fd versus find

Comparison of file search tools

Dr. Martin P.J. Zinser

LUG Frankfurt

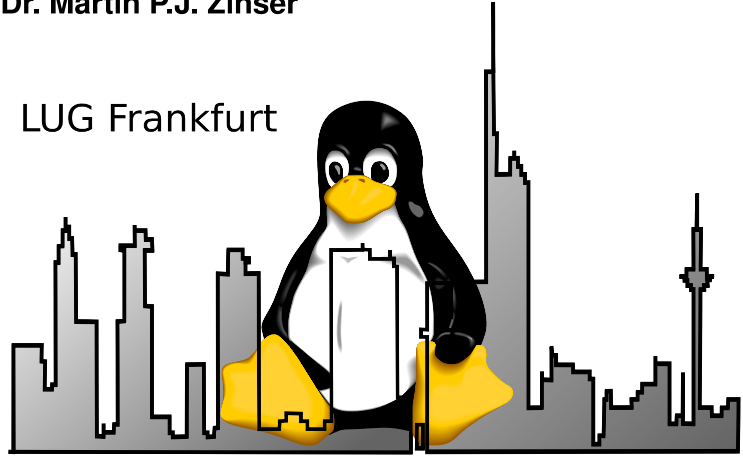


Table of contents I

Titel

Introduction

Use Cases - Comparison

Performance

Practical Examples

Conslusion

Coda

Table of Contents

Titel

Introduction

Use Cases - Comparison

Performance

Practical Examples

Conslusion

Coda

Purpose

find and fd scan a directory structure and return entries matching the provided selection criteria. The generated list may be processed further either directly in the command or via different tools (e.g. xargs).

find: `https://cgit.git.savannah.gnu.org/cgit/findutils.git/`
fd: `https://github.com/sharkdp/fd/`

LUG Frankfurt

Installation

find usually is part of the base OS installation.

fd likely needs to be added. Example openSUSE:

```
mzinser@cookie:~> cnf fd

The program 'fd' can be found in following packages:
  * fd [ path: /usr/bin/fd, repository: zypp (repo-oss) ]
  * fd [ path: /usr/bin/fd, repository: zypp (openSUSE:repo-oss) ]

Try installing with:
    sudo zypper install fd
```

Debian: Install package fd-find. Note: Command is fdfind

Who ist the impostor ?

	find		**fd**	
Verb	Long (4 char)	✗	Short (2 char)	✓
	Dictionary word	✗	Cryptic	✓
	Has Vowel	✗	No Vowels	✓
Syntax	-X –long -long	✗	-X –long	✓
Language	C	–	Rust	–
History	since 1987	✓	since 2017	✗

LUG Frankfurt

Table of Contents

Titel

Introduction

Use Cases - Comparison

Performance

Practical Examples

Conslusion

Coda

Simple Start

Plain fd and find output

```
find                 fd
.
./temp.txt           sub1/
./test.jpg           sub1/hurz.sh
./sub2               sub2/
./sub2/sheet.pdf     sub2/sheet.ods
./sub2/sheet.ods     sub2/sheet.pdf
./sub1               temp.txt
./sub1/hurz.sh       test.jpg
./test.jpg.1         test.jpg.1
```

Very similar, different sorting, fd omits .(/)

LUG Frankfurt

Default search scope

- find lists all files matching the search criteria
- find uses shell file globbing
- fd does
  - ... exclude hidden files by default (use fd -H/–hidden to change this)
  - ... honors .gitignore/.fdignore files.
    If you want results from them anyhow use fd -I/–no-ignore .
- fd uses regular expression patterns (change with -g/–glob)
- fd -E/–exclude inverses results (No equivalent in find)

Both do not follow symbolic links by default.
This can be changed with find -L, resp. fd -L :-)

LUG Frankfurt

What to search for? (I)

Names

- find -name/-iname <pattern>: Case sensitive (name), resp. case insensitive (iname) match of pattern against names.
- fd <pattern>: Uses *smart case*, i.e. all lowercase patterns are matched case insensitive, patterns with at least one uppercase character are matched case sensitive.
  Can be explicitly overridden with -s/–case-sensitive, resp. -i/–ignore-case

LUG Frankfurt

What to search for? (II)

File extensions

`fd -e <ext>` / `find -name ’*.<ext>’`

Multiple extensions (’or’):

```
fd -e <ext1> -e <ext2>
find -name ’*.<ext1>’ -or -name ’*.<ext2>’
```

Reg Exp!

`fd .<ext>` ≠ `fd -e <ext>`

Multiple path

`fd <pattern> <path1> <path2>.... <options>` Note: pattern is mandatory, but can be `.` (which matches everything)

`find <path1> <path2> ... <expression>`

What to search for? (III)

Directory transversal depth

- Maximum depth (Example: 2) fd -d 2 / find -maxdepth 2
- Minimum depth (Example: 3) fd –min-depth 3 / find -mindepth 3
- Exact depth (Example: 4) fd –exact-depth 4 / find -mindepth 4 -maxdepth 4

Don’t tarverse on to other file systems

fd –one-file-system / find -mount

What to search for? (IV)

Select by size

find -size / fd -S/–size

	find	**fd**
Prefix	+ $>$; ␣=; - $<$	+ $\geq$; ␣=; - $\leq$
Unit Type	Always binary	Default binary; suffix **i** Powers of ten
Units	b(blocks) c(bytes) w (words)	b(bytes)
	k M G (kilo/Mega/Gigabytes)	k m g t (kilo/Mega/Giga/Terabytes)

Note: find rounds up to the next unit, i.e. find -size -1M only matches empty files!

LUG Frankfurt

What to search for? (IV)

Filter by date

fd only supports filter by modification date (Timestamp or duration)
–changed-within/–change-newer-than/–newer and
–changed-before/–change-older-than

find supports a rich variety of options

	Access time	Status changed	Modified
Minutes	-amin n	-cmin n	-mmin n
Days	-atime n	-ctime n	-mtime n
Units	-anewer reference	-cnewer reference	-newer reference

. . . plus combinations find -newerXY reference: X of file newer than Y of reference

Examples:

- Find files modified in the last week:
  ```
  find . -type f -mtime -7
  fd . –type f –newer 7d
  ```
- Find files older than a year:
  ```
  find . -type f -mtime +365
  fd . –type f –changed-before 1year
  ```

What to do? (I)

(My) Best practise

Always check the resulting list of files before using fd/find to change/delete something!

LUG Frankfurt

What to do? (II)

Deleting files

- find <conditions> -delete
- fd <conditions> -x rm {}

Note: The -delete action is a special case in find. Could be also written as find <conditions> -exec rm {} ␣\;

What to do? (II)

Acting on files
Each entry individually

- find <conditions> -exec command {} ␣\;
- fd <conditions> -x/–exec <command>

Bulk execute for as many results as possible

- find <conditions> -exec command {} ␣+
- fd <conditions> -X/–exec-batch <command>

Note: If the <command> is suitable, bulk execution can speed up the overall processing considerably.

What to do? (III)

Placeholders

	Description	find	fd	Example
{}	Path	+	+	./sample/temp.txt
{/}	Basename	-	+	temp.txt
{//}	Parent Directory	-	+	./sample
{.}	Path without Extension	-	+	sample/temp
{/.}	Basename without Extension	-	+	temp

Table of Contents

Titel

Introduction

Use Cases - Comparison

Performance

Practical Examples

Conslusion

Coda

fd vs find shoot out

A major claim of fd is considerably improved performance compared to find.

- Running fair and reproducible tests is a non-trivial exercise
- In the end your specific environment\application is the yard stick to use

Rough indication: 11TB local USB drive `find . -name \*Star*` vs `time fd Star` using time command (Note: Caches had been pre-filled via previous run)

	fd	find
real	0m0.250s	0m1.275s
user	0m0.510s	0m0.843s
sys	0m0.380s	0m0.407s

For more detailed benchmarking on your own:

https://github.com/sharkdp/hyperfine

Notes (Courtesy to Jens Kühnel and Manfred Lotz):

Fedora 42 - SSD Samsung 990 PRO 4 TB; find/fd $HOME &>/dev/null

Setup cold cache: echo 3>/proc/sys/vm/drop_caches

	Cold cache		Hot cache	
	fd	find	fd	find
real	0m1.564s	4m2.658s	0m0.743s	2m28.711s
user	0m2.895s	0m18.413s	0m1.936s	0m12.413s
sys	0m4.445s	1m38.591s	0m2.035s	1m0.648s

```
% time                  seconds      usecs/call    calls      errors                     syscall
----                    ------       -------       -----      -----                      ---------
strace -c fd . $HOME
99,72                   4,471578     2235789       2                                     futex
0,21                    0,009457     591           16                                    mprotect
strace -c find $HOME
32,31                   38,733347    6             5838405                               write
15,79                   18,926337    3             4854353                               fcntl
14,87                   17,830930    9             1966426                               getdents64
% time                  seconds      usecs/call    calls      function
----                    ------       -------       -----      -----
ltrace -c fd . $HOME
86.21                   0.001313     1313          1          exit_group
13.79                   0.000210     210           1          __cxa_finalize
ltrace -c find $HOME
17.52                   31.409629    45            683576     __ctype_get_mb_cur_max
12.62                   22.623138    66            341770     __fprintf_chk
11.40                   20.440622    48            424275     readdir
10.17                   18.238516    45            396838     malloc
```

Notes (continued):

- The comparison is not entirely apples to apples, as fd ignores some files by default (see “Default Search Scope“). fd -H -I resp fd -u makes the behaviour more similar to the one of find .

Test setup 2: Aurora scanning a Samsung Protable SSD T5

	Cold cache		Hot cache	
	fd	find	fd	find
real	0m10.802s	0m18.865s	0m0.692s	0m3.470ss
user	0m4.307s	0m0.790s	0m2.379s	0m0.622s
sys	0m11.116s	0m6.072s	0m4.906s	0m2.821s

Notes (continued):

And with hyperfine:

```
hyperfine -i -warmup 3 'fd -H -I . $HOME' 'find $HOME'
Benchmark 1: fd -H -I . $HOME
Time (mean ± σ): 573.1 ms ± 5.8 ms [User: 2100.9 ms, System: 4168.3 ms]
Range (min ... max): 566.8 ms ... 586.1 ms 10 runs
Benchmark 2: find $HOME
Time (mean ± σ): 2.952 s ± 0.016 s [User: 0.540 s, System: 2.390 s]
Range (min ... max): 2.924 s ... 2.986 s 10 runs
Warning: Ignoring non-zero exit code.
Summary
fd -H -I . $HOME
5.15 ± 0.06 times faster than find $HOME
```

Notes^2: You might have noticed that in the output of time and hyperfine “real“ time in some cases is smaller than the sum of “user“ and “sys“. This can be the case if the system under test has more than one core (very likely nowadays) and the application is using multiple parallel threads.

Table of Contents

Titel

Introduction

Use Cases - Comparison

Performance

Practical Examples

Conslusion

Coda

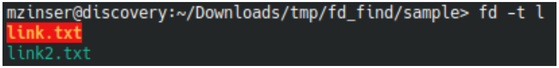
Broken Link detection

Purpose: Find broken symbolic links

```
mzinser@discovery:~> ls -l *.txt
lrwxrwxrwx 1 1024 users 8 Sep 27 14:15 link2.txt -> temp.txt
lrwxrwxrwx 1 1024 users 8 Sep 27 14:15 link.txt -> hurz.txt
-rw-r--r-- 1 1024 users 0 Aug 31 20:52 temp.txt
```

find <path> -xtype l

fd -t l



```
mzinser@discovery:~/Downloads/tmp/fd_find/sample> fd -t l
link.txt
link2.txt
```

https://github.com/sharkdp/fd/discussions/1298

Courtesy of Manfred Lotz

Files by date

Purpose: Show files in directory tree sorted by mtime

```
find -type f -printf ’%TF %.8TT %-8.8u %P\n’| sort -k1,2 | uniq –group -w10
```

```
fd –type f -X ls -ltG –time-style=long-iso | sort -k5,6 | \
gawk ’{print 5,6, 3,7}’ | uniq –group -w10
```

find
```
2024-12-01 18:47:10 mzinser sleep/salad_files/show_009.png

2024-12-23 10:34:02 mzinser gabriela_add_label.odt

2025-01-05 18:44:52 mzinser ev7.ods

2025-03-08 18:04:42 mzinser pgo_leader.ods

2025-05-24 20:21:12 mzinser Time_who.pdf
2025-05-24 20:22:06 mzinser Check.pdf
```

fd
```
2024-12-01 18:47 mzinser ./sleep/salad_files/show_009.png

2024-12-23 10:34 mzinser ./gabriela_add_label.odt

2025-01-05 18:44 mzinser ./ev7.ods

2025-03-08 18:04 mzinser ./pgo_leader.ods

2025-05-24 20:21 mzinser ./Time_who.pdf
2025-05-24 20:22 mzinser ./Check.pdf
```

Courtesy of Thomas Sattler

Find entries in /sys and /proc

Purpose: Show file system entries in /sys and /proc containing a name pattern

```
sysfind () {
      local pp=""
      if [ "$#" -eq 0 ]
      then
        echo -e ’searches /sys and /proc for given keyword.\nOptions:  -f  -  files only (no dirs)’
        return
      elif [ "-f" = "$1" ]
      then
        pp="-type f"
        shift
      fi
      for P in /sys /proc
      do
        find $P $pp -name "*$1*" 2>/dev/null
      done
}
```

Find entries in /sys and /proc (Continued)

To use fd: pp=“–type f" and fd "*$1*" $pp $P

Example:

```
$ sysfind -f snd
/sys/kernel/btf/snd_hda_codec_generic
/sys/kernel/btf/snd_seq_device
...
/proc/asound/oss/sndsta
```

Courtesy of Bernhard Gabler

Table of Contents

Titel

Introduction

Use Cases - Comparison

Performance

Practical Examples

Conslusion

Coda

LUG Frankfurt

Summary

- fd follows standard Linux conventions for options and arguments. The learning curve is much less steep than for find
- fd is indeed faster than find, but the exact speed up factor depends on the actual use case
- find covers some file attributes and use cases, that are not in scope of fd

LUG Frankfurt

Table of Contents

Titel

Introduction

Use Cases - Comparison

Performance

Practical Examples

Conslusion

Coda

LUG Frankfurt

(p|m)locate

locate is another tool to find entries in the file system, but with a different approach than fd/find. While fd/find traverse the file system to find matches, locate consults a database of all filesystem entries generated by updatedb. Typically the contents of the database is refreshed on a regular basis either using a cron job (old school :-)) or a systemd.timer .

- Pros: Fast and low resource consumption, since locate only consults the pre-existing database.
- Cons: Not "realtime", i.e. files created/deleted since the last database update are not reported correctly. ("deleted"can be fixed using the -e/–existing option at the price of performance degradation (since the filesystem needs to be queried for each hit).
- Cons: Does only allow to select on the name of the file, no other attributes

- Personal opinion: I like locate a lot for “static“ files like (kernel) sources.

So Long,
And Thanks
for All
the Fish