HOWTO zum Kommando "find"

(C) 2006-2024 T.Birnthaler/H.Gottschalk <howtos(at)ostc.de>
OSTC Open Source Training and Consulting GmbH
http://www.ostc.de

$Id: find-HOWTO.txt,v 1.44 2025/02/23 20:14:55 tsbirn Exp $

Dieses Dokument beschreibt die Kommandos "find" und "locate".

INHALTSVERZEICHNIS

1) Beschreibung
1.1) Die wichtigsten Bedingungen CONDITION
1.2) Weitere wichtige Bedingungen CONDITION
1.3) Bezugszeitpunkt von Datumvergleichen
1.4) Mögliche Aktionen ACTION
1.5) Verknüpfung von Bedingungen CONDITION
1.6) Nützliche "find"-Beispiele
1.7) GNU-"find" kennt weitere Startpunkt-Angabe PATH
1.8) GNU-"find" kennt weitere Bedingungen CONDITION
1.9) GNU-"find" kennt weitere Aktionen ACTION
1.10) GNU-"find" kennt weitere Verknüpfungen
1.11) GNU-"find" kennt folgende Optionen
2) Performance-Problem von "find -exec"
3) "find"-Ersatz "locate"

---

1) Beschreibung

Das Kommando "find" durchsucht Verzeichnisbäume rekursiv nach Dateien mit bestimmten Eigenschaften. Suchkriterien können neben dem Dateinamen alle (im Inode vorhandenen) Dateiattribute (Metadaten) wie Dateityp, Besitzer, Besitzergruppe, Zugriffsrechte, Dateilänge, Zeitstempel, ... sein. Diese Suchkriterien sind beliebig über AND, OR, NOT und Klammerung kombinierbar.

"find" listet entweder alle zu den Bedingungen CONDITION passenden Dateinamen auf (Standardverhalten) oder führt beliebige Kommandos ACTION auf ihnen durch. Dazu wird der Verzeichnisbaum ab den angegebenen Verzeichnissen PATH rekursiv durchlaufen (das kann einige Zeit dauern). Die allgemeine Aufrufsyntax lautet:

```
find [PATH...] [CONDITION...] [ACTION...]
```

Ohne Suchpfad PATH beginnt die Suche im aktuellen Verz. ".". Die Bedingungen CONDITION sind per Default UND-verknüpft (d.h. müssen ALLE erfüllt sein), fehlt CONDITION gilt sie als erfüllt. Die Standard-Aktion ACTION ist "-print" (alle ab Verz. PATH zu den Bedingungen CONDITION passenden Dateinamen werden ausgegeben). D.h. ein Aufruf der Form

```
find     # oder
find .   #
```

führt zum Auflisten ALLER Verz.- und Dateinamen ab aktuellem Verzeichnis ".".

1.1) Die wichtigsten Bedingungen CONDITION

Bedingung	Bedeutung

-mtime [+-]DAY	Inhalt-Änderung vor +mehr/-weniger/genau Tagen [modify]
-name "PATTERN"	Dateiname passt zu Pattern (* ? [...] ... schützen!)
-perm [-+/]NNNN	Oktale Rechte (-0000=mind, +0000 /0000=any, 0000=exakt)
	1=x=Execute, 2=w=Write, 4=r=Read [permission]
-size [+-]NUM...	Dateigröße (+mehr/-weniger/genau) in 512-Byte Blöcken
...[GMkwcb]	(bzw. G=Giga M=Mega k=Kilo w=2-Byte c=Byte b=512-Byte)
-type TYPE	Dateityp (f=file d=directory l=symboliclink D=door
	c=characterdev. b=blockdev. p=named pipe s=socket)
-group GNAME/GID	Besitzergruppe (Name oder GID)
-user UNAME/UID	Besitzer (Name oder UID)

Die Zahlenwerte (DAY, NUM) haben folgende Bedeutung:

```
   +N = Größer/mehr als angegebener Wert (gleich nicht enthalten!)
   -N = Kleiner/weniger als angegebener Wert (gleich nicht enthalten!)
    N = GENAU der angegebene Wert
```

Bei "-perm" (RIGHT) bedeutet der Präfix:

```
   /NNNN = Mind. EINES der angegebenen Rechte NNNN vorhanden (modern)
   +NNNN = Mind. EINES der angegebenen Rechte NNNN vorhanden (veraltet)
   -NNNN = Mind. ALLE der angegebenen Rechte NNNN vorhanden
    NNNN = Genau ALLE der angegebenen Rechte NNNN vorhanden
```

1.2) Weitere wichtige Bedingungen CONDITION

Bedingung	Bedeutung
-atime [+-]DAY	Lesender Zugr. vor +mehr/-weniger/genau Tagen [access]
-ctime [+-]DAY	Status-Änderung (Inode) vor +mehr/... Tagen [change]
-iname "PATTERN"	Dateinamen-Muster (* ? [...] ... vor Shell schützen!),
	GROSS/Kleinschreibung egal [ignorecase]
-path "PATTERN"	Pfadnamen-Muster (* ? [...] ... vor Shell schützen!)
-ipath "PATTERN"	Analog, aber GROSS/Kleinschreibung egal [ignorecase]
-inum NUM	Inode-Nummer NUM
-links [+-]NUM	Anzahl Hardlinks +mehr/-weniger/genau NUM (Anzahl Namen)
-nogroup	KEINER Gruppe in "/etc/groups" zugeordnet (GID ohne Name)
-nouser	KEINEM Benutzer in "/etc/passwd" zugeord. (UID ohne Name)

1.3) Bezugszeitpunkt vom Datumvergleich

Beim Datumvergleich per -mtime/-atime/-ctime DAY (bzw. -mmin/-amin/-cmin MIN) ist die Basis für die Tages/Minutenzählung der STARTZEITPUNKT des Kommandos "find" (d.h. eine DYNAMISCHE Definition!). Die Bedeutung der "Tage" DAY bzw. der Minuten MIN ist daher:

```
  0 =  0:00:00h-23:59:59h VOR Startzeitpunkt (bzw. vor 00:00 bei -daystart)
  1 = 24:00:00h-47:59:59h VOR Startzeitpunkt (bzw. vor 00:00 bei -daystart)
  2 = 48:00:00h-71:59:59h VOR Startzeitpunkt (bzw. vor 00:00 bei -daystart)
  3 = 72:00:00h-95:59:59h VOR Startzeitpunkt (bzw. vor 00:00 bei -daystart)
  ...
                                                        |
                                          Startzeitpunkt | 00:00 akt. Tag
                                                        |   (-daystart)
                                                        |
```

```
                    VERGANGENHEIT                          |   ZUKUNFT
                                                           |
  ...|<= 6 =>|<= 5 =>|<= 4 =>|<= 3 =>|<= 2 =>|<= 1 =>|<= 0 =>|
  ...|167-144|143-120|119-96h| 95-72h| 71-48h| 47-24h| 23-00h|
  <=========== +3 ==========>|       |<========= -3 ======================>
  ...|       |<============= +0 und -6 =============>|       |
  ...|<=================== -7 (akt. Woche + Zukunft!) ====================>
                                                           |
```

HINWEIS: Option "-daystart" legt die Basis des Zeitvergleichs auf den Beginn 00:00 des aktuellen Tages fest (statt auf den Startzeitpunkt von "find").

1.4) Mögliche Aktionen ACTION

Aktion	Beschreibung
-print	Ausgabe gefundener Dateinamen (Standard!)
-ls	Ausgabe analog "ls -dils" (directory inode long size)
-exec CMD {} \;	Kommando CMD auf gefundenen Dateien ausführen ({} = gef. Dateiname, \; = Kommando-Ende, quotieren wg. Shell)
-ok CMD {} \;	Analog "-exec", verlangt vorher Bestätigung mit "y"(es)
-delete	Datei + Verz. (mit Inhalt) löschen (Vorsicht!)

1.5) Verknüpfung von Bedingungen CONDITION

Verknpf	Beschreibung
\(...\)	Klammerung (quotieren wg. Shell!)
\!	Negation (quotieren wg. Shell!)
-a	UND-Verknüpfung (Standard falls keine Verknüpfung angegeben!)
-o	ODER-Verknüpfung

HINWEIS: Die Verknüpfungen sind nach fallender Vorrang angegeben, durch Klammerung lässt sich dieser Vorrang durchbrechen.

TIPP: Wird eine ODER-Verknüpfung -o verwendet, sollten die verknüpften Teile meist geklammert sein, da der Vorrang von ODER der kleinste ist:

```
find ... \( -name '*.pdf' -o -type f \)
```

1.6) Nützliche "find"-Beispiele

Suche nach Dateinamen:

find -name "*.c"	C-Dateien "*.c" ab akt. Verz. "."
find . -name "*.c"	Analog
find .. -name "*.c"	C-Dateien "*.c" ab Eltern-Verz. "."
find \! -name "*.c"	Alle außer C-Dateien "*.c" ab "."
find ~ -name "*.c"	C-Dateien "*.c" ab Heimat-Verz. "~"
find / -name "*.c"	C-Dateien "*.c" ab Root-Verz. "/"
find / -iname "*.c"	Analog + GROSS/kleinschreibung ignor.
find / -regex ".*\.c"	Analog per Regulärem Ausdruck
find / -iregex ".*\.c"	Analog + GROSS/kleinschreibung ignor.
find /usr -type d -name "*man*"	Verz. in "/usr" mit "man" im Namen
find /usr -type d -regex ".*man.*"	Analog per Regulärem Ausdruck
find /usr /lib /etc -name "*.o"	Objekt-Dateien "*.o" in 3 angeg. Verz.

Suche nach Dateialter (letzte Änderung/letzter Zugriff):

find / -mtime 0	Heute geänderte Dateien
find / -mtime 1	Gestern geänderte Dateien
find / -mtime -7	Letzte Woche (Tage 0-6) geänd. Dateien
find / -mtime +6 -mtime -14	Vorletzte Woche (Tage 7-13) geänd. Dat.
find / \(-mtime 7 -o -mtime 8 \ -o -mtime 9 -o -mtime 10 \ -o -mtime 11 -o -mtime 12 \ -o -mtime 13 \)	(analog: Tage einzeln aufzählen und per ODER verknüpfen)
find / -atime +365	Letzter Zugriff vor mehr als 1 Jahr
find / -atime +365 -daystart	(analog, aber Startzeitpunkt 00:00)
find / -newer FILE	Neuer als Datei FILE
find / -mmin -5	In letzten 5 Minuten geänd. Dateien
find / -mmin +15	Vor mehr als 15 Minuten geänd. Dateien

Suche nach Besitzer oder Besitzer-Gruppe:

find / -user tsbirn	Dateien des Anwenders "tsbirn"
find / \(-nouser -o -nogroup \)	Gehören keinem User oder keiner Gruppe
find / -uid +99 -uid -201	Gehören User mit 100 <= UID <= 200
find / -gid +0 -gid -100	Gehören Gruppe mit 1 <= GID <= 99
find / \(-uid 1000 -o -gid 1000 \) -print0 | xargs -0 chown 1005:1005	UID/GID 1000 --> 1005 konvertieren

Suche nach Zugriffsrechten:

find / -user root -perm -002	Gehören root + für andere schreibbar
find / -type f \(-perm -100 -o -perm -010 -o -perm -001 \)	Ausführbare Dateien (Programme)
find / -user root -perm -002	Gehören root + für andere schreibbar
find / -type f -perm -4000	Dateien mit SetUID-Recht (mindestens)
find / -type f -perm -2000	Dateien mit SetGID-Recht (mindestens)
find / -type d -perm -2000	Verz. mit SetGID-Recht (mindestens)
find / -type f -perm -1000	Dateien mit Sticky-Recht (mindestens)

Suche nach Inode und Anzahl Hardlinks:

find / -type f -inum 123	Dateien zu Inode 123
find / -type f \! -type l -links +1	Dateien mit Anz. Hardlinks > 1

Suche nach Dateiinhalt:

find / -type f -exec grep -il "made" {} \;	Dateien die Text "made" enthalten
find / -type f | xargs grep -il "made"	Analog
grep -ilR "made" /	Analog per grep "rekursiv"

Suche nach Dateien mit bestimmter Größe:

find / -type f -size 1234	Dateien mit Größe 1234 Byte
find / -type f -size +1M	Dateien größer als 1 MByte
find / -type f -size +1K -size -1M	Datei-Größe zw. 1 KByte und 1 MByte

Suche nach leeren Dateien/Verzeichnissen oder "dangling" Symlinks:

find / -type f -size 0	Leere Dateien
find / -type f -size 0 -ok rm {} \;	Leere Dateien löschen (mit Abfrage)
find / -type f -empty -ok rm {} \;	Analog
find / -type d -exec rmdir -p {} \;	Leere Verz. löschen (kein Inhalt)
find / -type d -empty -exec ...	Leere Verz. löschen (besser!)
find -L / -type l -exec rm {} \;	Broken Symlinks löschen (Ziel ex. nicht)
find / -xtype l -delete	Analog

TIPP: Alle Dateien im eigenen Heimatverz. auflisten, die in der vorletzten Woche (Tag 7-13) verändert wurden und den Text "muster" enthalten (GROSS/kleinschreibung egal):

```
find ~ -type f -mtime +6 -mtime -14 \              # f=file, m=modify
       -exec grep -li "muster" {} \; 2> /dev/null  # -l=list -i=ignorecase
find ~ -type f -mtime +6 -mtime -14 -print0 |      # Analog
       xargs grep -li "muster" 2> /dev/null        #
```

1.7) GNU-"find" kennt weitere Startpunkt-Angabe PATH

Bedingung	Bedeutung
-files0-from FILE	Startpunkt der Suche aus Datei einlesen (0-Byte trennt)
-files0-from -	Startpunkt der Suche von Stdin einlesen (0-Byte trennt)

HINWEIS: Hiermit können Datei/Verz.namen "sicher" übergeben werden, d.h. auch wenn sie Sonderzeichen der Shell enthalten (außer 0-Byte und "/" sind in einem Dateinamen ALLE Zeichen erlaubt).

1.8) GNU-"find" kennt weitere Bedingungen CONDITION

Bedingung	Bedeutung
-P	Symbolische Links NIE verfolgen (Standard!)
-L / -follow	Symbolische Links IMMER verfolgen [links, dereference]
-H	Symbolische Links NICHT verfolgen AUSSER Kmdo-Argumente
-xtype TYPE	Wie -type TYPE + bei symb. Link Zieltyp testen (-H/-P/-L)
-readable	Datei lesbar für akt. Benutzer (erweitert -perm)
-writable	Datei schreibbar für akt. Benutzer (erweitert -perm)
-executable	Datei ausführbar/Verz. durchsuchbar für akt. Ben. (erw...)
-maxdepth LVL	Max. LVL Verz. + Dateien absteigen (0=NUR Kmdo-Argumente)
-mindepth LVL	Verz. + Dateien bis LVL ignorieren (1=Kmdo-Argument ign.)
-d / -depth	Erst Verz.inhalt., dann Verz. selbst bearbeiten
	(Standard: Erst Verz. selbst, dann Verz.inhalt)
-mount / -xdev	Nur Verz. auf GLEICHEM Dateisystem betrachten
-prune	NICHT in Verz. absteigen
-noleaf	NICHT optimieren dass "." + ".." vorhanden bei -name...
-fstype TYPE	Filesystemtyp (mögl. Werte per -printf %F, z.B. "ext3")
-mmin [+-]MIN	Änderung vor +mehr/-weniger/genau MIN Minuten [modify]
-amin [+-]MIN	Lesender Zugriff vor ... [access]
-cmin [+-]MIN	Status-Änderung (Inode) vor ... [change]

-newer FILE	Inhalt-Änderung VOR Änderung an FILE [modify]
-anewer FILE	Lesender Zugriff VOR lesendem Zugriff auf FILE [access]
-cnewer FILE	Status-Änderung (Inode) VOR Änderung an FILE [change]
-newerXY FILE	X: a=access, B=birth, c=inode change, m=modif. time
	Y: t=FILE als Zeitangabe im "date -d" Format interpret.
-used [+-]DAY	Zugriff +mehr/-weniger/genau DAY Tage seit Änderung
-uid [+-]UID	Analog "-user" aber per Nummer (Bereich von/bis möglich)
-gid [+-]GID	Analog "-group" aber per Nummer (Bereich von/bis möglich)
-empty	Leere Datei/Verz. (leeres Verz. sonst schwer zu prüfen)
-regex "PATTERN"	Regulärer Ausdruck passt [regular expression]
-iregex "PATTERN"	Analog, aber GROSS/Kleinschr. egal [ignorecase]
-lname "PATTERN"	Symb. Link passt gemäß Shell-Muster "PATTERN"
-ilname "PATTERN"	Analog, aber GROSS/Kleinschr. egal [ignorecase]
-wholename "PAT"	Ganzer Pfadname passt gemäß Shell-Muster "PATTERN"
-iwholename "PAT"	Analog, aber GROSS/Kleinschr. egal [ignorecase]
-samefile "FILE"	Gleiche Inode-Nummer wie FILE (also Hardlink auf FILE)
-true	Immer wahr
-false	Immer falsch

HINWEIS: -regex ... muss VOLLSTÄNDIG auf Pfad passen (^ und $ sind wegzulassen!)
Teilmatch daher so zu schreiben: -regex ".*REGEX.*"
-E schaltet extended Regex (ERE) ein \(\) --> ( ) ? + {...}

1.9) GNU-"find" kennt weitere Aktionen ACTION

Bedingung	Bedeutung
-print0	Namen in gen. Liste durch "\0" (NUL-Bytes) trennen
	(Standard: durch "\n", für "xargs -0" sinnvoll)
-printf FMT	Benutzerdef. Ausgabe mit %-Formatelementen in FMT
-fls FILE	Analog "-ls", aber auf Datei FILE ausgeben
-fprint FILE	Analog "-print", aber auf Datei FILE ausgeben
-fprint0 FILE	Analog "-print0", aber auf Datei FILE ausgeben
-fprintf FILE FMT	Analog "-printf FMT", aber auf Datei FILE ausgeben
-exec CMD {} +	Analog "-exec", aber {} sammelt mögl. viele Dateien
-ok CMD {} +	Analog "-ok", aber {} sammelt mögl. viele Dateien
-execdir CMD {} \;	Analog "-exec", vorher Sprung in Verz. der gef. Datei
-okdir CMD {} \;	Analog "-ok", vorher Sprung in Verz. der gef. Datei
-execdir CMD {} +	Analog "-execdir", aber {} sammelt mögl. viele Dateien
-okdir CMD {} +	Analog "-okdir", aber {} sammelt mögl. viele Dateien
-quit	Sofort abbrechen (z.B. um nur 1 Datei zu finden)

Bei "-printf" und "-fprintf" sind folgende Formatbezeichner in FMT angebbar:

Fmt	Begriff	Bedeutung
%a	access	Letzter Zugriffszeitpunkt
%b	block	Größe in 512-Byte Blöcken

%c	change	Letzte Statusänderung
%d	depth	Tiefe im Verz.baum (0=Kommando-Argument)
%D	Device	Gerätenummer
%f	filename	Dateiname (letzte Komponente) ohne Pfadanteil
%F	File	Dateisystemtyp
%g	group	Besitzergruppenname (oder GID)
%G	GID	Besitzergruppen-ID (GID)
%h	dirpath	Pfadname ohne Dateiname (letzte Komponente)
%H		Kommando-Argument das Fund der Datei ausgelöst hat
%i	inode	Inode-Nummer
%k	kilo	Größe in 1K Blöcken
%l	link	Ziel eines Symb. Links (leer falls keiner)
%m	perm mode	Zugriffsrechte in Oktalnotation
%M	perm Mode	Zugriffsrechte in symbolischer Form
%n	hardlink	Anzahl Hardlinks auf Datei
%p	path	Vollständiger Dateiname
%P	Path	Dateiname ohne Kommando-Argument das Dateifund auslöst
%s	size	Dateigröße in Byte
%S	Sparse	Leerheit von Dateien (1.0=keine, <=1.0)
%t	change time	Letzte Inhalt-Änderung
%u	user	Besitzername (oder UID)
%U	UID	Besitzer-ID (UID)
%y	type	Dateityp "fdlcbpxD" (analog "ls -l")
%Y	type	Dateityp mit Verfolgung symb. Links (L=Loop, N=Nichtex.)
%%		Prozentzeichen "%"
\n		Zeilenvorschub (analog \r \t \v \f \\)

1.10) GNU-"find" kennt weitere Verknüpfungen

Verkn	Bedeutung
-and	Analog "-a" (Standard falls keine andere Verknüpfung angegeben!)
-or	Analog "-o"
-not	Analog "!" ("!" ist normalerweise zu schützen: "\!"
,	Erst linke Seite (Ergebnis verwerfen) dann rechte Seite (Ergebnis)

1.11) GNU-"find" kennt folgende Optionen

Option	Bedeutung
-daystart	Tagesbeginn 00:00 statt "find"-Startzeitpunkt als Basis!
-regextype TYPE	Typ der Regulären Ausdrücke in -regex/-iregex festlegen (Default: "emacs", Liste per "-regextype help")
-OLEVEL	Optimierung (Default: 0=1, 2=reordering, 3=cost based opt)
-O DEBUGOPT	Debugging aktivieren (exec,opt,rates,search,stat,time,tree) (alle per "-D all", Liste per "-D help")
-ignore_readdir_race	Fehlermeldung unterdrücken, falls -delete Datei gelöscht hat, bevor find Datei-Status ermittelt

2) Performance-Problem von "find -exec"

"find" startet das nach "-exec/-ok" angegebene Kommando für JEDE gefundene Datei neu. Dies ist bei sehr vielen gefundenen Dateien ineffektiv, da jedesmal ein neuer Prozess erzeugt wird.

```
find $HOME -type f -size +200 -exec gzip {} \;
```

Effektiver sind die folgenden Aufrufe (einfügen des Ergebnisses von "find" auf der Kommandozeile per Kommando-Substitution `...` oder $(...) bzw. sammeln der MAXIMAL möglichen Menge an Argumenten vor einem Aufruf von "gzip" per "xargs"):

```
gzip `find $HOME -type f -size +200 -print`          # 1) sh
gzip $(find $HOME -type f -size +200 -print)         # 2) bash, ksh
find $HOME -type f -size +200 -print | xargs gzip    # 3) xargs
```

Allerdings kann bei den Varianten 1)+2) ein "Überlaufproblem" eintreten, falls die Dateinamenliste zu lang wird (abhängig von der Shell ab 1-2 Mio Zeichen oder mehreren 1000 Namen).

Alle obigen Varianten 1)-3) haben den Nachteil, dass bei bestimmten Sonderzeichen in Dateinamen (Whitespace, ...) die erzeugte Liste falsch interpretiert wird. Bei GNU-"find" und GNU-"xargs" gibt es dafür eine Lösung, die als Trennzeichen das NUL-Byte "\0" verwendet (statt Zeilenvorschub "\n"), welches PRINZIPIELL nicht in einem Dateinamen vorkommen kann (neben dem "/"):

```
find $HOME -type f -size +200 -print0 | xargs -0 gzip       # oder
find $HOME -type f -size +200 -print0 | xargs --null gzip   #
```

HINWEIS: Viele weitere Kommandos, die Listen von Dateinamen verarbeiten, kennen inzwischen diesen Schalter -0/--null (z.B. tar, cpio, grep, sed, ...).

3) "find"-Ersatz "locate"

"locate" ist eine schnellere, aber nicht so leistungsfähige Variante von "find" (kann nur nach DATEINAMEN suchen, nicht aber nach sonstigen Dateiattributen). Sucht nicht im Dateibaum direkt, sondern in Indexdatei "/var/lib/locatedb".

```
locate "PATTERN"   # Datei gemäß Muster PATTERN suchen (* ? [...] ...)
```

Diese Indexdatei muss vorher mit "updatedb" erzeugt und von Zeit zu Zeit aktualisiert werden (da Änderungen am Dateisystem darin nicht vermerkt werden). Dies kann einige Zeit dauern und wird meist täglich um Mitternacht über einen "crontab"-Job erledigt (falls der Rechner zu diesem Zeitpunkt läuft). "updatedb" kann aber auch per Hand aufgerufen werden (mit root-Rechten!):

```
sudo updatedb      # locate-Datenbank updaten
```