

BASH QUICK REFERENCE

Basics

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
```

Listing & Creating

```
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
```

Copy, Move & Remove

```
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
```

Special Variables

<code>\$0</code>	Script name
<code>\$1 \$2 ...</code>	Positional arguments
<code>\$#</code>	Number of arguments

<code>\$@</code>	All arguments (separate words)
<code>\$*</code>	All arguments (single string)
<code>\$?</code>	Exit status of last command
<code>\$\$</code>	Current process ID
<code>\$!</code>	PID of last background process

Command Substitution & Arithmetic

```
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
```

String Operations

<code>\${#str}</code>	String length
<code>\${str:0:5}</code>	Substring (offset:length)
<code>\${str/old/new}</code>	Replace first match
<code>\${str//old/new}</code>	Replace all matches
<code>\${str^^}</code>	Uppercase
<code>\${str,,}</code>	Lowercase

Conditionals

if / elif / else

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

Test Operators

<code>-eq -ne</code>	Integer equal / not equal
<code>-lt -gt</code>	Integer less / greater than

<code>-le -ge</code>	Integer less/greater or equal
<code>== !=</code>	String equal / not equal
<code>-z "\$str"</code>	String is empty
<code>-n "\$str"</code>	String is not empty
<code>-f file</code>	File exists and is regular
<code>-d dir</code>	Directory exists
<code>-e path</code>	Path exists (any type)
<code>-r -w -x</code>	Readable / writable / executable
<code>&& </code>	Logical AND / OR

Loops

for Loop

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

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

C-style for Loop

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

while Loop

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

Loop Control

<code>break</code>	Exit the loop
<code>continue</code>	Skip to next iteration

BASH QUICK REFERENCE (continued)

Functions

Defining & Calling

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

Local Variables & Return Values

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

Pipes & Redirection

Pipes

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

Redirection

<code>cmd > file</code>	Redirect stdout (overwrite)
<code>cmd >> file</code>	Redirect stdout (append)
<code>cmd < file</code>	Redirect stdin from file
<code>cmd 2> file</code>	Redirect stderr
<code>cmd 2>&1</code>	Redirect stderr to stdout
<code>cmd &> file</code>	Redirect stdout + stderr
<code>cmd << EOF</code>	Here document (inline input)
<code>/dev/null</code>	Discard output: <code>cmd > /dev/null</code>

File Operations

Viewing Files

```
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
```

Counting & Finding

```
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
```

Other File Commands

<code>touch file</code>	Create file / update timestamp
<code>stat file</code>	File metadata (size, dates)
<code>file img.png</code>	Detect file type
<code>diff a.txt b.txt</code>	Compare two files
<code>sort file.txt</code>	Sort lines
<code>uniq</code>	Remove adjacent duplicates
<code>cut -d: -f1</code>	Extract fields by delimiter
<code>tr 'a-z' 'A-Z'</code>	Translate / replace characters

Text Processing

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
```

Permission Reference

<code>r (4)</code>	Read
<code>w (2)</code>	Write
<code>x (1)</code>	Execute
<code>u / g / o</code>	User / Group / Others
<code>755</code>	Owner: rwx, Group/Other: r-x
<code>644</code>	Owner: rw-, Group/Other: r--

Ownership

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