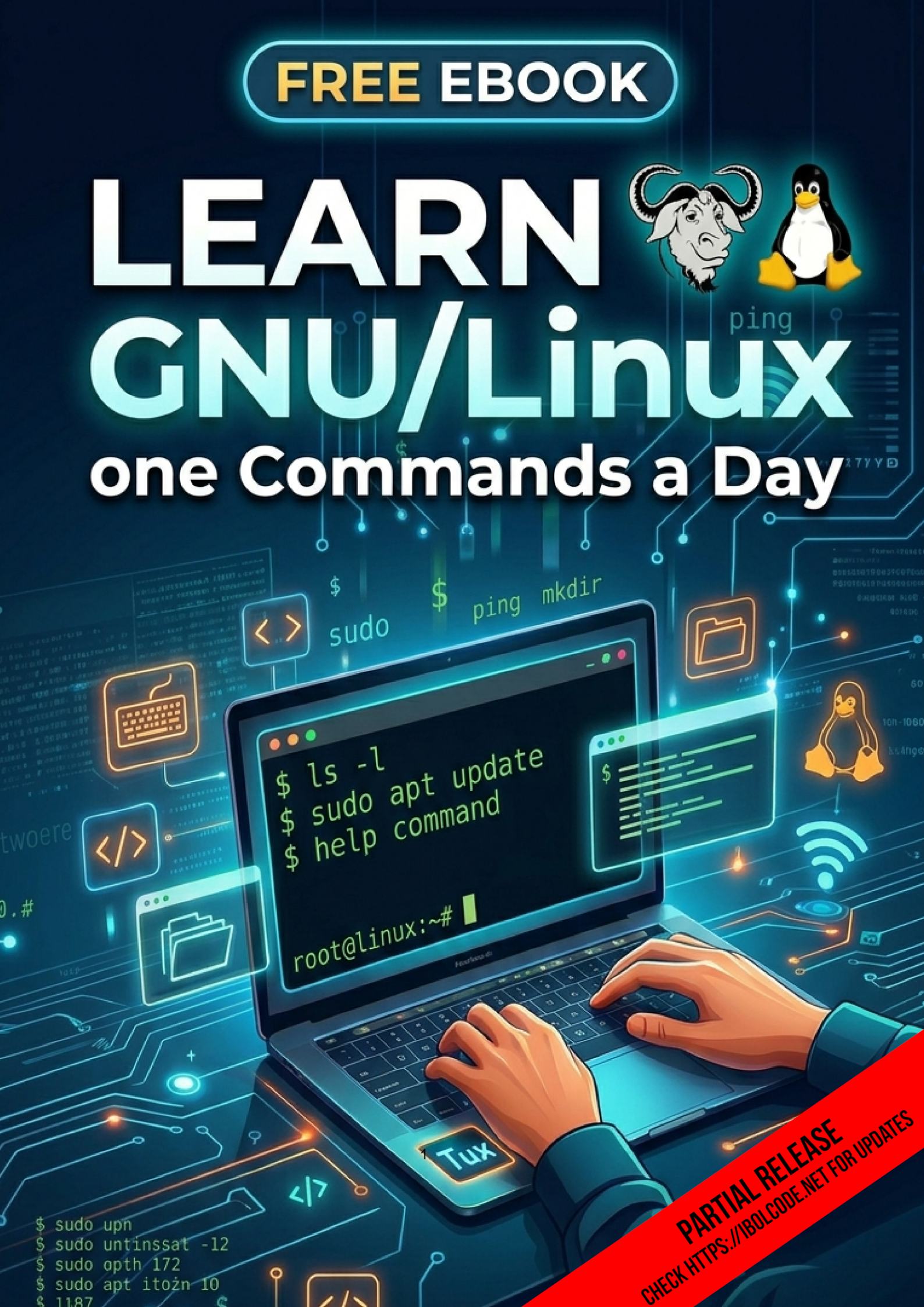
FREE EBOOK

LEARN GNU/Linux one Commands a Day



PARTIAL RELEASE
CHECK HTTPS://IBOLCODE.NET FOR UPDATES

Learn GNU/Linux

One commant 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

CD
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`
  Remmove 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 exits
- `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 free 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

LN

CREATE LINKS (HARD/SYMBOLIC)

- `ln file link`
  Create a hard link to foo
- `ln -b file link`
  If bar exists creates a backup
- `ln -i file link`
  If bar exists ask user
- `ln -f file link`
  Force creation of bar
- `ln -s file link`
  Create symbolic link instead of hard link
- `ln -s -r full\_path\_target link`
  Create symbolic link that stores a relative path to the target

DU
SHOW DISK USAGE

- `du -sh`
  Report disk usage summary in human readable format
- `du -h --max-depth=1`
  Only report disk usage one level below
- `du -ah / | sort -hr | head -20`
  Find the biggest files in the system
- `du -h --apparent-size file`
  Show apparent size instead of device usage

DF
SHOW FREE DISK SPACE

- `df -h`
  Shows free disk space in human readable
- `df -T`
  Shows type of file system
- `df dev|mount\_point`
  Shows free space for the associated disk
- `df -h | awk '\$5 > 80 {print}'`
  Shows filesystems over 80 % usage
- `df -i`
  List inode information instead of block usage
- `df foo`
  Tells you with mount point contains the file

sync
flush filesystem buffers

- `sync -d`
  Flush only file data, not unneeded metadata
- `sync -f file`
  Flush only file systems that contains the files

MOUNTPOINT

CHECK IF PATH IS MOUNTED

- `mountpoint /mnt`
  Indicates if /mnt is a mount point

SPLIT

SPLIT FILE INTO PIECES

- `split -l 100 file`
  Put 100 lines/records per output file (useful with logs)
- `split -b 1M file part\_`
  Put 1Mb per output file
- `split -b 200M file --filter'gzip > \$FILE.gz'`

  Compress chunks on the fly. Reconstruct with cat part*.gz >file
- `split -n l/4 data part\_`
  Splits file in 4 parts with same number of lines
- `split -n 5 file part\_`
  Split in 5 files
- `split -d -b 100 file part\_`
  Produces part_00 part_01 part_02... (decimal numbering)
- `split -x -b 100 file part\_`
  Produces part_00 ... part_0A, part_0B... (hex numbering)
- `split -f part\_ -n3 file '/PATTERN/'`
  Store files as part_001, part_002,...

CSPLIT

SPLIT FILE BY PATTERN

- `csplit file /pattern/`
  Split files at /pattern/
- `csplit file '/\^2026-/' '{*}'`
  Splits every time a line start with a date (2026-...)
- `csplit mailbox '/\^From /' '{*}'`
  Splits mailbox storing a message per file
- `objdump -j .text -d bin | csplit -f func\_ '/\^[0-9af]\\+ <.*>.\$/'`
  Extract all functions in a binary into its own file

SHRED
SECURELY OVERWRITE FILE

- `shred -u file`
  Deallocate and remove file after overwriting
- `shred -n 3 file`
  Overwrite file 3 times
- `shred -vzn3 /dev/sdX`
  Securely wipes a disk (zeroed + 3 passes)

WIPE

SECURE FILE DELETION FROM MAGNETIC MEDIA

- `wipe file`
  Secure delete file
- `wipe -r dir`
  Secure delete recursively
- `wipe -q file`
  Quick wipe (only 4 passes with random data)
- `wipe -kq /dev/hda1`
  Quick wipe /dev/hda1

SRM
SECURE REMOVE

- `srm file`
  Secure remove file
- `srm -r dir`
  Secure delete directory recursively
- `srm -s file`
  Simple delete (fast)
- `srm -m file`
  Medium. 7 pass (DoD 5220.22-M)
- `srm -z file`
  Gutmann. 38 passes.

CMP

COMPARE TWO FILES BYTE BY BYTE

- `cmp file1 file2`
  Compare file1 and file2
- `cmp -i 100K file1 file2`
  Skip initial 100Kb
- `cmp -n 123 file1 file2`
  Compare at most 123 bytes
- `cmp -s f1 f2 \&\& echo "Match"`
  Quick check

DIFF

COMPARE FILES LINE BY LINE

- `diff file1 file2`
  Shows difference between file1 and file2
- `diff -u file1 file2`
  Shows 3 lines of unified context per difference
- `diff -e file1 file2`
  Outputs and ed script
- `diff -y file1 file2`
  Outputs side by side
- `diff -t file1 file2`
  Expand tabs to spaces in output

PATCH

APPLY DIFF PATCH

- `patch < file.patch`
  Applies the patch in current folder. Apply bug fix
- `patch -R < changes.path`
  Tries to reverse the patch
- `patch -F 3 < patch`
  Allows approximate matching when code has shifted slightly
- `patch -b < patch`
  Makes backup files instead of overwriting
- `patch -p1 < patch`
  Removes initial / in path names so they are applied relative to current folder

RENAME

BATCH RENAME FILES

- `rename 's/ /\_/g' *`
  Changes spaces by underscores in file names
- `rename 's/(\\d+)/sprintf ("\%03d",\$1)/e'`
  Adds leading zeros to numbered files (file1.txt ->file001.txt)
- `rename 's/.txt/.md/' .txt`
  Rename .txt files to .md
- `rename 'y/A-Z/a-z/' *`
  Renames all files to lowercase
- `rename 's/ \\(\\d+\\)//' *`
  Clean-up browser downloads like (File (1).txt, File (2).txt,...)
- `rename 's/photo(\\d+)\_(\\d+)/image\_\$2\_\$1/' *`
  Renames files photo_1_2 ti image_2_1

FILEFRAG

SHOW FILE FRAGMENTATION

- `filefrag file`
  Reports file fragmentation for file file
- `filefrag -e file`
  Prints in extent format
- `filefrag -E file`
  Prints contents of ext4's extent status cache
- `filefrag -v file`
  Be verbose when checking file fragmentation

LSLOCKS
LIST FILE LOCKS

- `lslocks`
  List current file locks
- `lslocks -J`
  Outputs JSON format
- `lslocks -u`
  Do not truncate columns
- `lslocks -p PID`
  Only list locks associated to the indicated process

CAT

DISPLAY FILE CONTENTS

- `cat file`
  Shows file
- `cat file1 file2`
  Shows file1 and then file2
- `cat -E file`
  Shows a "$.ªt the end of each line
- `cat -n file`
  Numbers lines
- `cat -v file`
  Use ând M- notation (for binary files)

LESS

SCROLLABLE FILE VIEWER

- `less file`
  Pages file
- `color-output | less -R`
  Parses ANSI escape color sequences

MORE

BASIC PAGER

- `more file`
  Shows file in console stopping at page boundaries
- `more +N file`
  Starts showing file at line N
- `more +String file`
  Starts showing file at first occurrence of String
- `more -nN file`
  Pages every N lines

HEAD

SHOW FIRST LINES

- `head file`
  Shows the 10 first lines of a file
- `head -n 20 file`
  Shows the first 20 lines of a file
- `head -c 10 file`
  Shows the first 10 bytes from a file
- `head -z file`
  Use NUL as line delimiter instead of newline

TAIL

SHOW LAST LINES

- `tail file`
  Shows last 10 lines of file
- `tail -f file`
  Shows continuously new lines in file
- `tail -S S -f file`
  Shows end of file every S seconds
- `tail -n N file`
  Shows last N lines of file
- `tail -c N file`
  Shows last N bytes of file
- `tail -z file`
  Use NUL as line delimiter instead of newline

basename , 16

cat , 39
cd , 3
chgrp , 21
chmod , 20
chown , 19
cmp , 33
cp , 8
csplit , 29

df , 25
diff , 34
dirname , 17
du , 24

file , 12
filefrag , 37

head , 42

install , 18

less , 40
ln , 23
ls , 2
lslocks , 38

mkdir , 5
more , 41
mountpoint , 27
mv , 9

patch , 35
pwd , 4

readlink , 15
realpath , 14
rename , 36
rm , 7
rmdir , 6

shred , 30
split , 28
srm , 32
stat , 11
sync , 26

tail , 43
touch , 10
tree , 13

umask , 22

wipe , 31