RÉFÉRENCE RAPIDE BASH

Commandes, scripts, pipes, redirections, contrôle des tâches

Bases

echo & Navigation

```
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
```

Lister & Créer

```
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
```

Copier, Déplacer & Supprimer

```
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)
```

Variables & Expansion

Variables

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

Variables spéciales

`$0`	Nom du script
`$1 $2 ...`	Arguments positionnels
`$#`	Nombre d'arguments
`$@`	Tous les arguments (mots séparés)
`$*`	Tous les arguments (chaîne unique)
`$?`	Code de sortie de la dernière commande
`$$`	PID du processus courant
`$!`	PID du dernier processus en arrière-plan

Substitution de commande & Arithmétique

```
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
```

Opérations sur les chaînes

`${#str}`	Longueur de la chaîne
`${str:0:5}`	Sous-chaîne (offset:longueur)
`${str/old/new}`	Remplacer la première occurrence
`${str//old/new}`	Remplacer toutes les occurrences
`${str^^}`	Majuscules
`${str,,}`	Minuscules

Conditions

if / elif / else

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

Opérateurs de test

`-eq  -ne`	Entier égal / différent
`-lt  -gt`	Entier inférieur / supérieur
`-le  -ge`	Entier inférieur/supérieur ou égal
`==  !=`	Chaîne égale / différente
`-z "$str"`	Chaîne vide
`-n "$str"`	Chaîne non vide
`-f file`	Fichier ordinaire existant
`-d dir`	Répertoire existant
`-e path`	Chemin existant (tout type)
`-r  -w  -x`	Lisible / inscriptible / exécutable
`&&  ||`	ET / OU logique

Boucles

Boucle for

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

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

Boucle for style C

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

Boucle while

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

Contrôle de boucle

`break`	Quitter la boucle
`continue`	Passer à l'itération suivante

Fonctions

Définir & Appeler

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

Variables locales & Valeurs de retour

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

Pipes & Redirections

Pipes

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

Redirections

`cmd > file`	Rediriger stdout (écraser)
`cmd >> file`	Rediriger stdout (ajouter)
`cmd < file`	Lire stdin depuis un fichier
`cmd 2> file`	Rediriger stderr
`cmd 2>&1`	Rediriger stderr vers stdout
`cmd &> file`	Rediriger stdout + stderr
`cmd << EOF`	Here document (entrée inline)
`/dev/null`	Ignorer la sortie : `cmd > /dev/null`

Opérations sur les fichiers

Afficher des fichiers

```
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
```

Compter & Rechercher

```
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
```

Autres commandes fichier

`touch file`	Créer un fichier / mettre à jour l'horodatage
`stat file`	Métadonnées (taille, dates)
`file img.png`	Détecter le type de fichier
`diff a.txt b.txt`	Comparer deux fichiers
`sort file.txt`	Trier les lignes
`uniq`	Supprimer les doublons adjacents
`cut -d: -f1`	Extraire des champs par délimiteur
`tr 'a-z' 'A-Z'`	Traduire / remplacer des caractères

Traitement de texte

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
```

Permissions

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
```

Référence des permissions

`r (4)`	Lecture
`w (2)`	Écriture
`x (1)`	Exécution
`u / g / o`	Utilisateur / Groupe / Autres
`755`	Propriétaire : rwx, Groupe/Autres : r-x
`644`	Propriétaire : rw-, Groupe/Autres : r--

Propriété

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