

HOWTO zum Kommando "find"

(C) 2006-2017 T.Birnthaler/H.Gottschalk <howtos(at)ostc.de>
OSTC Open Source Training and Consulting GmbH
<http://www.ostc.de>

\$Id: find-HOWTO.txt,v 1.39 2018/07/31 06:13:40 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 Bedingungen CONDITION
 - 1.8) GNU-"find" kennt weitere Aktionen ACTION
 - 1.9) GNU-"find" kennt weitere Verknüpfungen
- 2) Performance-Problem von "find"
- 3) "find"-Ersatz "locate"

1) Beschreibung

Das äußerst mächtige 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...]
```

HINWEIS: Bei fehlendem Suchpfad PATH beginnt die Suche im aktuellen Verz. ".". 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" (d.h. alle ab Verz. PATH zu den Bedingungen CONDITION passenden Dateinamen werden ausgegeben). D.h. ein Aufruf der Form

```
find
```

führt zum Auflisten ALLER Verz.- und Dateinamen ab dem aktuellen 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 [+ /]RIGHT	Oktale Rechte (-000=mind, +000 bzw. /000=any, 000=exakt) 1=x=Execute, 2=w=Write, 4=r=Read [permission]
-size [+ -]NUM... ...[GkMkwb]	Dateigröße (+mehr/-weniger/genau) in 512-Byte Blöcken (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)

Bei Zahlenwerten (DAY, NUM) bedeutet die Zahl:

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

1.3) Bezugszeitpunkt von Datumvergleichen

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 => <= -1 => ...		
... 168-144 144-120 120-96h 96-72h 72-48h 48-24h 24-00h		
<===== +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 löschen (Vorsicht!)

1.5) Verknüpfung von Bedingungen CONDITION

Verknüpfung	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.

1.6) Nützliche "find"-Beispiele

Beispiel	Beschreibung (Std: Ergebnis ausgeben)
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.
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 / \(-mtime 7 -o -mtime 8 \	(analog: Tage einzeln aufzählen und
-o -mtime 9 -o -mtime 10 \	per ODER verknüpfen)
-o -mtime 11 -o -mtime 12 \	
-o -mtime 13 \)	
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 / -user tsbirn	Dateien des Anwenders "tsbirn"
find / -user root -perm -002	Gehören root + für andere schreibbar
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 / \(-uid 1000 -o -gid 1000 \)	UID/GID 1000 --> 1005 konvertieren
-print0 xargs -0 chown 1005:1005	
find / -type f \(-perm -100 -o	Ausführbare Dateien (Programme)
-perm -010 -o -perm -001 \)	
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)
find / -type f -inum 123	Dateien zu Inode 123
find / -type f \! -type l -links +1	Dateien mit Anz. Hardlinks > 1
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" aufrufen)
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)

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
```

1.7) GNU-"find" kennt weitere Bedingungen CONDITION

Bedingung	Bedeutung
-P	Symbol. Links NIE verfolgen (Standard!)

Jul 31, 18 8:13

find-HOWTO.txt

Page 4/5

-L / -follow	Symbol. Links IMMER verfolgen [links, dereference]	
-H	Symbol. Links NICHT verfolgen AUSSER Kmdo-Argumente	
+-----+		
-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...	
+-----+		
-maxdepth LEV	Max. LEV Verz. + Dateien absteigen (0=nur Kmdo-Argumente)	
-mindepth LEV	Verz. + Dateien bis LEV ignorieren (1=Kmdo-Argument ign.)	
+-----+		
-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 +mehr/... MIN Minuten [access]	
-cmin [+]-MIN	Inode-Änderung vor +mehr/... MIN Minuten [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 Status-Änd. an FILE [change]	
-newerXY FILE	X: a=access, B=birth, c=inode change, m=modification time Y: t=FILE als Zeitangabe im "date -d" Format interpretiert	
+-----+		
-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]	
+-----+		
-wholename "PAT"	Ganzer Pfadnamen gemäß Shell-Muster "PATTERN" passt	
-iwholename "PAT"	Analog, aber Groß/Kleinschr. egal [ignorecase]	
+-----+		
-samefile "FILE"	Gleiche Inode-Nummer wie FILE (also Hardlink auf FILE)	
-daystart	Tagesbeginn 00:00 statt "find"-Startzeitpunkt als Basis!	
+-----+		

1.8) 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

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

Fmt	Bedeutung
%a	Letzter Zugriffszeitpunkt [access]
%b	Größe in 512-Byte Blöcken [block]
%c	Letzte Statusänderung [change]
%d	Tiefe im Verz.baum (0=Kommando-Argument) [depth]
%D	Gerätenummer [Device]
%f	Reiner Dateiname (letzte Komponenten) ohne Pfadanteil [filename]
%F	Dateisystemtyp [File]
%g	Besitzergruppenname (oder GID) [group]
%G	Besitzergruppen-ID (GID) [GID]
%h	Pfadname ohne Dateiname (letzte Komponente) [dirpath]
%H	Kommando-Argument das Fund der Datei ausgelöst hat
%i	Inode-Nummer [inode]
%k	Größe in 1K Blöcken [kilo]
%l	Ziel eines Symb. Links (leer falls keiner) [link]
%m	Zugriffsrechte in Oktalnotation [permission]
%M	Zugriffsrechte in symbolischer Form [Permission]
%n	Anzahl Hardlinks auf Datei [hardlink]
%p	Vollständiger Dateiname [path]

```

| %P | Dateinamen ohne Kommando-Argument das Fund der Datei auslöste [Path] |
| %s | Dateigröße in Byte [size] |
| %S | Leerheit von Dateien (1.0=keine, <=1.0) [Sparse] |
| %t | Letzte Inhalt-Änderung [change time] |
| %u | Besitzername (oder UID) [user] |
| %U | Besitzer-ID (UID) [UID] |
| %y | Dateityp "fdlcbpxD" (analog "ls -l") [type] |
| %Y | Dateityp mit Verfolgung symb. Links (L=Loop, N=Nichtexistent) [type] |
+-----+
| %% | Prozentzeichen "%" |
| \n | Zeilenvorschub (analog \r \t \v \f \\) |
+-----+

```

1.9) 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) |
+-----+

```

2) Performance-Problem von "find"

"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 Kommandosubstitution `...` 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, wenn die einzufügende Liste zu lang ist (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 ein Lösung, die als Trennzeichen das NUL-Byte "\0" verwendet (statt Zeilenvorschub "\n"), das 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 (wenn der Rechner zu diesem Zeitpunkt läuft). "updatedb" kann aber auch per Hand aufgerufen werden (mit root-Rechten!):

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