

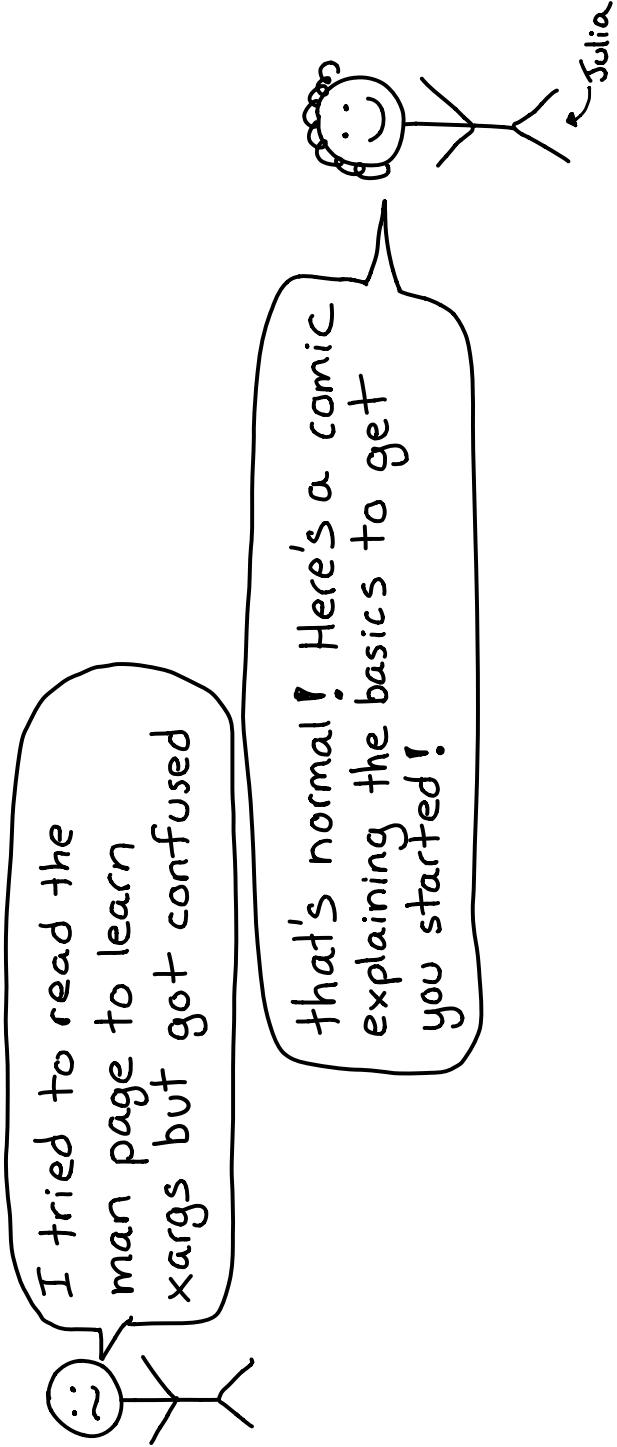
love this?
find more awesome zines at
→ wizardzines.com ←

Rite Size Line

By Julia Evans



This zine explains some of the most useful Unix command line tools in 1 page each.



Even if you've used the tool before, I might have a new trick or two for you ❤

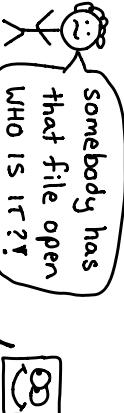
more useful tools

- make
- jq
- nohup
- disown
- cut/paste
- sponge
- xxd
- hexdump
- objdump
- strings
- screen
- tmux
- date
- entr
- seq
- join
- parallel:
 - GNU parallel
 - pigz/pixz
 - sort --parallel
- diff -U
- vipe
- imagemagick
- fish
- ranger
- chronic

lsof

lsof

stands for list open files



I can tell you!

-i

list open network sockets
(sockets are files!)

examples:

-i -n -P → -n & -P mean
"don't resolve
-i :8080 host names / ports"
-i TCP (also -Pni)
-i -s TCP:LISTEN

what lsof tells you
for each open file:

- pid
- file type (regular? directory?
FIFO? socket?)
- file descriptor (FD column)
- user
- filename/socket address

-P PID

list the files PID has
open

lsof /some /dir

list just the open files
in /some /dir

find deleted files

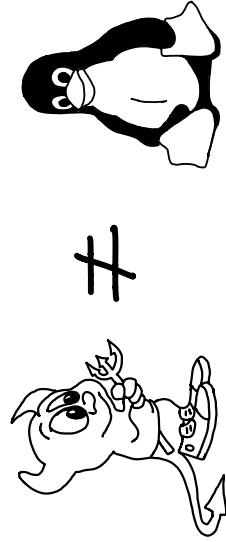
\$ lsof | grep deleted
will show you deleted files!
You can recover open deleted
files from
/proc/PID/fd/FD
process that opened the file

netstat

another way to list open
sockets on Linux is:
netstat -tunapl
↑
+tuna, please!
On Mac, netstat has
different args.

Table of contents

BSD#GNU.....4	bash tricks.....10-11	misc commands...17
grep.....5	disk usage.....12	head & tail.....18
find.....6	tar.....13	less.....19
xargs.....7	ps.....14	kill.....20
awk.....8	top.....15	cat.....21
sed.....9	sort & uniq.....16	lsof.....22



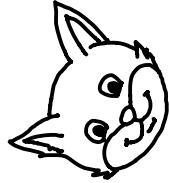
4

For almost all of these tools, there are at least 2 versions:

① The BSD version (on BSDs & Mac OS)

② The GNU version (on Linux)

All of the examples in this zine were tested on Linux. Some things (like sed -i) are different on Mac. Be careful when writing cross-platform scripts! You can install the GNU versions on Mac with brew install coreutils.



cat & friends

21

cat concatenates files

```
$ cat myfile.txt  
prints contents of myfile.txt  
$ cat *.txt  
prints all .txt files put together!
```

You can use cat as an EXTREMELY BASIC text editor:

- ① Run \$ cat > file.txt
- ② type the contents (don't make mistakes !)
- ③ press ctrl-d to finish

cat -n
prints out the file with line numbers!

- 1 Once upon a midnight...
- 2 Over many a quaint...
- 3 While I nodded, nearly

tee

```
'tee file.txt' will write its stdin to both stdout and file.txt  
$ echo "hi" | sudo tee -a x.txt  
will open x.txt as root ↗
```

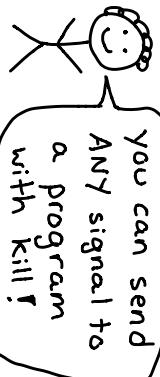
zcat

cats a gzipped file!
Actually just a 1-line shell script that runs gzip -cd, but easier to remember.

how to redirect to a file owned by root
\$ sudo echo "hi" >> x.txt
this will open x.txt as your user, not as root, so it fails!
\$ echo "hi" | sudo tee -a x.txt
will open x.txt as root ↗

Kill

kill doesn't just kill programs



You can send ANY signal to a program with kill!

\$ **kill -SIGNAL PID**

name or number

killall -SIGNAL NAME

signals all processes called NAME for example:

\$ killall firefox

useful flags:

- w wait for all signaled processes to die
- i ask before signalling

which signal kill sends	<u>name</u>	<u>num</u>
kill	=> SIGTERM	15
kill -q	=>	SIGKILL 9
kill -KILL	=>	SIGKILL 9
kill -HUP	=>	SIGHUP 1
kill -STOP	=>	SIGSTOP 19
1 HUP	2 INT	3 QUIT
5 TRAP	6 ABRT	7 BUS
9 KILL	10 USR1	11 SEGV
13 PIPE	14 ALRM	15 TERM
17 CHLD	18 CONT	19 STOP
21 TTIN	22 TTOU	23 URG
25 XFSZ	26 VTIME	27 PROF
29 POLL	30 PWR	28 WINCH
		31 SYS

pgrep

prints PIDs of matching running programs

pgrep fire matches firefox
firebird
NOT bash firefox.sh

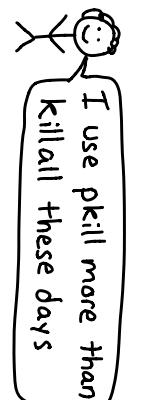
To search the whole command line (eg bash firefox.sh), use {pgrep -f}

kill -l	lists all signals
1 HUP	2 INT
3 QUIT	4 ILL
5 TRAP	6 ABRT
7 BUS	8 FPE
9 KILL	10 USR1
11 SEGV	12 USR2
13 PIPE	14 ALRM
15 TERM	16 STKFLT
17 CHLD	18 CONT
19 STOP	20 TSTP
21 TTIN	22 TTOU
23 URG	24 XCPU
25 XFSZ	26 VTIME
27 PROF	28 WINCH
29 POLL	30 PWR
31 SYS	

pkill

same as pgrep, but signals PIDs found. Example:

\$ pkill -f firefox



I use pkill more than killall these days

grep

5

 **-r** recursive! Search all the files in a directory.

 **-o** only print the matching part of the line instead of the whole line

 **-a** search binaries: treat binary data like it's text instead of ignoring it!

Grep alternatives
   ripgrep (better for searching code!)

 **-A** Show context for your search. For example:

\$ grep -A 3 cats
will show 3 lines of context after a match

 **-B**

Show context for your search. For example:

\$ grep -B 3 cats
will show 3 lines of context after a match

 **-C**

aka fgrep
don't treat the match string as a regex

find

6

find searches a directory for files

\$ find /tmp -type d, -print
 ↗ ↗
 directory which files action to do with the files
 ↗ here are my favourite find arguments!

-name/-iname
 ↗ case insensitive
 the filename! Example:
 -name '*.txt'

-path/-ipath
 ↗ search the full path!
 -path '/home/*/*.go'

-type TYPE
 ↗ f: regular file l: sym link
 ↗ d: directory + more!

-maxdepth NUM
 ↗ only descend NUM levels when searching a directory

-size 0
 ↗ find empty files!
 Useful to find files you created by accident

-exec COMMAND
 ↗ action: run COMMAND on every file found

-print0
 ↗ print null-separated filenames.
 Use with xargs -0!
 ↗ -delete
 ↗ action: delete all files found

locate
 ↗ The locate command searches a database of every file on your system.
 ↗ good: faster than find
 ↗ bad: can get out of date
 ↗ \$ sudo updatedb
 ↗ updates the database

less

less is a pager
 that means it lets you view (not edit) text files or piped-in text
 man uses your pager (usually less) to display man pages

many vim shortcuts work in less

/ search
 ↗ n/N next / prev match
 ↗ j/K down / up a line
 ↗ m/, mark/return to line beginning/end of file
 ↗ g/G (gg in vim)

less -r
 displays bash escape codes as colours
 try \$ ls --color | less -r with -r without -r ↗ ush
 ↗ a.+x+a.txt ↗ ESC C O M E S C ↗ Col; 31m a.txt ↗ g2 red, bold

q quit ↴ ↵
 ↗ runs a command when less starts
 ↗ less +F : follow updates
 ↗ less +G : start at end of file
 ↗ less +20% : start 20% into file
 ↗ less +/foo : search for 'foo' right away

F Press F to keep reading from the file as it's updated (like tail -f)
 ↗ press Ctrl+C to stop reading updates

V ↴ lowercase edit file in your \$EDITOR
 ↗ arrow keys, Home/End, PgUp, PgDn work in less

19

head & tail

head

shows you the first 10 lines of a file

if you pipe a program's output to head, the program will stop after printing 10 lines (it gets sent SIGPIPE)

-c NUM

show the first/last NUM bytes of the file

\$ head -c 1k

will show the first 1024 bytes

tail

tail shows the last 10 lines!

tail -f FILE will follow:

print any new lines added to the end of FILE. Super useful for log files!

tail --retry

keep trying to open file if it's inaccessible

tail --pid PID

stop when process PID stops running (with -f)

-n NUM

-n NUM (either head or tail) will change the # lines shown

NUM can also be negative. Example:

\$ head -n -5 file.txt

will print all lines except the last 5

tail --follow=name

Usually tail -f will follow a file descriptor.

tail --follow=name FILENAME

will keep following the same filename, even if the file descriptor changes

Xargs

7

how to replace "foo" with "bar" in all .txt files:

```
find . -name '*.txt' |  
xargs sed -i 's/foo/bar/g'
```

this is useful when you want to run the same command on a list of files!

- delete (xargs rm)
- combine (xargs cat)
- search (xargs grep)
- replace (xargs sed)

more useful xargs options:

-n 1 makes xargs run a separate process for each input
-P is the max number of parallel processes xargs will start

how to lint every Python file in your Git repo:

```
git ls-files | grep .py |  
xargs pep8
```

awk

8

awk is a tiny programming language for manipulating columns of data

I only know how to do 2 things with awk but it's still useful!

basic awk program structure:

```
BEGIN { ... }
CONDITION { ACTION }
CONDITION { ACTION }
END { ... }
```

do ACTION on lines matching CONDITION

extract a column of text with awk

```
awk -F, '{print $5}'
```

column separator quotes! 5th column

this is 99% of what I do with awk

so MANY unix commands print columns of text (ps! ls!)

so being able to get the column you want with awk is GREAT!

awk program example: sum the numbers in the 3rd column

```
{ s += $3;
END {print s;}
```

at the end, print the sum!

awk program example: print every line over 80 characters

```
length($0) > 80
```

condition
(there's an implicit {print} as the action)

misc commands

17

rlwrap

adds history & ctrl support to REPLs that don't already have them (rl stands for readline)

```
$ rlwrap python
```

watch

rerun a command every 2 seconds

file

figures out what kind of file (png? pdf?) a file is

ncdu

figure out what's using all your disk space

+s

add a timestamp in front of every input line

comm

find lines 2 sorted files have in common

diff

diff 2 files. Run with -U 8 for context.

xsel / xclip

copy/paste from system clipboard.
(pbcopy/pbpaste on Mac)

sort & uniq

sort sorts its input

\$ sort names.txt
the default sort is alphabetical.

uniq removes duplicates

a
b
b => a
a
c
c
notice there are still 2 'a's! uniq only uniques adjacent matching lines

sort -n
numeric sort
'sort' order
'sort-n' order

12	12
15000	=
48	=
96	=
6020	=
6020	=
15000	=

sort + **uniq** = ♥
Pipe something to 'sort | uniq' and you'll get a deduplicated list of lines! 'sort -u' does the same thing.
or sort uniq

b	b
b	b
b	b

sort -h: human sort
'sort -n' order
'sort-h' order

156	45 K
30M	30M
=	=
45 K	156
2006	2006

useful example:
\$ du -sh * | sort -h

uniq -c
counts each line it saw.

Recipe: get the top 10 most common lines in a file:
\$ sort cats.txt
| uniq -c
| sort -n
| tail -n 10

Sed

sed is most often used for replacing text in a file

\$ sed s/cat/dog/g file.txt

can be a regular expression

sed 5d
delete 5th line

sed /cat/d

delete lines matching /cat/

sed -n 5,30P
print lines 5-30

sed s+cat/+dog/+
use + as a regex delimiter
way easier than escaping / is like s/cat\\//dog\\//!

sed -n s/cat/dog/p
only print changed lines

sed G
double space a file
(good for long error lines)

sed '/cat/a dog'
append 'dog' after lines containing 'cat'

sed '17 panda'
insert "panda" on line 17

bash tricks

10

* **ctrl + r ***

search your history!

I use this **constantly** to rerun commands

* **magical braces ***

```
$ convert file.{jpg,png}  
expands to  
$ convert file.jpg file.png
```

{1..5} expands to 1 2 3 4 5

loops

```
for i in *.png  
do  
convert $i $i.jpg  
done
```

for loops:
easy & useful!

* **ctrl + r ***

```
$ convert file.{jpg,png}  
expands to  
$ convert file.jpg file.png
```

{1..5} expands to 1 2 3 4 5

loops

```
$( )  
gives the output of a  
command. Example:  
$ touch file-$(date -I)
```

creates a file named
file-2018-05-25

* **magical braces ***

```
$ convert file.{jpg,png}  
expands to  
$ convert file.jpg file.png
```

{1..5} expands to 1 2 3 4 5

loops

```
$( )  
gives the output of a  
command. Example:  
$ touch file-$(date -I)
```

creates a file named
file-2018-05-25

* **magical braces ***

```
$ convert file.{jpg,png}  
expands to  
$ convert file.jpg file.png
```

{1..5} expands to 1 2 3 4 5

loops

```
$( )  
gives the output of a  
command. Example:  
$ touch file-$(date -I)
```

creates a file named
file-2018-05-25

top

top

a live-updating summary of the top users of your system's resources

who's using all my memory?

top abv!

let's explain some numbers in top!

% CPU

350%? what?

this column is given as the % of a single core. If you have 4 cores, this can go up to 400%!

RES

this column is the "resident set size", aka how much RAM your process is using.

SHR is how much of the RES is shared with other processes

15

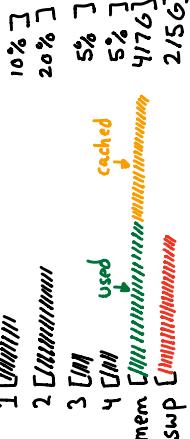
memory

4 numbers:

total / free / used / cached

One perhaps unexpected thing: total is **not** free + used!

total = free + used + cached filesystem cache



htop

a prettier & more interactive version of top *

PS

ps

ps shows which processes are running
I usually run ps like this:

```
$ ps aux
u means include username column
a+* together show all process
(ps -ef works too)
```

* process state *

Here's what the letters in ps's STATE column mean:

- R: running
- S/D: asleep
- Z: zombie
- I: multithreaded
- +: in the foreground

is for wide. ps auxWWWW will show all the command line args for each process

e

is for environment. ps auxe will show the environment vars!

```
$ ps -eo user, pid, wchan, cmd
```

f is for "forest" !. ps auxf will show you an ASCII art process tree !

pstree can display a process tree, too.

f

is for "forest" !. ps auxf will show you an ASCII art process tree !

ps has 3 different sets of command line arguments ↪

1. UNIX (1 dash)
2. BSD (no dash)
3. GNU (2 dashes)

You can write monstrosities like:

```
$ ps f -f
↑
forest(BSD) → full format (UNIX)
```

wchan

you can choose which columns to show with ps (ps -eo ...) One cool column is 'wchan', which tells you the name of the kernel function if the process is sleeping.

try it:

```
$ ps -eo user, pid, wchan, cmd
```

more bash tricks 11

cd -

changes to the directory you were last in

pushd & popd let you keep a stack

< ()

process substitution

treat process output like a file (no more temp files!)

Example:

```
$ diff <(ls) <(ls -a)
```

ctrl+z

suspends (SIGTSTP) the running program

fg

brings background/suspended program to the foreground

fc

"fix command"

open the last command you ran in an editor and then run the edited version

type

tells you if something is a builtin, program, or alias

try running: \$ type time

\$ type ping

\$ type pushd

bg

starts suspended program & backgrounds it (use after ctrl+z)

shellcheck

shell script linter! helps spot common mistakes.

disk usage

12

du

tells you how much disk space files / directories take up

- S summary: total size of all files in a directory
- h human readable sizes

df

tells you how much free space each partition has.

Filesystem	Size	Used	Avail	Use%	Mounted on
/dev/sda3	18G	6	2.5G	86%	/
udev	483M	4.0K	483M	1%	/dev
tmpfs	99M	1.4M	97M	2%	/run
/dev/sda4	1.67G	157G	9.9G	95%	/home

df -i

instead of % disk free, report how many **inodes** are used/ free on each partition

running out of inodes is **VERY ANNOYING**. You can't create new files!

ncdu

see what's using disk space in an interactive way

17.5 GiB [#####]	/music
3.2 GiB [##]	/photos
5.7 MiB []	/code
2.0 MiB []	file.pdf

iostat

get statistics about disk reads / writes

interval to report at
iostat 5
Device: kB_read/s kB_wrtn/s
sda 2190.21 652.87
sdb 6.00 0.00

tar

13

The .tar file format combines many files into one file.
a.txt
b.txt
dir/c.txt

.tar files aren't compressed by themselves. Usually you gzip them: .tar.gz or .tgz!

Usually when you use the 'tar' command, you'll run some incantation To unpack a tar.gz, use:
tar -xzf file.tar.gz



-t is for list lists the contents of a tar archive

-f is for file which tar file to create or unpack & more! see the man page

tar can compress / decompress

-Z gzip format (.gz)
-j bzip2 format (.bz2)
-T xz format (.xz)
more! see the man page

-X list contents of a .tar.bzz:

\$ tar -tf file.tar.bzz
create a .tar.gz
\$ tar -czf file.tar.gz dir/
files to go in the archive