RIFERIMENTO RAPIDO BASH

Comandi, scripting, pipe, redirezione, controllo dei processi

Basi

echo e Navigazione

```
echo "Hello, World!"     # print text
pwd                      # print working directory
cd /path/to/dir          # change directory
cd ..                    # go up one level
cd ~                     # go to home directory
cd -                     # go to previous directory
```

Elencare e Creare

```
ls                # list files
ls -la            # long format, show hidden
ls -lh            # human-readable sizes
mkdir mydir       # create directory
mkdir -p a/b/c    # create nested directories
```

Copia, Sposta e Rimuovi

```
cp file.txt copy.txt      # copy file
cp -r dir/ backup/        # copy directory recursively
mv old.txt new.txt        # rename / move
rm file.txt               # delete file
rm -r dir/                # delete directory recursively
rm -rf dir/               # force delete (no prompt)
```

Variabili ed Espansione

Variabili

```
name="Alice"              # assign (no spaces!)
echo "$name"              # variable expansion
echo "${name}_file"       # braces for clarity
readonly PI=3.14          # constant
unset name                # delete variable
```

Variabili Speciali

`$0`	Nome dello script
`$1 $2 ...`	Argomenti posizionali
`$#`	Numero di argomenti
`$@`	Tutti gli argomenti (parole separate)
`$*`	Tutti gli argomenti (stringa unica)
`$?`	Stato di uscita dell'ultimo comando
`$$`	ID del processo corrente
`$!`	PID dell'ultimo processo in background

Sostituzione di Comandi e Aritmetica

```
files=$(ls)               # capture output
today=$(date +%Y-%m-%d)   # command substitution
count=$((5 + 3))          # arithmetic: 8
echo $((10 / 3))          # integer division: 3
echo $((10 % 3))          # modulo: 1
```

Operazioni sulle Stringhe

`${#str}`	Lunghezza della stringa
`${str:0:5}`	Sottostringa (offset:lunghezza)
`${str/old/new}`	Sostituisce prima occorrenza
`${str//old/new}`	Sostituisce tutte le occorrenze
`${str^^}`	Maiuscolo
`${str,,}`	Minuscolo

Condizionali

if / elif / else

```
if [[ "$name" == "Alice" ]]; then
    echo "Hi Alice"
elif [[ "$name" == "Bob" ]]; then
    echo "Hi Bob"
else
    echo "Who are you?"
fi
```

Operatori di Test

`-eq  -ne`	Uguale / diverso (interi)
`-lt  -gt`	Minore / maggiore (interi)
`-le  -ge`	Minore o uguale / maggiore o uguale
`==  !=`	Uguale / diverso (stringhe)
`-z "$str"`	Stringa vuota
`-n "$str"`	Stringa non vuota
`-f file`	Il file esiste ed è un file regolare
`-d dir`	La directory esiste
`-e path`	Il percorso esiste (qualsiasi tipo)
`-r  -w  -x`	Leggibile / scrivibile / eseguibile
`&&  ||`	AND / OR logico

Cicli

Ciclo for

```
for fruit in apple banana cherry; do
    echo "$fruit"
done

for f in *.txt; do
    echo "File: $f"
done
```

Ciclo for in stile C

```
for ((i=0; i<5; i++)); do
    echo "$i"
done
```

Ciclo while

```
count=0
while [[ $count -lt 5 ]]; do
    echo "$count"
    ((count++))
done
```

Controllo del Ciclo

`break`	Esci dal ciclo
`continue`	Passa all'iterazione successiva

Funzioni

Definire e Chiamare

```
greet() {
    echo "Hello, $1!"   # $1 = first arg
    return 0            # exit status
}
greet "Alice"           # Hello, Alice!
```

Variabili Locali e Valori di Ritorno

```
add() {
    local sum=$(($1 + $2))
    echo "$sum"         # "return" via stdout
}
result=$(add 3 5)       # capture: 8
```

Pipe e Redirezione

Pipe

```
ls -l | grep ".txt"     # pipe output
cat log | sort | uniq   # chain commands
cmd1 | tee out.txt      # pipe + save to file
```

Redirezione

`cmd > file`	Redirige stdout (sovrascrittura)
`cmd >> file`	Redirige stdout (aggiunta)
`cmd < file`	Redirige stdin da file
`cmd 2> file`	Redirige stderr
`cmd 2>&1`	Redirige stderr su stdout
`cmd &> file`	Redirige stdout + stderr
`cmd << EOF`	Here document (input inline)
`/dev/null`	Scarta output: `cmd > /dev/null`

Operazioni sui File

Visualizzare File

```
cat file.txt            # print entire file
head -n 10 file.txt     # first 10 lines
tail -n 10 file.txt     # last 10 lines
tail -f log.txt         # follow (live updates)
less file.txt           # paginated viewer
```

Contare e Trovare

```
wc -l file.txt          # count lines
wc -w file.txt          # count words
wc -c file.txt          # count bytes
find . -name "*.txt"    # find by name
find . -type d          # find directories
find . -mtime -7        # modified in last 7 days
```

Altri Comandi sui File

`touch file`	Crea file / aggiorna timestamp
`stat file`	Metadati del file (dimensione, date)
`file img.png`	Rileva tipo di file
`diff a.txt b.txt`	Confronta due file
`sort file.txt`	Ordina le righe
`uniq`	Rimuove duplicati adiacenti
`cut -d: -f1`	Estrae campi per delimitatore
`tr 'a-z' 'A-Z'`	Traduce / sostituisce caratteri

Elaborazione del Testo

grep

```
grep "error" log.txt       # search for pattern
grep -i "error" log.txt    # case-insensitive
grep -r "TODO" src/        # recursive search
grep -n "func" file.go     # show line numbers
grep -c "error" log.txt    # count matches
grep -v "debug" log.txt    # invert match
```

sed

```
sed 's/old/new/' file      # replace first per line
sed 's/old/new/g' file     # replace all
sed -i 's/old/new/g' file  # edit in place
sed -n '5,10p' file        # print lines 5-10
sed '/pattern/d' file      # delete matching lines
```

awk

```
awk '{print $1}' file      # print first field
awk -F: '{print $1}' file  # custom delimiter
awk '$3 > 100' file        # filter by field value
awk '{sum+=$1} END{print sum}' file  # sum column
```

Permessi

chmod

```
chmod 755 script.sh    # rwxr-xr-x
chmod +x script.sh     # add execute
chmod -w file.txt      # remove write
chmod u+x,g-w file     # user +exec, group -write
```

Riferimento Permessi

`r (4)`	Lettura
`w (2)`	Scrittura
`x (1)`	Esecuzione
`u / g / o`	Utente / Gruppo / Altri
`755`	Proprietario: rwx, Gruppo/Altri: r-x
`644`	Proprietario: rw-, Gruppo/Altri: r--

Proprietà

```
chown user file.txt          # change owner
chown user:group file.txt    # change owner + group
chown -R user:group dir/     # recursive
```