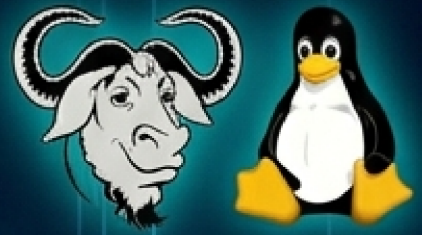


FREE EBOOK

LEARN GNU/Linux

one Commands a Day



ping

\$ sudo \$ ping mkdir

```
$ ls -l
$ sudo apt update
$ help command
```

root@linux:~#

```
$ sudo upn
$ sudo untinssat -12
$ sudo oph 172
$ sudo apt ito2n 10
$ 1187
```

PARTIAL RELEASE

CHECK [HTTPS://180LCODE.NET](https://180LCODE.NET) FOR UPDATES

LEARN GNU/LINUX

ONE COMMAND A DAY

A FREE E-BOOK BROUGHT TO YOU BY IBOLCODE

(C) DAVID MARTÍNEZ OLIVEIRA AKA "PICO", 2026
PUBLISHED BY IBOLCODE
LICENCE CC-BY

ANY PRODUCT AND COMPANY NAMES MENTIONED HEREIN MAY BE THE TRADEMARKS OF THEIR RESPECTIVE OWNERS. RATHER THAN USE A TRADEMARK SYMBOL WITH EVERY OCCURRENCE OF A TRADEMARKED NAME, WE ARE USING THE NAMES ONLY IN AN EDITORIAL FASHION AND TO THE BENEFIT OF THE TRADEMARK OWNER, WITH NO INTENTION OF INFRINGEMENT OF THE TRADEMARK.

THE INFORMATION IN THIS BOOK IS DISTRIBUTED ON AN AS IS BASIS, WITHOUT WARRANTY. WHILE EVERY PRECAUTION HAS BEEN TAKEN IN THE PREPARATION OF THIS WORK, NEITHER THE AUTHOR NOR NO STARCH PRESS, INC. SHALL HAVE ANY LIABILITY TO ANY PERSON OR ENTITY WITH RESPECT TO ANY LOSS OR DAMAGE CAUSED OR ALLEGED TO BE CAUSED DIRECTLY OR INDIRECTLY BY THE INFORMATION CONTAINED IN IT

INTRODUCTION

This short ebook is a compendium of everyday GNU/Linux commands that any user should know. The booklet includes a command per-page and it's intended to introduce you to one command every single day.

This is not the classical book about GNU/Linux that will give you a detailed description about each command. There is no need for that, the man pages in your system already gives you all the details and information you may ever need. The booklet is intended for you to discover one command every day, get a rough idea of what it does and then you can go deeper into how to use it.

The booklet is autogenerated from the Ibolcode's website section "The Command of The Day", where you'll find the same commands on line if you prefer. The website section will be run for a complete year until one command per day is shown. The booklet will be updated weekly to include the commands added every week and eventually it'll be completed.

You can check all this commands on-line at:

<https://ibolcode.net>

LS

LIST DIRECTORY CONTENTS

- `ls -l`
Long listing including details
- `ls -a`
List hidden files
- `ls -t`
List files sorted by modification time
- `ls -r bar`
List directory recursively
- `ls -S`
List files sorted by size
- `ls -Z`
Shows SeLinux context if any
- `ls -m`
Outputs items separated by commas
- `ls -i`
Outputs the associated inode



CHANGE CURRENT DIRECTORY

- `cd ..`
Change to parent dir
- `cd ~`
Change to home dir
- `cd ~user1`
Change to home dir of user user1
- `cd -`
Change to previous dir

PWD

PRINT CURRENT WORKING DIRECTORY

- `pwd`
Print current working directory
- `pwd -L`
Logical, do not follow symlinks
- `pwd -P`
Physical follows symlinks

MKDIR

CREATE DIRECTORIES

- `mkdir bar`
Create directory bar
- `mkdir bar1 bar2`
Creates directories bar1 and bar2
- `mkdir bar{1,2}`
Creates directories bar1 and bar2
- `mkdir -m 700 bar`
Create directory with specific permission
- `mkdir -p a/b/c`
Create a directory path
- `mkdir data-\$(date +%Y-%m-%d)`
Create folder named against current date
- `mkdir bar \&\& cd \$\_`
Create bar and move into it

RMDIR

REMOVE EMPTY DIRECTORIES

- `rmdir bar`
Remove directory bar
- `rmdir -p /a/b/c`
Removes directory and all ancestors
- `find . -type d -empty -exec rmdir {} \;`
Remove empty directories

RM

REMOVE FILES/DIRECTORIES

- `rm file`
Remove File
- `rm -r dir`
Remove File recursively
- `rm -rf dir`
Remove all Files recursively and bar (force)
- `rm -i dir/*`
Removes all files in bar interactively
- `rm -I dir/*`
Removes all files in bar only asking once before removing more than 3 files
- `rm -- -file`
Removes file starting with -

CP

COPY FILES/DIRECTORIES

- `cp file1 file2`
Copy foo1 into foo2
- `cp -r dir1 dir2`
Copy recursively bar1 into bar2
- `cp -a dir1 dir2`
Copy recursively bar1 into bar2 keeping file attributes
- `cp -u file dir/`
Only copies if source is newer than destination
- `cp -i file dir/`
Interactive overwrite protection
- `cp file{,.bak}`
Create a backup copy of file

mv

MOVE OR RENAME FILES

- `mv file1 file2`
Rename a file
- `mv -b file1 file2`
Makes a backup of bar if it exists
- `mv -i dir1 dir2`
Moves files and interactively ask to overwrite target if exists
- `mv -n file dir/`
Moves if destination doesn't exist
- `mv file dir/`
Move a file into a directory
- `for f in *.txt; do mv "$f" "${f%.txt}.md"; done`
Converts all .txt to .md (f%.txt strips extension)
- `find . -type f -mtime +7 -exec mv {} archive/ \;`
Move files older than 7 days
- `mv -- -file new\_file`
Move files starting with -

TOUCH

CREATE EMPTY FILE/UPDATE TIMESTAMP

- `touch file1 file2`
Create file1 and file2 files. Update timestamp if files exist
- `touch file{1..10}.txt`
Creates 10 files
- `touch dir/{a,b}`
Creates files dir/a and dir/b
- `touch -a file`
Change access time of foo
- `touch -m file`
Change modification time of foo
- `touch -r file1 file2`
Uses times from file1 instead of current time
- `touch -t 202601012310 file`
Changes foo timestamp to 1 Jan 2026 23 10
- `find . -exec touch {} \;`
Update timestamps recursively

STAT

DISPLAY FILE METADATA

- `stat file`
Displays foo metadata
- `stat -t file`
Terse format (all data in 1 line)
- `stat -f mount\_point`
Displays file system metadata
- `stat -c "%n %s %y" file`
Shows name, size and timestamp
- `stat -c "%i" file`
Get inode number
- `stat -c "%F" file`
Type of file

FILE

DETECT FILE TYPE

- `file file`
Shows type of file. Identifies scripts, binaries, compression,...
- `file -i file`
Shows MIME type of file

TREE

DISPLAY DIRECTORY TREE

- `tree`
Shows directory as a tree
- `tree dir`
Shows directory tree for path dir
- `tree -L 2`
Shows tree directory up to level 2
- `tree -d`
Shows only directories
- `tree -a`
Shows hidden files
- `tree -h --sort=size`
Shows file size human readable sorted by size
- `tree -H dir -o list.html`
Generates a browsable HTML file
- `tree -P "*.py" --prune`
Shows only python file and remove empty branches

REALPATH

RESOLVE ABSOLUTE PATH

- `realpath file`
Resolve absolute path to file
- `readpath ../dir/./other/../../final/file`
Resolve path for file
- `realpath --relative-to=dir file`
Resolves absolute path to file relative to dir
- `realpath --relative-to=. /path/file`
Generates relative path to file from current dir

READLINK

SHOW SYMLINK TARGET

- `readlink file`
Resolve link for foo. Shows original relative path
- `readlink -f file`
Shows complete path for link
- `readlink -e file`
Shows complete path for link resolving components recursively
- `find . -type l -exec readlink {} \;`
Find all symlinks and shows their targets

BASENAME

EXTRACT FILENAME FROM PATH

- `basename /path/file`
Extract file from path
- `basename -a file1 file2`
Apply to multiple files
- `basename file.ext ext`
Removes extension ext from file name
- `basename "$(readlink /proc/$$/exe)"`
Gives you the executable name of the current shell
- `find . -type f | xargs -n1 basename`
List only filenames (ignoring full paths)

DIRNAME

EXTRACT DIRECTORY PATH

- `dirname /path/file`
Returns /path
- `src\_dir="\$(dirname "$0")"`
In a script returns base directory

INSTALL

COPY FILES AND SET PERMISSIONS

- `install file1 file2`
Copy file
- `install -C file1 file2`
Only copy file if contents or metadata differs
- `install -m 755 file1 file2`
Copy file and set permissions
- `install -d /a/b/c/`
Creates folder path (like `mkdir -p`)

CHOWN

CHANGE FILE OWNER

- `chown user file`
Change owner of file
- `chown -R user dir`
Recursively change owner of directory and the files it contains
- `chown -f user file`
Silent. Suppress errors
- `chown --from=user.group file`
Changes ownership of foo if current owner and/or group matches the given values
- `chown -R -c user dir`
Change recursively ownership in bar only reporting when a change is made
- `chown user.group file`
Change owner and group of file

CHMOD

CHANGE FILE PERMISSIONS

- `chmod 755 file`
Sets file permissions to `rwxr-xr-x`
- `chmod -f o+x file`
Set execution permission for others and suppress possible errors
- `chmod +x script.sh`
Add execution permission to `script.sh`
- `chmod -R +x dir/*`
Add execution permissions recursively
- `chmod -R -c +w dir/*`
Add write permissions recursively and reports only the changes made

CHGRP

CHANGE FILE GROUP

- `chgrp grp file`
Change group for file to grp
- `chgrp grp -f file`
Change group for file to grp and suppress errors
- `chgrp -R grp *`
Changes group to grp recursively
- `chgrp grp -R -c dir`
Change recursively group of dir and reports only the changes made

UMASK

SET DEFAULT PERMISSIONS

- `umask`
Shows octal mask for permissions
- `umask -S`
Shows umask symbolically
- `umask 022`
Set umask to the indicated octal value

basename , 16

cd , 3

chgrp , 21

chmod , 20

chown , 19

cp , 8

dirname , 17

file , 12

install , 18

ls , 2

mkdir , 5

mv , 9

pwd , 4

readlink , 15

realpath , 14

rm , 7

rmdir , 6

stat , 11

touch , 10

tree , 13

umask , 22