# Bash Shell and scripting

**START HERE**

**`cmd > <file>`**
Write output to a file

**`"$var"`**
Use a variable, quoted

**`cmd >> <file>`**
Append output

**`$(cmd)`**
Insert a command's output

**`cmd1 | cmd2`**
Pipe output onward

**`cmd1 && cmd2`**
Run second only on success

**`cmd 2>&1`**
Merge errors into output

**`Ctrl+R`**
Search command history

**Read line** split into words → `expand` **Expansions** braces, $var, $(cmd), globs → `redirect` **Redirections** stdin, stdout, stderr → `run` **Command** exit status in $?

## CONCEPTS

Bash expands a line first (variables, globs, substitutions), then runs it. Quotes decide what gets expanded.

Exit status 0 means success, anything else failure. && , || and if all test it.

Always quote "$var": unquoted values are split at spaces and globbed.

## REDIRECTION & PIPES

**`cmd > <file>`**
Stdout to file, overwrite
*Set noclobber to refuse overwriting; >| forces it.*

**`cmd >> <file>`**
Stdout appended to file

**`cmd < <file>`**
Read stdin from file

**`cmd1 | cmd2`**
Pipe stdout into next command

**`cmd 2> <file>`**
Stderr to file

**`cmd > <file> 2>&1`**
Stdout and stderr to file
*Order matters: the 2>&1 must come after > file.*

**`cmd &> <file>`**
Shorthand for both streams

**`cmd > /dev/null 2>&1`**
Discard all output

**`cmd | tee <file>`**
Show output and save it

**`cmd <<< "<text>"`**
Feed a string as stdin

**`diff <(cmd1) <(cmd2)`**
Process substitution as a file

**`cmd |& cmd2`**
Pipe stdout and stderr

**`cmd <<'EOF'`**
Here-document, no expansion
*Unquoted EOF expands variables inside; end with a line EOF.*

**`exec > <file>`**
Redirect the rest of the script

## QUOTING & EXPANSION

**`"$var"`**
Expand, keep as one word

**`'$var'`**
Literal text, no expansion

## QUOTING & EXPANSION (cont.)

**`$(cmd)`**
Command substitution

**`*.txt`**
Glob: any characters

**`file?.txt`**
Glob: exactly one character

**`{a,b,c}`**
Brace expansion list
*mkdir -p proj/{src,docs} makes both.*

**`{1..10}`**
Brace expansion range

**`$((a + b))`**
Arithmetic expansion

**`~ / ~user`**
Home directory

**`\<char>`**
Escape one character

**`**/*.txt`**
Recursive glob
*Needs shopt -s globstar.*

## VARIABLES

**`name=<value>`**
Set a variable (no spaces)

**`export name=<value>`**
Pass to child processes

**`$1 $2 ... $@`**
Script arguments
*Use "$@" to keep arguments separate.*

**`$?`**
Exit status of last command

**`$#`**
Number of arguments

**`$$ / $!`**
PID of shell / last background job

**`${var:-<default>}`**
Use default if unset or empty

**`${#var}`**
Length of value

**`${var#<pat>} / ${var%<pat>}`**
Strip prefix / suffix
*Doubled ## and %% strip the longest match.*

**`${var/<pat>/<new>}`**
Replace first match

**`arr=(a b c)`**
Create an array

**`"${arr[@]}"`**
All array elements

**`local name=<value>`**
Function-scoped variable

## CONTROL FLOW

**`if cmd; then ...; fi`**
Branch on exit status

**`[[ -f <file> ]]`**
Test: regular file exists
*Also -d directory, -e any, -s non-empty, -r/-w/-x.*

**`[[ "$a" == "$b" ]]`**
Test: strings equal

**`[[ $n -lt 5 ]]`**
Test: numeric compare
*-eq -ne -lt -le -gt -ge; or use (( n < 5 )).*

**`[[ -z "$s" ]] / [[ -n "$s" ]]`**
Test: empty / non-empty

**`for x in a b c; do ...; done`**
Loop over words

**`for f in *.txt; do ...; done`**
Loop over files

**`for ((i=0; i<n; i++)); do ...; done`**
C-style loop

**`while read -r line; do ...; done < <file>`**
Loop over lines

**`case $x in a) ...;; *) ...;; esac`**
Match patterns

**`cmd1 || cmd2`**
Run second only on failure

**`name() { ...; }`**
Define a function

**`[[ $s =~ <regex> ]]`**
Regex match, groups in BASH_REMATCH

## COMMAND LINE (READLINE)

**Ctrl** + **R**
Search history backwards
*Press again for older matches.*

**Ctrl** + **A** / **Ctrl** + **E**
Start / end of line

**Ctrl** + **C**
Interrupt running command

**Ctrl** + **L** Clear the screen

**Ctrl** + **U** / **Ctrl** + **K**
Cut to line start / end

**Ctrl** + **W** Cut previous word

**Ctrl** + **Y** Paste last cut text

**Alt** + **B** / **Alt** + **F**
Back / forward one word

## COMMAND LINE (READLINE) (cont.)

**Alt** + **.**
Insert last argument of previous command

**Ctrl** + **D**
Exit shell or end input

**Tab**
Complete command or path

**Ctrl** + **X** › **Ctrl** + **E**
Edit the line in $EDITOR

**Ctrl** + **_** Undo last edit

**`!!`**
Repeat the previous command
*sudo !! reruns it with sudo.*

**`history | grep <text>`**
Find an old command

## JOBS & PROCESSES

**`cmd &`**
Run in the background

**Ctrl** + **Z**
Suspend foreground job

**`jobs`**
List jobs

**`bg / fg`**
Resume job in back / foreground
*Add %<n> to pick job number n.*

**`kill <pid>`**
Ask a process to stop

## SAFE SCRIPTS

**`#!/usr/bin/env bash`**
Shebang line

**`set -euo pipefail`**
Stop on errors and unset vars
*Has caveats: -e is ignored inside conditions such as if or ||.*

**`trap '<cmd>' EXIT`**
Run cleanup when script ends

**`set -x`**
Trace each command run

**`source <file> / . <file>`**
Run a file in current shell

Use [[ ]] over [ ] in bash scripts; it does not split or glob unquoted variables.

## HELP

**`help <builtin>`**
Help for a builtin

**`man bash`**
Full manual