaticamente al login
  - In modo annidato dentro un'altra shell
    - Come un programma normale
      - /bin/tcsh, /bin/bash, ...
- Una shell termina digitando
  - Il comando **exit**
  - Il carattere di EOF (tipicamente ctrl-d)
    - Terminando la shell più interna si ritorna a quella esterna

bash ... bash ...
ps oppure ps -l
exit ... exit

Introduzione alla bash

- All'avviamento (e alla terminazione), ogni shell esegue alcuni file di configurazione che contengono i comandi necessari all'inizializzazione (e alla terminazione)
- I **file di avvio** (start-up files) si differenziano in
  - File di login
    - La shell viene eseguita a seguito di autenticazione nel sistema (password)
  - File non di login
    - La shell viene eseguita mediante icona o menu di sistema

Introduzione alla bash

- Alla chiamata di shell con login si attivano
  - Script globali
    - /etc/profile
  - Script utente
    - ~/.bash_profile
    - ~/.bash_login
    - ~/.profile

    … viene eseguito il primo file esistente
  - Si ha un errore in caso di file errato o non leggibile

Introduzione alla bash

- Alla chiamata di shell senza login si attivano
  - ~/.bashrc
  - Tale file spesso si riferisce a ~/.bashrc_profile
  - È anche il file normalmente eseguito nel login da remoto
- All'uscita dal sistema, ovvero per ogni logout, la shell esegue
  - ~/.bash_logout

Espansione della shell

- Alcuni caratteri assumono un significato particolare all'interno delle shell
- Gli script bash mettono a disposizione complessi meccanismi di sostituzione
  - Dopo aver suddiviso la linea di comando in frammenti, la shell espande o risolve tali frammenti, i.e., vi applica diversi tipi di sostituzione con un ordine preciso
    - Parentesi graffe, tilde, variabili e parametri, comandi, espressioni aritmetiche, etc.
  - La sostituzione è complessa e avviene con un ordine ben preciso

Parentesi

- Le parentesi (), [], {}
  - Servono per racchiudere variabili, operazioni aritmetiche, etc.
  - In alcuni casi sono soggette a espansione automatica (brace expansion)

```
nome=Gian
echo $nomemarco

echo {$nome}marco
{Gian}marco
echo ${nome}marco
Gianmarco
```

echo: comando di stampa

Variabile non esistente

Quoting

- Per "quoting" si intende l'utilizzo degli
  - Apici ` `
    - Identificano una stringa al cui interno **non sono espanse** le variabili
    - Non possono essere annidati
  - Le virgolette " "
    - Identificano una stringa al cui interno le variabili **sono espanse**
    - Possono essere annidate
  - Il backslash \
    - Identifica il carattere di escape, ovvero elimina il significato speciale del carattere che lo segue

Esempi

```
➢ myVar="Val ore"
➢ echo $myVar
Val ore

➢ echo 'v = $myVar'
v = $myVar

➢ echo "v = $myVar"
v = Val ore

➢ echo \$myVar
$myVar
➢ echo "virgoletta\""
virgoletta"
```

Osservare l'utilizzo delle variabili: assegnate senza $ utiizzate con il $

' ... ' → nessuna espansione

" ... " → espansione

\ annulla il significato del "metacarattere" successivo

Cattura dello stdout di un comando

- Lo standard output di un comando può essere **catturato** mediante
  - La sequenza di caratteri $(...)
  - Gli apici inversi (back-quote) ``

Alt-96 → `
Alt-239 → `
Alt-123 → {
Alt-125 → }

Versione obsoleta

- In particolare l'output di un comando può essere memorizzato in una variabile

Esempi

```
d=$(date)
echo $d
Fri Nov 22 10:00:0 \
    CET 2013
d=`date`
...
```

```
out=`cat file.txt`
echo $out
... listato file ...

out=`< file.txt`
echo $out
... listato file ...
```

History

- Ogni shell
  - Ricorda l'elenco degli ultimi comandi digitati
  - In bash tale elenco è contenuto nel file **.bash_history**
    - Memorizzato nella home dell'utente
  - Le shell permettono di fare riferimento a tale elenco

Comando	Significato
history	Mostra l'elenco dei comandi eseguiti precedentemente
!n	Segue il comando numero n nel buffer
!str	Esegue l'ultimo comando che inizia con str
^str1^str2	Sostituisce nell'ultimo comando str1 con la str2 e lo esegue nuovamente

Aliasing

- Nelle shell è possibile definire nomi nuovi per comandi esistenti
  - Il commando alias permette di definire tali nomi
    - alias nome="stringa"
      - Definisce un nuovo alias per "stringa"

No spazi intorno al simbolo di =

  - La shell conserva un elenco di alias
    - alias
      - Fornisce l'elenco degli alias attivi nella shell utilizzata
  - Vecchi alias possono venire eliminati
    - unalias nome
      - Elimina l'alias **nome** dalla shell

Esempi

- `alias`

```
alias egrep='egrep --color=auto'
alias emacs='emacs -r -geometry 100x36 -fn 9x15 &'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias ls='ls --color=auto'
alias mx='xdvi -mfmode ljfour:1200'
```

Alias esistenti

- `alias ll="ls -la"`

Definizione di un nuovo alias

- `unalias emacs`
- `unalias ll`

Cancellazione di un alias pre-esistente
(l'eventuale commando ritorna ad essere quello che era)