

HOWTO zum Kommando "find"  
 (C) 2006-2008 T.Birnthaler/H.Gottschalk <howtos(at)ostc.de>  
 OSTC GmbH, <http://www.ostc.de>  
 \$Id: find-HOWTO.txt,v 1.8 2008-12-22 21:44:42 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 Datumsvergleichen
  - 1.4) Mögliche Aktionen ACTION
  - 1.5) Verknüpfung von Bedingungen CONDITION
  - 1.6) Einige einfache "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 nach Dateien mit bestimmten Eigenschaften. Als Suchkriterien können neben dem Dateinamen alle (im Inode vorhandenen) Dateiattribute wie Dateityp, Besitzer, Zugriffsrechte, Dateilänge, Zeitstempel, ... verwendet werden. Diese Suchkriterien sind beliebig über AND, OR, NOT und Klammerung kombinierbar.

"find" listet entweder alle zu den Bedingungen passenden Dateinamen aus (Standardverhalten) oder führt beliebige Kommandos auf ihnen durch. Dazu wird der Verzeichnisbaum ab dem/den angegebenen Verzeichnis(sen) rekursiv durchlaufen (kann einige Zeit dauern). Die Aufrufsyntax lautet:

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

Bei fehlendem Suchpfad PATH beginnt die Suche ab dem aktuellen Verzeichnis ".", die Bedingungen CONDITION sind per Default AND-verknüpft (d.h. müssen alle erfüllt sein) und die Standard-Aktion ACTION ist "-print" (d.h. die ab dem Suchpfad PATH zu den Bedingungen CONDITION passenden Dateinamen werden auf dem Bildschirm ausgegeben).

### 1.1) Die wichtigsten Bedingungen CONDITION

HINWEIS: Ein Präfix "+" bedeutet "größer/mehr als", ein Präfix "-" bedeutet "kleiner/weniger als", sonst wird der genau passende Wert gesucht.

-mtime [+/-]DAYS	Inhalt-Änderung vor +mehr/-weniger/genau 24h-Tagen [modify]
-name "PATTERN"	Dateiname-Muster (* ? [...] ... vor Shell schützen!)
-perm [+/-]RIGHT	Oktale Rechte (-000=mind. /000 bzw. +000=any 000=exakt) [permission]
-size [+/-]NUM... ...[GMkwcB]	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=characterdevice b=blockdevice p=named pipe s=socket)
-group GNAME/GID	Besitzergruppe (Name oder GID)
-user UNAME/UID	Besitzer (Name oder UID)

### 1.2) Weitere wichtige Bedingungen CONDITION

-atime [+/-]DAYS	Lesender Zugr. vor +mehr/-weniger/genau 24h-Tagen [access]
-ctime [+/-]DAYS	Status-Änderung (Inode) vor +mehr/... 24h-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)

Oct 06, 09 20:28

**find-HOWTO.txt**

Page 2/4

-nogroup	KEINER Gruppe aus "/etc/groups" zugeordnet (GID ohne Name)
-nouser	KEINEM Benutzer aus "/etc/passwd" zugeord. (UID ohne Name)

## 1.3) Bezugszeitpunkt von Datumsvergleichen

Bei Datumsvergleichen mit `-mtime/-atime/-ctime` (bzw. `-mmin/-amin/-cmin`) ist die Basis für die Tageszählung der jeweilige Startzeitpunkt des "find"-Kommandos (d.h. eine höchst dynamische Definition!). Die Bedeutung der "Tage" DAYS ist daher:

0 = 0:00:00h-23:59:59h vor dem Startzeitpunkt	
1 = 24:00:00h-47:59:59h vor dem Startzeitpunkt	
2 = 48:00:00h-71:59:59h vor dem Startzeitpunkt	
...	
	Startzeitpunkt --+-- 00:00 aktueller Tag
	(exakt)    (-daystart)
	vv
...  144-120 120-96h  96-72h  72-48h  48-24h  24-0h  0-24h  ...	
... == 5 == == 4 == == 3 == == 2 == == 1 == == 0 ==	
...<===== +3 ===== == 3 ==	===== -3 =====>...

HINWEIS: Mit Option "`-daystart`" läßt sich die Basis des Zeitvergleichs auf den Beginn 00:00 des aktuellen Tages festlegen (statt Aufrufzeitpunkt von "find").

## 1.4) Mögliche Aktionen ACTION

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

## 1.5) Verknüpfung von Bedingungen CONDITION

HINWEIS: Die Verknüpfungen sind nach fallender Vorrang angegeben.

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

## 1.6) Einige einfache "find"-Beispiele

find / -name "*.c" -print	C-Dateien ab Root-Verz.
find .. -mtime 1 -print	Gestern geänderte Dateien ab ".."
find / -mtime -7 -print	Letzte Woche (Tage 0-6) modifiz. Dateien
find / -user tsbirn -print	Dateien des Anwenders "tsbirn"
find /usr -type d -name "*man*"	Verz. in "/usr" mit "man" im Namen
find / -size 0 -ok rm {} \;	Leere Dateien löschen (mit Abfrage)
find / -exec grep -l "made" {} \;	Dateien die Text "made" enthalten
find / -user root -perm -002	Gehören root und für andere schreibbar
find / +uid 99 -uid 201	100 <= User-ID <= 200

## 1.7) GNU-"find" kennt weitere Bedingungen CONDITION

-P	Symbol. Links NIE verfolgen (Standard!)
-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 (mögliche Werte per -printf %F, z.B. "ext3")
-mmin [+]-MIN	Änderung vor +mehr/-weniger/genau MIN Minuten [modify]
-cmin [+]-MIN	Lesender Zugriff vor +mehr/... MIN Minuten [access]
-amin [+]-MIN	Status-Änderung (Inode) 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]
-used [+]-DAYS	Zugriff +mehr/-weniger/genau DAYS 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 schwierig zu prüfen)
-regex "PATTERN"	Regulärer Ausdruck passt [regular expression]
-iregex "PATTERN"	Analog, aber Groß/Kleinschr. egal [ignorecase]
-wholename "PAT"	Ganzer Pfadnamen gemäß Shell-Muster passt
-iwholename "PAT"	Analog, aber Groß/Kleinschr. egal [ignorecase]
-samefile "FILE"	Gleiche Inode-Nummer wie FILE (also Hardlink auf gl. Datei)
-daystart	Tagesbeginn 00:00 statt "find"-Startzeitpunkt als Basis

## 1.8) GNU-"find" kennt weitere Aktionen ACTION

-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 (%f=filename %p=filepath %h=dirpath %s=size %m=permissionbits %a/%t/%c=access/change/modifytime %i=inodenr %n=hardlinks %F=fstype %l=symlink/empty %d=depth(0=Kmdo-Arg) %y=filetype(-type fdclbpd) %u/U=user/UID %g/G=group/GID \n=Zeilenvorschub %%=%)
-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

## 1.9) GNU-"find" kennt weitere Verknüpfungen

-and	analog "-a" (Standard falls keine andere Verknüpfung angegeben!)
-or	Analog "-o"
-not	Analog "!"
,	Erst linke Seite (Ergebnis verwerfen) dann rechte Seite (Ergebnis)

## 2) Performance-Problem von "find"

"find" startet das nach "-exec/-ok" angegebene Kommando für jede Datei erneut. 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` # sh
gzip $(find $HOME -type f -size +200 -print) # bash, ksh
find $HOME -type f -size +200 -print | xargs gzip
```

Allerdings kann bei den Varianten 1+2 ein "Überlaufproblem" eintreten, wenn die einzufügende Liste zu lang ist (abhängig von der Shell ab 20.000 Zeichen oder 4000 Namen). Alle Varianten haben den Nachteil, dass bei bestimmten Sonderzeichen in Dateinamen ("Whitespaces", ...) die 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"), das prinzipiell nicht in einem Dateinamen vorkommen kann (neben dem "/"):

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

### 3) "find"-Ersatz "locate"

"locate" ist eine schnellere, aber nicht so leistungsfähige Variante von "find" (kann nur nach Dateinamen suchen, aber nicht nach sonstigen Dateiattributen). Sie sucht nicht im Dateibaum direkt, sondern in einer Indexdatei namens "/var/lib/locatedb". Diese Indexdatei muss vorher mit "updatedb" erzeugt und von Zeit zu Zeit aktualisiert 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:

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