REFERÊNCIA RÁPIDA DE BASH

Comandos, scripts, pipes, redirecionamento, controle de jobs

Básico

echo & Navegação

```
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
```

Listar & Criar

```
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
```

Copiar, Mover & Remover

```
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)
```

Variáveis & Expansão

Variáveis

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

Variáveis Especiais

`$0`	Nome do script
`$1 $2 ...`	Argumentos posicionais
`$#`	Número de argumentos
`$@`	Todos os argumentos (palavras separadas)
`$*`	Todos os argumentos (string única)
`$?`	Status de saída do último comando
`$$`	PID do processo atual
`$!`	PID do último processo em segundo plano

Substituição de Comando & Aritmética

```
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
```

Operações com Strings

`${#str}`	Comprimento da string
`${str:0:5}`	Substring (offset:tamanho)
`${str/old/new}`	Substituir primeira ocorrência
`${str//old/new}`	Substituir todas as ocorrências
`${str^^}`	Maiúsculas
`${str,,}`	Minúsculas

Condicionais

if / elif / else

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

Operadores de Teste

`-eq  -ne`	Inteiro igual / diferente
`-lt  -gt`	Inteiro menor / maior que
`-le  -ge`	Inteiro menor/maior ou igual
`==  !=`	String igual / diferente
`-z "$str"`	String vazia
`-n "$str"`	String não vazia
`-f file`	Arquivo existe e é regular
`-d dir`	Diretório existe
`-e path`	Caminho existe (qualquer tipo)
`-r  -w  -x`	Legível / gravável / executável
`&&  ||`	E lógico / OU lógico

Laços

Laço for

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

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

Laço for estilo C

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

Laço while

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

Controle de Laço

`break`	Sair do laço
`continue`	Pular para próxima iteração

Funções

Definir & Chamar

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

Variáveis Locais & Retorno

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

Pipes & Redirecionamento

Pipes

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

Redirecionamento

`cmd > file`	Redirecionar stdout (sobrescrever)
`cmd >> file`	Redirecionar stdout (append)
`cmd < file`	Redirecionar stdin do arquivo
`cmd 2> file`	Redirecionar stderr
`cmd 2>&1`	Redirecionar stderr para stdout
`cmd &> file`	Redirecionar stdout + stderr
`cmd << EOF`	Here document (entrada inline)
`/dev/null`	Descartar saída: `cmd > /dev/null`

Operações com Arquivos

Visualizar Arquivos

```
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
```

Contar & Buscar

```
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
```

Outros Comandos de Arquivo

`touch file`	Criar arquivo / atualizar timestamp
`stat file`	Metadados do arquivo (tamanho, datas)
`file img.png`	Detectar tipo de arquivo
`diff a.txt b.txt`	Comparar dois arquivos
`sort file.txt`	Ordenar linhas
`uniq`	Remover duplicatas adjacentes
`cut -d: -f1`	Extrair campos por delimitador
`tr 'a-z' 'A-Z'`	Traduzir / substituir caracteres

Processamento de Texto

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
```

Permissões

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
```

Referência de Permissões

`r (4)`	Leitura
`w (2)`	Escrita
`x (1)`	Execução
`u / g / o`	Usuário / Grupo / Outros
`755`	Dono: rwx, Grupo/Outros: r-x
`644`	Dono: rw-, Grupo/Outros: r--

Propriedade

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