TUTORIALS

How to Use find -printf in Linux?

By Hank Cheah • Posted on August 25, 2021 • Updated on August 27, 2021

In this tutorial, you will learn how to use the GNU/Linux find utility's printf option.

Among the functionalities that make this file searching utility so powerful is its printf option.

The printf option allows you to override the default output which is just filenames and their relative paths.

Using -printf, you can instruct `find` to show permission, owner, modification time, ..., and the list goes on.

If you're interested in finding out how to make find shows more than just the filenames, this article is for you.

In the following sections, I will go through most of the directives supported by -printf and show you the actual outputs I generated on my local machine using these directives.

As mentioned, we'll be using the GNU find which is commonly found on GNU/Linux distros like Ubuntu, Debian, Centos, etc.

Find -printf Examples

The -printf option is used in the following way:-

```
$ find -printf '<input>'
```

Where <input> can include any user-provided strings, escapes(\), and directives(%).

Both single quotes(') and double quotes(") are supported.

Quick Links

Escapes

In my opinion, these two are the most important escapes to remember:-

\n – Newline

\t – Horizontal tab

They're pretty self-explanatory, so I won't be providing any examples here. But you will see them in the next section.

Directives

Using directives, you can instruct find to print almost any information pertaining to the found files.

For example, file path, access time, modified time, file size, etc.

Below I'll provide find command examples that make use of most of the directives that are supported by find. I'll explain them using simple words, as well as show you the outputs generated by these commands on my WSL2 Ubuntu terminal.

However, for more precise and concise explanations, please refer to the GNU manual for find.

Name Directives

%p – files with paths relative to the starting point of the search

```
$ find . -printf "%p\n"
.
./wip-2-tables.sql
./subdir
./subdir/demo.txt
./month.sql
```

This is equivalent to the default find output.

```
$ find .
.
./wip-2-tables.sql
./subdir
```

```
./subdir/demo.txt
./month.sql
```

%P – like %p, but the starting point is removed

```
$ find . -printf "%P\n"


wip-2-tables.sql
subdir
subdir/demo.txt
month.sql
```

In the example above, the starting point which is the current directory (.) is not shown in the search result.

%f – just filenames (basename), no paths

```
$ find . -printf "%f\n"
.
wip-2-tables.sql
subdir
demo.txt
month.sql
```

%f prints just the last part of the file path, which is the basename. This applies to both files and directories. In the example above, subdir is a directory.

%H – the starting point for find as specified by the user.

For example, if two files are found under subdir/, the command would print subdir/ twice:-

```
$ find subdir/ -printf "%H\n"
subdir/
subdir/
```

Here's another example. The command found five results so it prints the user-specified search starting point (.) five times.

```
$ find . -printf "%H\n"
.
.
.
.
.
```

%h – this shows the matching files' relative file paths, without the filename.

For example, say we have these files in the directory:-

```
$ find .
.
./date
./date/wip-2-tables.sql
./date/subdir
./date/subdir/demo.txt
./date/month.sql
```

Using the %h directive would give us the following results:-

```
$ find . -printf "%h\n"
.
.
./date
./date
./date/subdir
./date
```

Ownership Directives

The following printf directives deal with file ownership and file permission.

The first few directives are straightforward, so I won't be providing any examples of them.

%g – print the file’s group name. If the file has no group name, print the group ID instead.

%G – print the file’s group ID.

%u – show the file’s owner or owner ID if the owner has no name.

%U – the ID of the file’s owner

%m – the file permission. e.g. 777, 640, 000

```
$ find . -printf "%m\t%p\n"
755     .
644     ./wip-2-tables.sql
755     ./subdir
644     ./subdir/demo.txt
644     ./month.sql
```

In the preceding example, %m show the file permission, \t adds a horizontal tab, and %p prints the file paths.

%M – file type and file permission in symbolic form

Example:-

```
$ find . -printf "%M\t%p\n"
drwxr-xr-x      .
-rw-r--r--      ./wip-2-tables.sql
drwxr-xr-x      ./subdir
-rw-r--r--      ./subdir/demo.txt
-rw-r--r--      ./month.sql
```

Size Directives

%k – file size in KB

```
$ find . -printf "%k\t%p\n"
4       .
```

```
4       ./wip-2-tables.sql
4       ./subdir
0       ./subdir/demo.txt
4       ./month.sql
```

%b – file size in 512-byte blocks.

```
$ find . -printf "%b\t%p\n"
8       .
8       ./wip-2-tables.sql
8       ./subdir
0       ./subdir/demo.txt
8       ./month.sql
```

%s – file size in bytes.

```
$ find . -printf "%s\t%p\n"
4096    .
2395    ./wip-2-tables.sql
4096    ./subdir
0       ./subdir/demo.txt
1407    ./month.sql
```

Location Directives

%d – file’s depth in the directory tree.

```
$ find . -printf "%d\t%p\n"
0       .
1       ./date
2       ./date/wip-2-tables.sql
2       ./date/subdir
3       ./date/subdir/demo.txt
2       ./date/month.sql
```

%D – show the device number of the device where the file resides.

```
$ find . -printf "%D\t%p\n"
2064    .
2064    ./date
2064    ./date/wip-2-tables.sql
2064    ./date/subdir
2064    ./date/subdir/demo.txt
2064    ./date/month.sql
2064    ./date/solutions
```

%F – show the type of filesystem

E.g. ext2, ext3, ext4, etc

```
$ find . -printf "%F\t%p\n"
ext4    .
ext4    ./date
ext4    ./date/wip-2-tables.sql
ext4    ./date/subdir
ext4    ./date/subdir/demo.txt
ext4    ./date/month.sql
ext4    ./date/solutions
```

Location: Link Directives

Officially grouped under location directives, I found that the following directives mainly focus on links, both symbolic links and hard links.

For this section, we'll be using the following file structure:-

```
$ find . -printf "%p\n"
.
./src
./src/src-file-2
./src/src-file-1
./dst
./dst/tgt-file-2
./dst/tgt-file-1
```

src/ only contains regular files, there are no symbolic links. dst/ contains two files:-

- tgt-file-1 is a hard link of src/src-file-1
- tgt-file-2 is a symbolic link of src/src-file-2

%l – this shows the object (origin) of the symbolic link.

The result is empty if the file is not a symbolic link.

```
$ find . -printf "%l\t%p\n"
        .
        ./src
        ./src/src-file-2
        ./src/src-file-1
        ./dst
/home/hgcheah/linux/find/links/src/src-file-2   ./dst/tgt-file-2
        ./dst/tgt-file-1
```

Since ./dst/tgt-file-2 is the only symbolic link, %l returns its source which is /home/hgcheah/linux/find/links/src/src-file-2. For other files, which include hard links, %l return empty strings.

Observe that the output looks messed up. This is because the first column (%l) has variable length.

See the Formatting Tips section to see how to set minimum field width.

%i – the file's inode number.

```
$ find . -printf "%i\t%p\n"
103100  .
103104  ./src
103107  ./src/src-file-2
103106  ./src/src-file-1
103105  ./dst
103108  ./dst/tgt-file-2
103106  ./dst/tgt-file-1
```

%n – number of hard links to the file's inode

```
$ find . -printf "%n\t%p\n"
4       .
2       ./src
1       ./src/src-file-2
2       ./src/src-file-1
2       ./dst
1       ./dst/tgt-file-2
2       ./dst/tgt-file-1
```

In the example above, ./dst/tgt-file-1 is a hard link of ./src/src-file-1. This means their shared inode is linked to two files and the %n directive returns 2 as the result.

%y – file type

f – file, d – directory, l – symbolic link, b – block device files, s – socket file, etc.

```
$ find . -printf "%y\t%p\n"
d       .
d       ./src
f       ./src/src-file-2
f       ./src/src-file-1
d       ./dst
l       ./dst/tgt-file-2
f       ./dst/tgt-file-1
```

%Y – a more inquisitive version of %y

Where %y just prints the file type as presented, %Y does more investigative work into symbolic links. Namely, it will try to trace the links and report back if there's any problem.

For this example, let's focus on dst/tgt-file-2, which is a symbolic link of src/src-file-2.

Using %y from the previous section, this is the find result:-

```
$ find . -name tgt-file-2 -printf "%y\t%p\n"
l       ./dst/tgt-file-2
```

%y returns the letter l (representing symbolic link) which is the expected outcome.

On the other hand, this is the %Y result:-

```
$ find . -name tgt-file-2 -printf "%Y\t%p\n"
f       ./dst/tgt-file-2
```

As you can see, %Y has resolved the symbolic link and found that it is linked to a file (as indicated by the letter f). Similarly, if the symbolic link was pointing to a directory, the letter d would have been shown.

%Y can also be used to show other problems a symbolic link may be having. For example, if I delete the source of the symbolic link (src/src-file-2), orphaning tgt-file-2 in the process:-

```
$ find . -name tgt-file-2 -printf "%Y\t%p\n"
N       ./dst/tgt-file-2
```

%Y shows N which means that the symbolic link is broken.

Here's a list of possible return values for %Y:-

- file type: f, d, b, s, etc.
- N – broken link
- L – loop found
- ? – error

Time Directives

Using printf time directives, the user can show the files' time-related properties such as last access time last modified time, and last status change time.

These time directives can be roughly split into two groups. Group 1 uses C ctime format. Group 2 allows you to specify your own time format (but it has to be in a format that is supported by your local strftime).

Below is the list of time directives supported by find as well as some sample outputs.

Group 1 – C ctime format

%a – last access time in C ctime format.

%c – last status change time in C ctime format.

%t – last modification time in C ctime format.

Here's an example using %a. The output is similar for %c and %t.

```
$ find . -printf "%a\t%p\n"
Tue Aug 24 22:59:30.0500000000 2021     .
Wed Aug 25 08:51:24.8500000000 2021     ./date
Tue Aug 24 22:59:26.9000000000 2021     ./date/wip-2-tables.sql
Tue Aug 24 22:59:41.0400000000 2021     ./date/subdir
Tue Aug 24 22:59:26.9000000000 2021     ./date/subdir/demo.txt
Tue Aug 24 22:59:26.9000000000 2021     ./date/month.sql
Wed Aug 25 08:51:24.8500000000 2021     ./date/solutions
```

Group 2 – Custom format

%Ak – last access time in format specified by k.

%Ck – last status change time in format specified by k.

%Tk – ast modification time in format specified by k.

The time format k is explained in the following section.

Time Formats

As shown in the previous section, some time directives (%T, %C, %A) support user-specified formats.

These time formats are in conjunction with these directives in the following way:-

```
find . -printf "%T<format>\n"
```

In the examples below, I'll be using %T<format> which prints the modification time in various formats.

find uses format supported by strftime, so refer to strftime for the complete list of supported time formats. This implies that depending on your strftime version, the following examples may or may not be supported by your system.

Quick tip: use `man strftime` to see the list of supported time formats on your system.

@ – seconds since the Unix Epoch

```
$ find . -printf "%T@ %p\n"
1629858460.3700000000 .
1629817166.9000000000 ./demo.txt
1629858460.3700000000 ./demo2.txt
```

The output is sort-friendly, but unfortunately it's not very readable. %T+ (near the bottom of this section) provides a more readable output that is sortable.

a – abbreviated weekday name (Sun to Sat)

```
$ find . -printf "%Ta    %p\n"
Wed    .
Tue    ./demo.txt
Wed    ./demo2.txt
```

A – full weekday name (Sunday to Saturday)

```
$ find . -printf "%TA    %p\n"
Wednesday    .
Tuesday    ./demo.txt
Wednesday    ./demo2.txt
```

b or h – abbreviated month name (Jan to Dec)

```
$ find . -printf "%Tb    %p\n"
Aug    .
Aug    ./demo.txt
Aug    ./demo2.txt

$ find . -printf "%Th    %p\n"
Aug    .
Aug    ./demo.txt
Aug    ./demo2.txt
```

B – full month name (January..December)

```
$ find . -printf "%TB    %p\n"
August    .
August    ./demo.txt
August    ./demo2.txt
```

m – month (01..12)

```
$ find . -printf "%Tm    %p\n"
08    .
08    ./demo.txt
08    ./demo2.txt
```

d – day of month (01..31)

```
$ find . -printf "%Td    %p\n"
25    .
24    ./demo.txt
25    ./demo2.txt
```

w – day of week (0..6)

0 – Sunday, 1 – Monday, ..., 6 – Saturday

```
$ find . -printf "%Tw    %p\n"
3    .
2    ./demo.txt
3    ./demo2.txt
```

j – day of year (001..366)

```
$ find . -printf "%Tj    %p\n"
237    .
236    ./demo.txt
237    ./demo2.txt
```

Adding in %Ta to show the weekday.

```
$ find . -printf "%Tj    %Ta    %p\n"
237    Wed    .
236    Tue    ./demo.txt
237    Wed    ./demo2.txt
```

U – week number of year with Sunday as first day of week (00 to 53)

```
$ find . -printf "%TU    %p\n"
34    .
34    ./demo.txt
34    ./demo2.txt
```

W – week number of year with Monday as first day of week (00 to 53)

```
$ find . -printf "%TW    %p\n"
34    .
34    ./demo.txt
34    ./demo2.txt
```

Y – year (1970…)

```
$ find . -printf "%TY    %p\n"
2021    .
2021    ./demo.txt
2021    ./demo2.txt
```

y – last two digits of year (00 to 99)

```
$ find . -printf "%Ty    %p\n"
21    .
21    ./demo.txt
21    ./demo2.txt
```

r – time in 12-hour format (hh:mm:ss [AP]M)

```
$ find . -printf "%Tr    %p\n"
10:27:40 AM    .
10:59:26 PM    ./demo.txt
10:27:40 AM    ./demo2.txt
```

T – time in 24-hour format (hh:mm:ss.xxxxxxxxxx)

```
$ find . -printf "%TT    %p\n"
10:27:40.3700000000    .
22:59:26.9000000000    ./demo.txt
10:27:40.3700000000    ./demo2.txt
```

X – locale time (hh:mm:ss.xxxxxxxxxx)

```
$ find . -printf "%TX    %p\n"
10:27:40.3700000000    .
22:59:26.9000000000    ./demo.txt
10:27:40.3700000000    ./demo2.txt
```

c – locale time in ctime format

```
$ find . -printf "%Tc    %p\n"
Wed Aug 25 10:27:40 2021    .
Tue Aug 24 22:59:26 2021    ./demo.txt
Wed Aug 25 10:27:40 2021    ./demo2.txt
```

D – date (mm/dd/yy)

```
$ find . -printf "%TD    %p\n"
08/25/21    .
08/24/21    ./demo.txt
08/25/21    ./demo2.txt
```

F – date (yyyy-mm-dd)

```
$ find . -printf "%TF    %p\n"
2021-08-25    .
2021-08-24    ./demo.txt
2021-08-25    ./demo2.txt
```

x – locale date (mm/dd/yy)

```
$ find . -printf "%Tx    %p\n"
08/25/21    .
08/24/21    ./demo.txt
08/25/21    ./demo2.txt
```

R – hour and minute in 24 hour format (HH:MM)

```
$ find . -printf "%TR | %p\n" | sort -r
22:59 | ./wip-2-tables.sql
```

+ both date and time, separated by ’+’

Show both date and time down to the fractional seconds. Format: yyyy-mm-dd+hh:mm:ss.xxxxxxxxxx

This is a sort-friendly format.

```
$ find . -printf "%T+ | %p\n"
2021-08-25+08:51:23.5500000000 | .
2021-08-24+22:59:26.9000000000 | ./wip-2-tables.sql
2021-08-25+10:27:40.3700000000 | ./subdir
2021-08-24+22:59:26.9000000000 | ./subdir/demo.txt
2021-08-25+10:27:40.3700000000 | ./subdir/demo2.txt
2021-08-16+00:00:00.0000000000 | ./month.sql
2021-08-25+08:51:23.5500000000 | ./solutions
```

Sort the results in ascending order (oldest file at the top, newest file at the bottom) by piping the results to sort.

```
$ find . -printf "%T+ | %p\n" | sort
2021-08-16+00:00:00.0000000000 | ./month.sql
2021-08-24+22:59:26.9000000000 | ./subdir/demo.txt
2021-08-24+22:59:26.9000000000 | ./wip-2-tables.sql
2021-08-25+08:51:23.5500000000 | .
2021-08-25+08:51:23.5500000000 | ./solutions
2021-08-25+10:27:40.3700000000 | ./subdir
2021-08-25+10:27:40.3700000000 | ./subdir/demo2.txt
```

Similar commands can be used to find recently modified files.

To sort the results in descending order (newest file at the top, oldest file at the bottom), use sort -r to reverse the sort order.

```
$ find . -printf "%T+ | %p\n" | sort -r
2021-08-25+10:27:40.3700000000 | ./subdir/demo2.txt
2021-08-25+10:27:40.3700000000 | ./subdir
2021-08-25+08:51:23.5500000000 | ./solutions
2021-08-25+08:51:23.5500000000 | .
2021-08-24+22:59:26.9000000000 | ./wip-2-tables.sql
2021-08-24+22:59:26.9000000000 | ./subdir/demo.txt
2021-08-16+00:00:00.0000000000 | ./month.sql
```

Formatting Tips

Specify minimum field width for a directive by adding a number between % and the letter of the directive.

For instance, say you're using %TA to show the weekday. You can specify the min width to 15 by adding a 15 between % and TA, like this %15TA.

```
$ find . -printf "%15TA | %p\n";
      Wednesday | .
        Tuesday | ./demo.txt
      Wednesday | ./demo2.txt
```

This is extremely useful when you use a directive like %TA which has variable length. It keeps the results tidy and readable.

In the example above, the weekday column is right-aligned. To change the alignment to left align, just add a minus sign (-) in front of the field width.

```
%15TA -> %-15TA
```

```
$ find . -printf "%-15TA | %p\n";
Wednesday       | .
Tuesday         | ./demo.txt
Wednesday       | ./demo2.txt
```

Mix Matching Directives

Using the directives described above, you can customize find's results in a way that is useful to you. Here are some examples.

1. Show the last modified time, weekday, and permission for all files (files only). Sort the results such that the most recently changed files are at the bottom.

```
$ find . -type f -printf "%10T+ %-10TA | %m | %p\n" | sort
2021-08-16+00:00:00.0000000000 Monday     | 644 | ./month.sql
2021-08-24+22:59:26.9000000000 Tuesday    | 644 | ./subdir/demo.txt
```

```
2021-08-24+22:59:26.9000000000 Tuesday    | 644 | ./wip-2-tables.sql
2021-08-25+10:27:40.3700000000 Wednesday  | 644 | ./subdir/demo2.txt
```

2. For all directories under the current directory, print the depth, file path, owner, group, permission, and modification date.

```
$ find . -type d -printf "%d    %-30p %-10u %-10g %-5m %T+\n" | sort
0    .                              hgcheah    hgcheah    755   2021-08-25+09:19:03.46000000
1    ./links                        hgcheah    hgcheah    755   2021-08-25+09:19:18.40000000
1    ./mysql-tut                    hgcheah    hgcheah    755   2021-08-24+22:59:26.90000000
1    ./sql                          hgcheah    hgcheah    755   2021-08-24+22:59:14.26000000
2    ./links/dst                    hgcheah    hgcheah    755   2021-08-25+09:50:11.85000000
2    ./links/src                    hgcheah    hgcheah    755   2021-08-25+09:48:02.20000000
2    ./mysql-tut/date               hgcheah    hgcheah    755   2021-08-25+08:51:23.55000000
2    ./sql/mysql-tut                hgcheah    hgcheah    755   2021-08-24+22:59:14.26000000
2    ./sql/upwork-sql-test          hgcheah    hgcheah    755   2021-08-24+22:59:14.26000000
3    ./mysql-tut/date/subdir        hgcheah    hgcheah    755   2021-08-25+10:27:40.37000000
```

Linux Find Tutorials

Find: files modified between X and Y days ago

Find: recently modified files

Find: printf

#Find #Linux

Hank Cheah

Software developer looking for remote Golang/Python opportunities.