Bash Cheat Sheet

A reference covering essential commands and scripting concepts. This cheat sheet encompasses everything from basic file operations to advanced system administration tasks, including directory navigation, file permissions, process management, networking utilities, and bash scripting fundamentals. It serves as both a quick reference for common commands and a learning resource for understanding core Unix/Linux shell operations.

Command History

```
!!            # Run the last command

touch foo.sh
chmod +x !$   # !$ is the last argument of the last command i.e. foo.sh
```

Navigating Directories

```
pwd                       # Print current directory path
ls                        # List directories
ls -a|--all               # List directories including hidden
ls -l                     # List directories in long form
ls -l -h|--human-readable # List directories in long form with human
readable sizes
ls -t                     # List directories by modification time, newest
first
stat foo.txt              # List size, created and modified timestamps for a
file
stat foo                  # List size, created and modified timestamps for a
directory
tree                      # List directory and file tree
tree -a                   # List directory and file tree including hidden
tree -d                   # List directory tree
cd foo                    # Go to foo sub-directory
cd                        # Go to home directory
cd ~                      # Go to home directory
cd -                      # Go to last directory
pushd foo                 # Go to foo sub-directory and add previous
directory to stack
popd                      # Go back to directory in stack saved by `pushd`
```

Creating Directories

```
mkdir foo                        # Create a directory
mkdir foo bar                    # Create multiple directories
mkdir -p|--parents foo/bar       # Create nested directory
mkdir -p|--parents {foo,bar}/baz # Create multiple nested directories

mktemp -d|--directory            # Create a temporary directory
```

Moving Directories

```
cp -R|--recursive foo bar                                  # Copy directory
mv foo bar                                                 # Move directory

rsync -z|--compress -v|--verbose /foo /bar                 # Copy directory,
overwrites destination
rsync -a|--archive -z|--compress -v|--verbose /foo /bar # Copy directory,
without overwriting destination
rsync -avz /foo username@hostname:/bar                     # Copy local
directory to remote directory
rsync -avz username@hostname:/foo /bar                     # Copy remote
directory to local directory
```

Deleting Directories

```
rmdir foo                        # Delete empty directory
rm -r|--recursive foo            # Delete directory including contents
rm -r|--recursive -f|--force foo # Delete directory including contents,
ignore nonexistent files and never prompt
```

Creating Files

```
touch foo.txt          # Create file or update existing files modified
timestamp
touch foo.txt bar.txt  # Create multiple files
touch {foo,bar}.txt    # Create multiple files
touch test{1..3}       # Create test1, test2 and test3 files
touch test{a..c}       # Create testa, testb and testc files

mktemp                 # Create a temporary file
```

Standard Output, Standard Error and Standard Input

```
echo "foo" > bar.txt       # Overwrite file with content
echo "foo" >> bar.txt      # Append to file with content

ls exists 1> stdout.txt    # Redirect the standard output to a file
ls noexist 2> stderror.txt # Redirect the standard error output to a file
ls 2>&1 > out.txt          # Redirect standard output and error to a file
ls > /dev/null             # Discard standard output and error

read foo                   # Read from standard input and write to the
variable foo
```

Moving Files

```
cp foo.txt bar.txt                                  # Copy file
mv foo.txt bar.txt                                  # Move file

rsync -z|--compress -v|--verbose /foo.txt /bar    # Copy file quickly if not
changed
rsync z|--compress -v|--verbose /foo.txt /bar.txt # Copy and rename file
quickly if not changed
```

Deleting Files

```
rm foo.txt            # Delete file
rm -f|--force foo.txt # Delete file, ignore nonexistent files and never
prompt
```

Reading Files

```
cat foo.txt           # Print all contents
less foo.txt          # Print some contents at a time (g - go to top of
file, SHIFT+g, go to bottom of file, /foo to search for 'foo')
head foo.txt          # Print top 10 lines of file
tail foo.txt          # Print bottom 10 lines of file
open foo.txt          # Open file in the default editor
wc foo.txt            # List number of lines words and characters in the
file
```

File Permissions

#	Permission	rwx	Binary
7	read, write and execute	rwx	111
6	read and write	rw-	110
5	read and execute	r-x	101
4	read only	r--	100
3	write and execute	-wx	011
2	write only	-w-	010
1	execute only	--x	001
0	none	---	000

For a directory, execute means you can enter a directory.

User	Group	Others	Description
6	4	4	User can read and write, everyone else can read (Default file permissions)
7	5	5	User can read, write and execute, everyone else can read and execute (Default directory permissions)

- u - User
- g - Group
- o - Others
- a - All of the above

```
ls -l /foo.sh            # List file permissions
chmod +100 foo.sh        # Add 1 to the user permission
chmod -100 foo.sh        # Subtract 1 from the user permission
chmod u+x foo.sh         # Give the user execute permission
chmod g+x foo.sh         # Give the group execute permission
chmod u-x,g-x foo.sh     # Take away the user and group execute permission
chmod u+x,g+x,o+x foo.sh # Give everybody execute permission
chmod a+x foo.sh         # Give everybody execute permission
chmod +x foo.sh          # Give everybody execute permission
```

Finding Files

Find binary files for a command.

```
type wget                                   # Find the binary
which wget                                  # Find the binary
whereis wget                                # Find the binary, source, and
manual page files
```

`locate` uses an index and is fast.

```
updatedb                                    # Update the index

locate foo.txt                              # Find a file
locate --ignore-case                        # Find a file and ignore case
locate f*.txt                               # Find a text file starting with
'f'
```

`find` doesn't use an index and is slow.

```
find /path -name foo.txt                    # Find a file
find /path -iname foo.txt                   # Find a file with case
insensitive search
find /path -name "*.txt"                    # Find all text files
find /path -name foo.txt -delete            # Find a file and delete it
find /path -name "*.png" -exec pngquant {} # Find all .png files and execute
pngquant on it
find /path -type f -name foo.txt            # Find a file
find /path -type d -name foo                # Find a directory
find /path -type l -name foo.txt            # Find a symbolic link
find /path -type f -mtime +30               # Find files that haven't been
modified in 30 days
find /path -type f -mtime +30 -delete       # Delete files that haven't been
modified in 30 days
```

Find in Files

```
grep 'foo' /bar.txt                          # Search for 'foo' in file
'bar.txt'
grep 'foo' /bar -r|--recursive               # Search for 'foo' in directory
'bar'
grep 'foo' /bar -R|--dereference-recursive  # Search for 'foo' in directory
'bar' and follow symbolic links
grep 'foo' /bar -l|--files-with-matches     # Show only files that match
grep 'foo' /bar -L|--files-without-match    # Show only files that don't
match
grep 'Foo' /bar -i|--ignore-case            # Case insensitive search
```

```
grep 'foo' /bar -x|--line-regexp            # Match the entire line
grep 'foo' /bar -C|--context 1              # Add N line of context above
and below each search result
grep 'foo' /bar -v|--invert-match           # Show only lines that don't
match
grep 'foo' /bar -c|--count                  # Count the number lines that
match
grep 'foo' /bar -n|--line-number            # Add line numbers
grep 'foo' /bar --colour                    # Add colour to output
grep 'foo\|bar' /baz -R                     # Search for 'foo' or 'bar' in
directory 'baz'
grep --extended-regexp|-E 'foo|bar' /baz -R # Use regular expressions
egrep 'foo|bar' /baz -R                     # Use regular expressions
```

Replace in Files

```
sed 's/fox/bear/g' foo.txt              # Replace fox with bear in foo.txt
and output to console
sed 's/fox/bear/gi' foo.txt             # Replace fox (case insensitive)
with bear in foo.txt and output to console
sed 's/red fox/blue bear/g' foo.txt     # Replace red with blue and fox
with bear in foo.txt and output to console
sed 's/fox/bear/g' foo.txt > bar.txt    # Replace fox with bear in foo.txt
and save in bar.txt
sed 's/fox/bear/g' foo.txt -i|--in-place # Replace fox with bear and
overwrite foo.txt
```

Symbolic Links

```
ln -s|--symbolic foo bar            # Create a link 'bar' to the 'foo'
folder
ln -s|--symbolic -f|--force foo bar # Overwrite an existing symbolic link
'bar'
ls -l                               # Show where symbolic links are pointing
```

Compressing Files

zip

Compresses one or more files into *.zip files.

```
zip foo.zip /bar.txt                # Compress bar.txt into foo.zip
zip foo.zip /bar.txt /baz.txt       # Compress bar.txt and baz.txt into
```

```
foo.zip
zip foo.zip /{bar,baz}.txt          # Compress bar.txt and baz.txt into
foo.zip
zip -r|--recurse-paths foo.zip /bar # Compress directory bar into foo.zip
```

gzip

Compresses a single file into *.gz files.

```
gzip /bar.txt foo.gz           # Compress bar.txt into foo.gz and then
delete bar.txt
gzip -k|--keep /bar.txt foo.gz # Compress bar.txt into foo.gz
```

tar -c

Compresses (optionally) and combines one or more files into a single *.tar,* .tar.gz, *.tpz or* .tgz file.

```
tar -c|--create -z|--gzip -f|--file=foo.tgz /bar.txt /baz.txt # Compress
bar.txt and baz.txt into foo.tgz
tar -c|--create -z|--gzip -f|--file=foo.tgz /{bar,baz}.txt    # Compress
bar.txt and baz.txt into foo.tgz
tar -c|--create -z|--gzip -f|--file=foo.tgz /bar              # Compress
directory bar into foo.tgz
```

Decompressing Files

unzip

```
unzip foo.zip          # Unzip foo.zip into current directory
```

gunzip

```
gunzip foo.gz           # Unzip foo.gz into current directory and delete
foo.gz
gunzip -k|--keep foo.gz # Unzip foo.gz into current directory
```

tar -x

```
tar -x|--extract -z|--gzip -f|--file=foo.tar.gz # Un-compress foo.tar.gz
into current directory
```

```
tar -x|--extract -f|--file=foo.tar              # Un-combine foo.tar into
current directory
```

Disk Usage

```
df                    # List disks, size, used and available space
df -h|--human-readable # List disks, size, used and available space in a
human readable format

du                    # List current directory, subdirectories and file
sizes
du /foo/bar           # List specified directory, subdirectories and file
sizes
du -h|--human-readable # List current directory, subdirectories and file
sizes in a human readable format
du -d|--max-depth     # List current directory, subdirectories and file
sizes within the max depth
du -d 0               # List current directory size
```

Memory Usage

```
free                  # Show memory usage
free -h|--human       # Show human readable memory usage
free -h|--human --si  # Show human readable memory usage in power of 1000
instead of 1024
free -s|--seconds 5   # Show memory usage and update continuously every
five seconds
```

Packages

```
apt update                  # Refreshes repository index
apt search wget             # Search for a package
apt show wget               # List information about the wget package
apt list --all-versions wget # List all versions of the package
apt install wget            # Install the latest version of the wget
package
apt install wget=1.2.3      # Install a specific version of the wget
package
apt remove wget             # Removes the wget package
apt upgrade                 # Upgrades all upgradable packages
```

Shutdown and Reboot

```
shutdown                    # Shutdown in 1 minute
shutdown now "Cya later"    # Immediately shut down
shutdown +5 "Cya later"     # Shutdown in 5 minutes

shutdown --reboot           # Reboot in 1 minute
shutdown -r now "Cya later" # Immediately reboot
shutdown -r +5 "Cya later"  # Reboot in 5 minutes

shutdown -c                 # Cancel a shutdown or reboot

reboot                      # Reboot now
reboot -f                   # Force a reboot
```

Identifying Processes

```
top                  # List all processes interactively
htop                 # List all processes interactively
ps all               # List all processes
pidof foo            # Return the PID of all foo processes

CTRL+Z               # Suspend a process running in the foreground
bg                   # Resume a suspended process and run in the
background
fg                   # Bring the last background process to the foreground
fg 1                 # Bring the background process with the PID to the
foreground

sleep 30 &           # Sleep for 30 seconds and move the process into the
background
jobs                 # List all background jobs
jobs -p              # List all background jobs with their PID

lsof                 # List all open files and the process using them
lsof -itcp:4000      # Return the process listening on port 4000
```

Process Priority

Process priorities go from -20 (highest) to 19 (lowest).

```
nice -n -20 foo      # Change process priority by name
renice 20 PID        # Change process priority by PID
ps -o ni PID         # Return the process priority of PID
```

Killing Processes

```
CTRL+C                  # Kill a process running in the foreground
kill PID                # Shut down process by PID gracefully. Sends TERM
signal.
kill -9 PID             # Force shut down of process by PID. Sends SIGKILL
signal.
pkill foo               # Shut down process by name gracefully. Sends TERM
signal.
pkill -9 foo            # force shut down process by name. Sends SIGKILL
signal.
killall foo             # Kill all process with the specified name
gracefully.
```

Date & Time

```
date                    # Print the date and time
date --iso-8601         # Print the ISO8601 date
date --iso-8601=ns      # Print the ISO8601 date and time

time tree               # Time how long the tree command takes to execute
```

Scheduled Tasks

```
   *      *         *         *           *
Minute, Hour, Day of month, Month, Day of the week
```

```
crontab -l                  # List cron tab
crontab -e                  # Edit cron tab in Vim
crontab /path/crontab       # Load cron tab from a file
crontab -l > /path/crontab  # Save cron tab to a file

* * * * * foo               # Run foo every minute
*/15 * * * * foo            # Run foo every 15 minutes
0 * * * * foo               # Run foo every hour
15 6 * * * foo              # Run foo daily at 6:15 AM
44 4 * * 5 foo              # Run foo every Friday at 4:44 AM
0 0 1 * * foo               # Run foo at midnight on the first of the month
0 0 1 1 * foo               # Run foo at midnight on the first of the year

at -l                       # List scheduled tasks
at -c 1                     # Show task with ID 1
```

```
at -r 1                    # Remove task with ID 1
at now + 2 minutes         # Create a task in Vim to execute in 2 minutes
at 12:34 PM next month     # Create a task in Vim to execute at 12:34 PM
next month
at tomorrow                # Create a task in Vim to execute tomorrow
```

HTTP Requests

```
curl https://example.com                                  # Return response
body
curl -i|--include https://example.com                     # Include status code
and HTTP headers
curl -L|--location https://example.com                    # Follow redirects
curl -o|--remote-name foo.txt https://example.com         # Output to a text
file
curl -H|--header "User-Agent: Foo" https://example.com # Add a HTTP header
curl -X|--request POST -H "Content-Type: application/json" -d|--data
'{"foo":"bar"}' https://example.com # POST JSON
curl -X POST -H --data-urlencode foo="bar" http://example.com
# POST URL Form Encoded

wget https://example.com/file.txt .                           # Download a
file to the current directory
wget -O|--output-document foo.txt https://example.com/file.txt # Output to a
file with the specified name
```

Network Troubleshooting

```
ping example.com           # Send multiple ping requests using the ICMP
protocol
ping -c 10 -i 5 example.com # Make 10 attempts, 5 seconds apart

ip addr                    # List IP addresses on the system
ip route show              # Show IP addresses to router

netstat -i|--interfaces    # List all network interfaces and in/out usage
netstat -l|--listening     # List all open ports

traceroute example.com     # List all servers the network traffic goes
through

mtr -w|--report-wide example.com                                  #
Continually list all servers the network traffic goes through
mtr -r|--report -w|--report-wide -c|--report-cycles 100 example.com # Output
```

```
a report that lists network traffic 100 times

nmap 0.0.0.0                # Scan for the 1000 most common open ports on
localhost
nmap 0.0.0.0 -p1-65535      # Scan for open ports on localhost between 1 and
65535
nmap 192.168.4.3            # Scan for the 1000 most common open ports on a
remote IP address
nmap -sP 192.168.1.1/24     # Discover all machines on the network by
ping'ing them
```

DNS

```
host example.com            # Show the IPv4 and IPv6 addresses

dig example.com             # Show complete DNS information

cat /etc/resolv.conf        # resolv.conf lists nameservers
```

Hardware

```
lsusb                  # List USB devices
lspci                  # List PCI hardware
lshw                   # List all hardware
```

Terminal Multiplexers

Start multiple terminal sessions. Active sessions persist reboots. `tmux` is more modern than `screen`.

```
tmux             # Start a new session (CTRL-b + d to detach)
tmux ls          # List all sessions
tmux attach -t 0 # Reattach to a session

screen           # Start a new session (CTRL-a + d to detach)
screen -ls       # List all sessions
screen -R 31166  # Reattach to a session

exit             # Exit a session
```

Secure Shell Protocol (SSH)

```
ssh hostname                # Connect to hostname using your current user
name over the default SSH port 22
ssh -i foo.pem hostname     # Connect to hostname using the identity file
ssh user@hostname           # Connect to hostname using the user over the
default SSH port 22
ssh user@hostname -p 8765   # Connect to hostname using the user over a
custom port
ssh ssh://user@hostname:8765 # Connect to hostname using the user over a
custom port
```

Set default user and port in `~/.ssh/config`, so you can just enter the name next time:

```
$ cat ~/.ssh/config
Host name
  User foo
  Hostname 127.0.0.1
  Port 8765
$ ssh name
```

Secure Copy

```
scp foo.txt ubuntu@hostname:/home/ubuntu # Copy foo.txt into the specified
remote directory
```

Bash Profile

- bash - `.bashrc`
- zsh - `.zshrc`

```
# Always run ls after cd
function cd {
  builtin cd "$@" && ls
}

# Prompt user before overwriting any files
alias cp='cp --interactive'
alias mv='mv --interactive'
alias rm='rm --interactive'

# Always show disk usage in a human readable format
alias df='df -h'
alias du='du -h'
```

Bash Script

Variables

```bash
#!/bin/bash

foo=123                # Initialize variable foo with 123
declare -i foo=123     # Initialize an integer foo with 123
declare -r foo=123     # Initialize readonly variable foo with 123
echo $foo              # Print variable foo
echo ${foo}_'bar'      # Print variable foo followed by _bar
echo ${foo:-'default'} # Print variable foo if it exists otherwise print
default

export foo             # Make foo available to child processes
unset foo              # Make foo unavailable to child processes
```

Environment Variables

```bash
#!/bin/bash

env            # List all environment variables
echo $PATH     # Print PATH environment variable
export FOO=Bar # Set an environment variable
```

Functions

```bash
#!/bin/bash

greet() {
  local world = "World"
  echo "$1 $world"
  return "$1 $world"
}
greet "Hello"
greeting=$(greet "Hello")
```

Exit Codes

```bash
#!/bin/bash

exit 0   # Exit the script successfully
```

```
exit 1   # Exit the script unsuccessfully
echo $?  # Print the last exit code
```

Conditional Statements

Boolean Operators

- `$foo` - Is true
- `!$foo` - Is false

Numeric Operators

- `-eq` - Equals
- `-ne` - Not equals
- `-gt` - Greater than
- `-ge` - Greater than or equal to
- `-lt` - Less than
- `-le` - Less than or equal to
- `-e` foo.txt - Check file exists
- `-z` foo - Check if variable exists

String Operators

- `=` - Equals
- `==` - Equals
- `-z` - Is null
- `-n` - Is not null
- `<` - Is less than in ASCII alphabetical order
- `>` - Is greater than in ASCII alphabetical order

If Statements

```
#!/bin/bash

if [[$foo = 'bar']]; then
  echo 'one'
elif [[$foo = 'bar']] || [[$foo = 'baz']]; then
  echo 'two'
elif [[$foo = 'ban']] && [[$USER = 'bat']]; then
  echo 'three'
else
```

```
  echo 'four'
fi
```

Inline If Statements

```
#!/bin/bash

[[ $USER = 'rehan' ]] && echo 'yes' || echo 'no'
```

While Loops

```
#!/bin/bash

declare -i counter
counter=10
while [$counter -gt 2]; do
  echo The counter is $counter
  counter=counter-1
done
```

For Loops

```
#!/bin/bash

for i in {0..10..2}
  do
    echo "Index: $i"
  done

for filename in file1 file2 file3
  do
    echo "Content: " >> $filename
  done

for filename in *;
  do
    echo "Content: " >> $filename
  done
```

Case Statements

```
#!/bin/bash

echo "What's the weather like tomorrow?"
read weather

case $weather in
  sunny | warm ) echo "Nice weather: " $weather
  ;;
  cloudy | cool ) echo "Not bad weather: " $weather
  ;;
  rainy | cold ) echo "Terrible weather: " $weather
  ;;
  * ) echo "Don't understand"
  ;;
esac
```