Bash per il principiante talentuoso

Audace Sailing Team

19 novembre 2025

Introduzione

Obiettivi della presentazione

- Esplorare elementi più avanzati del funzionamento di Bash
- Comprendere permessi dei file
- Utilizzare redirezioni, pipe e alias
- Modificare il proprio ambiente
- Gestire i lavori e i processi
- Comunicare con server remoti via `ssh`

Permessi e proprietà

Struttura dei permessi

- Ogni file ha:
  - proprietario (utente)
  - gruppo
  - permessi per:
    - utente
    - gruppo
    - altri
- Esempio:

```
ls -l file.txt
# -rw-r--r-- 1 utente gruppo 1234 Nov 13 10:00 file.txt
```

Modifica di permessi e proprietario

- Cambiare permessi:

```
chmod u+x script.sh     # aggiunge esecuzione per utente
chmod 755 script.sh     # rwx per utente, rx per gruppo e
```

- Cambiare proprietario (richiede privilegi):

```
sudo chown nuovo_utente file.txt
```

Redirezioni

Output e errori

- Redirezioni principali:
  - `> file` → sovrascrive
  - `>> file` → aggiunge in fondo
  - `2> file` → solo errori (stderr)
  - `> file 2>&1` → stdout + stderr

Esempi

- Scrivere l'output su file:

```
ls > elenco.txt
ls >> elenco.txt
```

- Separare output e errori:

```
comando > out.log 2> errori.log
comando > out.log 2>&1
```

Pipe (|)

Collegare comandi

- La pipe collega output di un comando all'input del successivo
- Permette di costruire "catene" di elaborazione

Esempi

- Scorrere l'output di `ls`:

  ```
  ls | less
  ```

- Cercare processi:

  ```
  ps aux | grep nginx
  ```

- Contare righe con ERROR:

  ```
  cat file.log | grep ERROR | wc -l
  ```

Variabili d'ambiente

Variabili di shell

- Definizione e uso:

```
NOME="Valore"
echo "$NOME"
```

- Esportare ai processi figli:

```
export NOME="Valore"
```

PATH

- Contiene le directory in cui la shell cerca i comandi:

```
echo "$PATH"
```

- Aggiungere una directory:

```
export PATH="$PATH:/home/utente/bin"
```

Alias

Scorciatoie per comandi frequenti

- Definizione:

```
alias ll='ls -lh'
alias gs='git status'
```

- Uso:

```
ll    # equivalente a ls -lh
gs    # equivalente a git status
```

- Inseriti in `~/.bashrc` diventano permanenti

Quoting e sostituzione di comandi

Tipi di quoting

- Apici singoli:

```
echo '$HOME'    # stampa letteralmente $HOME
```

- Doppi apici:

```
echo "$HOME"    # espande la variabile HOME
```

Sostituzione di comandi

- Forma consigliata:

```
echo "Oggi è: $(date)"
```

- `$(...)` esegue il comando e inserisce l'output

Job control e processi

Esecuzione in background

- Avviare un comando in background:

```
comando_lungo &
```

- Gestire i job:

```
jobs        # elenca i job
fg %1       # porta job 1 in foreground
bg %1       # continua job 1 in background
```

Processi persistenti

- Sospendere con `Ctrl+Z`
- Riprendere con `fg` o `bg`
- Eseguire anche dopo chiusura terminale:

```
nohup comando_lungo > out.log 2>&1 &
```

Monitoraggio dei processi (top, htop, altri)

- `top`
  - Strumento standard per monitorare i processi in tempo reale
  - Mostra uso CPU, RAM, load average, uptime
  - Comandi utili:
    - `P` ordina per CPU
    - `M` ordina per RAM
    - `k` invia un segnale (kill) a un processo
    - `q` esce
- `htop`
  - Versione più moderna e interattiva
  - Interfaccia a colori, navigazione con frecce
  - Permette filtraggio, ricerca, kill tramite tasti funzione
  - Visualizzazione immediata di CPU multiple e RAM/swap

Monitoraggio dei processi (top, htop, altri) (2)

- Altri strumenti utili
  - `ps aux` — elenco statico di tutti i processi
  - `pgrep` / `pkill` — cercare o terminare processi per nome
  - `dstat` — monitoraggio combinato CPU, dischi, rete (se installato)
  - `iotop` — mostra processi con maggior I/O su disco (richiede permessi)

Accesso remoto con SSH

Cos'è SSH

- Protocollo sicuro per collegarsi a sistemi remoti
- Offre cifratura, autenticazione e integrità della connessione
- `ssh utente@host`
- Esempio: `ssh alice@server.example.com`

Chiavi SSH

- Generazione chiave:
  - `ssh-keygen -t ed25519`
- Copia della chiave sulla macchina remota:
  - `ssh-copy-id utente@host`
- Vantaggi:
  - non richiede password
  - più sicura dell'autenticazione via password
- MAI condividere la chiave privata!

Copia file con scp

- Da locale a remoto:
  - `scp file.txt utente@host:/percorso/`
- Da remoto a locale:
  - `scp utente@host:/percorso/file.txt .`
- Per directory:
  - `scp -r cartella/ utente@host:/path`

Trasferimenti efficienti con rsync (1)

- Sincronizza due directory/host in modo incrementale.
- Copia solo le differenze $\rightarrow$ molto più veloce di `scp`.
- Mantiene permessi, timestamp, link simbolici, ecc.

Trasferimenti efficienti con rsync (2)

- Da locale a remoto:
  - `rsync -avh cartella/ utente@host:/path/`
  - Con compressione:
    - `rsync -avhz cartella/ utente@host:/path/`
  - Con eliminazione dei file remoti non più presenti in locale:
    - `rsync -avh --delete cartella/ utente@host:/path/`
- Da remoto a locale:
  - `rsync -avh utente@host:/path/cartella/ .`

Trasferimenti efficienti con rsync (3)

- Opzioni utili:
  - `-a` → archivio (mantiene permessi, owner, timestamp...)
  - `-v` → verbose
  - `-h` → output leggibile
  - `--dry-run` → mostra cosa farebbe senza eseguire nulla
  - `--progress` → mostra avanzamento
  - `--exclude \"pattern\"` → esclude file/dir specifici

Tunneling e forwarding

- Port forwarding locale:
  - `ssh -L 8888:localhost:80 utente@host`
- X11 forwarding (se supportato):
  - `ssh -X utente@host`