Bash scripting

Jakub Pieńkowski

2025-06-19

Bash

Original author(s)	Brian Fox
Developer(s)	Chet Ramey
Initial release	8 June 1989; 35 years ago
Stable release	5.2.37[1] / 23 September 2024
Repository	git.savannah.gnu.org/cgit/bash.git
Written in	C

https://en.wikipedia.org/wiki/Bash_(Unix_shell)

Getting Bash

```
# Macos
$ brew install bash

# Debian/Ubuntu
$ apt-get install bash

# Arch Linux
$ pacman -S bash

# Windows
# Install together with Git
# https://git-scm.com/downloads/win
```

Useful commands

man

help

type

Flow control

```
case "$feature" in
elixir) pkgs+=(elixir inotify-tools) ;;
postgresql) pkgs+=(postgresql) ;;
nginx) pkgs+=(nginx-extras) ;;
docker) module_namespaced_docker ;;
esac
```

```
if [ "$in_cmd" = 1 ]; then
  cmd+=("$arg")
  continue
fi
```

```
for j in "${enabled[@]}"; do
  if [ "$j" = "$i" ]; then
    found=1
    break
  fi
done
```

Types

```
declare -a packages=(
  git
  neovim
  php-cli
  php-xml
  php-curl
  nodejs
  npm
  docker-compose
  waypipe
)
```

```
[0] jakub@miho:~
$ a=1
[0] jakub@miho:~
$ a+=2
[0] jakub@miho:~
$ echo "$a"
12
```

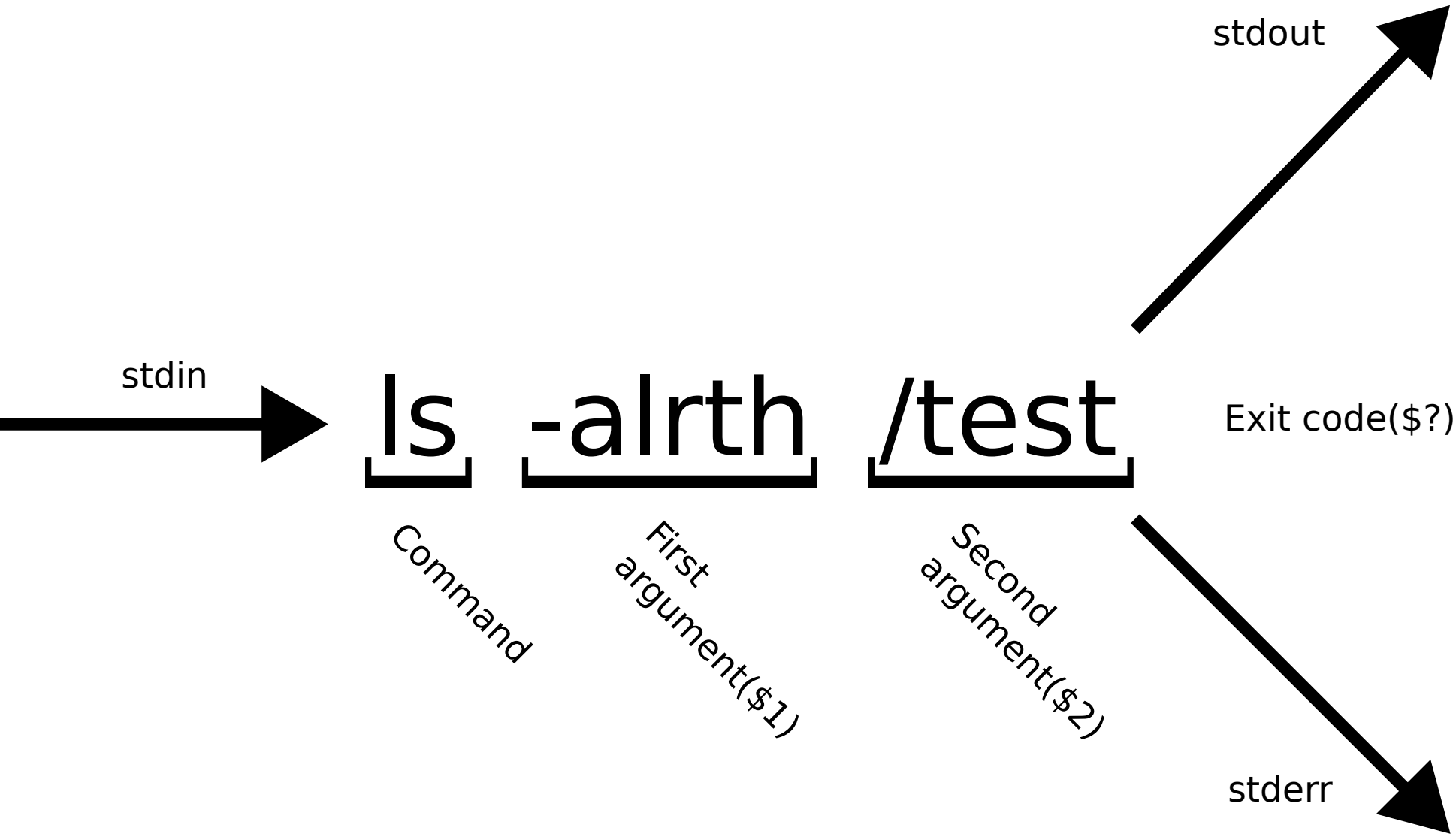
```
declare -A apt_remaps=(
  ["elasticsearch-7"]="https://artifacts.elastic.co/packages/7.x/apt"
  ["nodesource-18"]="https://deb.nodesource.com/node_18.x"
  ["sury-php"]="https://packages.sury.org/php/"
  ["mysql-8"]="http://repo.mysql.com/apt/debian/"
)
```

Functions

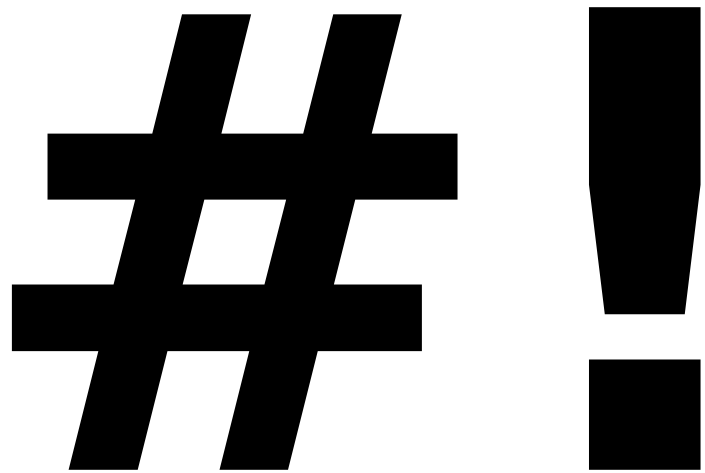
```
[0] jakub@miho:~
$ function1() { a=2; }
[0] jakub@miho:~
$ a=1
[0] jakub@miho:~
$ function1
[0] jakub@miho:~
$ echo "$a"
2
```

```
$ function1() { printf "%s\n" "$@"; }
[0] jakub@miho:~
$ function1 1 2 3
1
2
3
```

```
[0] jakub@miho:~
$ function1() { declare a=2; }
[0] jakub@miho:~
$ a=1
[0] jakub@miho:~
$ function1
[0] jakub@miho:~
$ echo "$a"
1
```


stdout
stdin
ls -alrth /test
Exit code($?)
Command
First argument($1)
Second argument($2)
stderr

Pipelines



Network services

curl

ssh

wget

nc

Challenge 1

Download success notification

Arguments:
- $1 - download URL
- $2 - notification URL

curl://

Strict mode

```
#!/usr/bin/env bash

set -euo pipefail
shopt -s nullglob inherit_errexit lastpipe

```

- errexit
- nounset
- pipefail
- inherit_errexit
- nullglob
- lastpipe

Linting

```
$ shellcheck bash-workshops.sh

In bash-workshops.sh line 23:
declare -a unused=()
          ^----^ SC2034 (warning): unused appears unused. Verify use (or export if used externally).


In bash-workshops.sh line 50:
        echo ${output[@]}
             ^----------^ SC2068 (error): Double quote array expansions to avoid re-splitting elements.

For more information:
  https://www.shellcheck.net/wiki/SC2068 -- Double quote array expansions to ...
  https://www.shellcheck.net/wiki/SC2034 -- unused appears unused. Verify use...
```

https://www.shellcheck.net/

Pitfalls

- "new folder" vs new folder
- declare var=$(failing cmd)
- echo "$(failing cmd)"
- cat file | grep line > file

...

https://mywiki.wooledge.org/BashPitfalls

Myth:
Bash is hard to debug

Have fun spotting the faulty line

```
$ bash challenge1.sh
ls: cannot access '/app': No such file or directory
```

Better?

```
$ bash challenge1.sh
ls: cannot access '/qwe': No such file or directory
Failing with exit code 2 at challenge1.sh:18 in command: ls /qwe
```

Traps

```
on_error() {
    declare \
        cmd=$BASH_COMMAND \
        exit_code=$?
    echo "Failing with exit code ${exit_code} at ${*} in command: ${cmd}" >&2
    exit "$exit_code"
}

trap 'on_error ${BASH_SOURCE[0]}:${BASH_LINENO[0]}' ERR
```

Stacktraces

```
$ bash challenge1.sh
ls: cannot access '/qwe': No such file or directory
Failing with exit code 2 in command: ls /qwe
        challenge1.sh:25:subfunction2
        challenge1.sh:29:subfunction1
        challenge1.sh:33:function1
        challenge1.sh:37:main
        challenge1.sh:40:main
```

```
3   set -eEuo pipefail
1   on_error() {
2   ¦ declare \
3   ¦ ¦ cmd=$BASH_COMMAND \
4   ¦ ¦ exit_code=$? \
5   ¦ ¦ i=0 \
6   ¦ ¦ end="${#FUNCNAME[@]}" \
7   ¦ ¦ next
8   ¦ end=$((end - 1))
9   ¦ echo "Failing with exit code ${exit_code} in command: ${cmd}" >&2
10  ¦ while [ "$i" != "$end" ]; do
11  ¦ ¦ next=$((i + 1))
12  ¦ ¦ echo "¦ ${BASH_SOURCE["$next"]:-}:${BASH_LINENO["$i"]}:${FUNCNAME["$next"]}" >&2
13  ¦ ¦ i=$next
14  ¦ done
15  ¦ exit "$exit_code"
16  }
17  trap on_error ERR
```

Myth:
Bash is complex

```
$ curl -sSL https://registry.npmjs.org/sealious \
> | jq -r --arg arg clone '.versions | to_entries | last | .value.dependencies | to_entries[] | select(.key == $arg)
{
  "key": "clone",
  "value": "^1.0.2"
}
```

Working with structured data

./jq

XMLStarlet

sed

awk

while read

Challenge 2

Checking NPM modules updates

Arguments:
- $1 - Package name from npm
- $2 - Dependency name

Output:
- Version requirement from latest release

Sources:

Bash logo: https://en.wikipedia.org/wiki/File:Gnu-bash-logo.svg
jq logo: https://jqlang.org/
Healthchecks logo: https://github.com/healthchecks/healthchecks
curl logo: https://curl.se/logo/
NPM logo: https://github.com/npm/logos