Wiki / find

Dieser Artikel wurde für die folgenden Ubuntu-Versionen getestet:

Dieser Artikel ist größtenteils für alle Ubuntu-Versionen gültig.

Zum Verständnis dieses Artikels sind folgende Seiten hilfreich:

1. **Ein Terminal öffnen**

Inhaltsverzeichnis

1. Installation
2. Übersichten
3. Beispiele
4. Weitere Informationen
5. Links

[//media-cdn.ubuntu-de.org/wiki/attachments/10/28/terminal.png] find ist ein Kommandozeilenprogramm für die Dateisuche. Dabei kann man auf vielfältige Weise die Suche filtern, z.B. nach Dateiname, -alter, -größe und die Suchergebnisse weiterverarbeiten und/oder formatiert ausgeben.

Da unter unixoiden Systemen der Leitsatz *"Alles ist eine Datei"* gilt, werden auch Verzeichnisse gefunden. Eine Alternative zu find (mit Vor- und Nachteilen) bietet der Befehl **locate** [https://wiki.ubuntuusers.de/locate/].

Installation

Das Programm ist im Paket

- **findutils**

von Ubuntu enthalten und ist deshalb auf jedem System vorinstalliert.

Übersichten

Bevor einige praktische Beispiele die vielfältigen Verwendungsmöglichkeiten von find zeigen, zwei kurze Übersichten.

Suchkriterien

Im Folgenden sind einige, wenn auch nicht alle Suchkriterien aufgeführt.

Suchkriterien für find	
Kriterium	**Beschreibung**
-amin n	Es wird nach Dateien gesucht, die innerhalb der letzten n Minuten berührt wurden.
-atime n	Es wird nach Dateien gesucht, die zuletzt vor n*24h berührt wurden. Dabei werden Bruchteile ignoriert, um -atime +1 zu entsprechen muss die Datei also vor höchstens zwei Tagen berührt worden sein.
-mtime n	Es wird nach Dateien gesucht, die zwischen n und n+1 Tage alt sind.
-mtime +n	Es wird nach Dateien gesucht, die älter als n Tage sind. Beispiel: -mtime +3 alles was älter als 3 Tage ist.
-mtime -n	Es wird nach Dateien gesucht, die weniger als n Tage alt sind.

-name Datei	Es wird nach Dateien mit dem Namen "Datei" gesucht. Sollen bei der Suche Platzhalter Verwendung finden, so muss der ganze Ausdruck in Anführungszeichen gestellt werden, zum Beispiel -name "*.txt"
-iname Datei	Es wird nach Dateien mit dem Namen "Datei" - ohne Beachtung der Groß und Kleinschreibung - gesucht.
-newer Datei	Es wird nach Dateien gesucht, die nach Datei verändert wurden.
-nouser	Es wird nach Dateien gesucht, deren User-ID keinem Benutzer entspricht.
-nogroup	Es wird nach Dateien gesucht, deren Gruppen-ID keiner Gruppe entspricht.
-type f	Es wird nur nach regulären Dateien gesucht.
-type d	Es wird nur nach Verzeichnissen gesucht.
-depth	Es wird der Inhalt des Verzeichnisses vor dem Verzeichnis bearbeitet.
-maxdepth Zahl	Es werden nur Unterverzeichnisse bis zur Tiefe "Zahl" durchsucht.
-size n[cwbkMG]	Die Datei belegt n Zuordnungseinheiten. Die folgenden Endungen können verwendet werden: b für 512-Byte-Blöcke (Standard), c für Bytes, 'w' für Zwei-Byte-Wörter, 'k' für Kilobytes, 'M' für Megabytes, 'G' für Gigabytes.
-user User	Es werden nur Dateien des Benutzers User gesucht.

Aktionen

Aktionen für find	
Aktion	**Beschreibung**
-fprint Datei	Gibt die gefunden Dateinamen nicht auf die Standardausgabe (Bildschirm) aus, sondern schreibt diese in die Datei "Datei"
-exec Kommando {} \;	Wendet auf alle gefundenen Dateien den Shellbefehl "Kommando" an. {} steht dabei als Platzhalter für die gefundenen Dateinamen. Das Zeichen ; terminiert den von find aufzurufenden Shellbefehl, damit es nicht unbeabsichtigt von der Shell interpretiert wird muss es mit \ maskiert werden.
-execdir Kommando {} +	Wendet auf alle gefundenen Dateien den Shellbefehl "Kommando" an. Im Ggs. zu -exec wird das Kommando im Verzeichnis, in dem die Datei liegt, ausgeführt. Das Plus kann (ebenso wie im Kommando -exec) statt ";" verwendet werden, wenn die {} der letzte Parameter sind; dann werden mehrere Funde auf einmal an Kommando übergeben.
-ok Kommando {} \;	Wie -exec, allerdings wird vor jeder Aktion eine Bestätigung erfragt. {} steht dabei als Platzhalter für die gefundenen Dateinamen
-okdir Kommando {} +	Wie eine Kombination von -ok und -execdir, d.h. es wird eine Bestätigung erfragt, und das Kommando wird im Fundordner ausgeführt.
-delete	Löscht die gefundenen Dateien

Beispiele

Es gibt eine Unzahl weiterer Beispiele, z.B. die Suche nach Eigentümer, Berechtigungen, Dateisystem u.v.m - eine komplette Übersicht bieten die **Manpage** [https://wiki.ubuntuusers.de/man/], die **Infoseiten** [https://wiki.ubuntuusers.de/info/] zu find sowie die sehr ausführliche Seite **findutils auf gnu.org** [http://www.gnu.org/software/findutils/manual/html_mono/find.html].

Startverzeichnis(se)

- Suche nach allen Dateien und allen Unterverzeichnissen im aktuellen Verzeichnis:

```
find
```

- Suche nach allen Dateien und Verzeichnissen im Unterverzeichnis **foo** des aktuellen Verzeichnisses:

```
find foo
```

- Suche alle Dateien und Verzeichnisse im übergeordneten Verzeichnis:

```
find ../
```

- Suche in einem ganz anderen Verzeichnis beginnen (im absoluten Pfad **/tmp**):

```
find /tmp
```

- Suche in zwei Verzeichnissen (sucht sowohl in **/tmp** als auch in **/boot**):

```
find /tmp /boot
```

- Suche überall (Sucht im Wurzelverzeichnis alle Dateien und alle Unterverzeichnisse - das kann dauern!):

```
find /
```

Dateitypus (Verzeichnis/Datei)

- Finde nur Dateien

```
find -type f
```

Das `f` steht für 'files'. Findet keine Verzeichnisse, aber alle herkömmlichen Dateien in allen Unterverzeichnissen.

- Finde nur Verzeichnisse

```
find -type d
```

Das `d` steht für 'directories'. Findet alle Unterverzeichnisse, aber keine sonstigen Dateien.

Name

- Mit vollständigem Namen

```
find -name hausarbeit.odt
```

Sucht die Datei **hausarbeit.odt** im aktuellen Verzeichnis und allen Unterverzeichnissen. Sind mehrere Dateien dieses Namens vorhanden, werden alle Fundstellen aufgelistet. Jokerzeichen, **Wildcards** [http://de.wikipedia.org/wiki/Wildcard_(Informatik)], wie der `*` oder das `?` müssen maskiert werden, sonst interpretiert sie schon die Shell.

-
```
find -name "*.pdf"
```

Sucht im aktuellen Verzeichnis nach PDF-Dateien. `-name` berücksichtigt die Groß-/Kleinschreibung bei der Suche, findet also mit obigen Beispiel keine Datei(en) mit Endung ***.PDF**. Das `*` steht für eine beliebige Anzahl Zeichen.

-
```
find -iname "a*.pdf"
```

Sucht im aktuellen Verzeichnis nach **.pdf**- und **.PDF**-Dateien, die mit `a` oder `A` beginnen.

-
```
find -name "katze.*"
```

findet **katze.jpg**, **katze.png**, **katze.txt** usw.

-
```
find -name "katze.??g"
```

findet **katze.jpg**, **katze.png** usw. Jedes Fragezeichen steht für ein einzelnes Zeichen.

-
```
find -name "*foo*.*x*"
```

findet **foo.x**, **afoo.x**, **foob.txt** usw.

Verzeichnis

Sucht man mit Verzeichnisnamen, in denen ein Schrägstrich '/' vorkommt, kommt man mit '-name' nicht weiter. Der Parameter '-path' ist hier die Lösung, denn er erlaubt die Interpretation des Zeichens '/'.

-
```
find -path "*2013/J*"
```

findet **~/fotos/2013/Juni** und **~/musik/2013/Juli**, aber nicht **~/dokumente/2013-Juni**.

Größe

- Maximale Größe

  ```
  find -size -100c -ls
  ```

  Nach Dateien suchen, die bis zu 100 Bytes belegen. Das - vor der Zahl bedeutet "bis zu". Das c hinter der Zahl bedeutet character, welche früher 1 Byte belegten, aber b ist schon für die Maßeinheit Block (= 512 Bytes) vergeben. Mittels -ls läßt sich die Größe der gefundenen Dateien überprüfen. Deswegen wird es hier, vor den anderen Aktionen, erwähnt.

- Exakte Größe

  ```
  find -size 100c -ls
  ```

  nach Dateien suchen, die genau 100 Bytes groß sind (kein Vorzeichen).

- Mindestgröße

  ```
  find -size +100M
  ```

  Nach Dateien suchen, die 100 Megabytes oder größer sind (Vorzeichen +). Statt M kann man auch k und G für Kilobytes und Gigabytes angeben.

- Zwischen Mindest- und Maximalgröße

  ```
  find -type f -size +64c -size -4096c | wc -l
  ```

  Nur nach Dateien suchen, die zwischen 64 und 4096 Bytes groß sind, und per Pipe an **wc** [https://wiki.ubuntuusers.de/wc/] übergeben, um die Anzahl der gefundenen Dateien zu bestimmen.

- Blockgröße

  ```
  find -size 10
  ```

  Ohne Angabe zur Maßeinheit wird die Zahl als Anzahl Blöcke interpretiert, hier also nach 10 Blöcken à 512 Bytes. Die Angabe b ist gleich bedeutend. Wenn es Probleme mit der Größe gibt, dann wahrscheinlich, weil man c oder k oder M vergessen hast.

Alter

- Änderungszeit in Tagen

  ```
  find -mtime -365
  ```

  Sucht nach Dateien, deren Inhalt innerhalb der letzten 365 Tage geändert wurde (mtime = modification time). Weitere Optionen sind
  - ctime (change time): Zeitpunkt, an dem der Status der Datei geändert wurde (Name, Rechte)
  - atime (access time): Zeitpunkt an dem auf die Datei zugegriffen wurde. Ein Dateilisting selbst ist damit nicht gemeint. Bei Bilddateien zählt die Vorschaufunktion eines grafischen Dateimanagers aber bereits als Zugriff. Auch hier gibt es, ähnlich wie bei der **Größe**, Mindestalter (+), Höchstalter (-) und genaues Alter.

- Änderungszeit in Minuten

  ```
  find -amin -5
  ```

  nach Dateien suchen, auf die in den letzten 5 Minuten zugegriffen wurde. Analog: -cmin, -mmin.

- Im Vergleich zu einer bestimmten Datei

  ```
  find -cnewer /tmp/referenz
  ```

  Nach Dateien suchen, die nach der Referenzdatei geändert wurden. Mit touch --date='15:00' /tmp/referenz erstellt man sich eine Referenzdatei, wenn man keine hat.

Sonstige Ausdrücke

- Leere Verzeichnisse und Dateien der Größe 0

  ```
  find -empty
  ```

- Verzeichnistiefe

  ```
  find -maxdepth 3
  ```

  steigt bei der Suche nur 3 Verzeichnisebenen herab.

- Alle Dateien des Benutzers BENUTZERNAME:

```
find / -user BENUTZERNAME
```

Kombinationen

- Vorgabe: Und-Kombination, die Funde müssen alle Kriterien erfüllen

```
find -mindepth 3 -maxdepth 5
```

  Finde ab Unterverzeichnis(se) 3 (mindepth 3) UND bis Unterverzeichnis(se) 5 (-maxdepth 5).

- Weiters Beispiel der UND-Kombination:

```
find -mindepth 3 -type f -name "*.avi" -size +5M
```

  Beginnt die Suche ab Unterverzeichnis(se) 3 (-mindepth 3), UND findet nur gewöhnliche Dateien (-type f), die die Endung **.avi** besitzen UND mindestens 5 MB groß sind (-size +5M).

Man kann die Suchoptionen aber auch per ODER bzw. NICHT verknüpfen:

- Negation

```
find ! -name "*.avi" -not -name "*.mp*g"
```

  Sucht Dateien die von der Dateiendung weder **avi**, noch **mpg** oder **mpeg** sind. Ausrufezeichen und -not sind gleichbedeutend.

- Oder-Kombination

```
find -name "susi.*" -or -name "susanne.*"
```

  Sucht alle Dateien die mit "susi." ODER "susanne." beginnen.

Bei umfangreichen Kombinationen kann eine Klammerung erforderlich sein, um das gewünschte Resultat zu erhalten:

- ohne Klammern

```
find -name "susi.*" -or -name "susanne.*" -name "*.txt"
```

  Ohne Klammern wird erst die UND-Verbindung gebildet, also "susanne.*" und "*.txt", danach erst ODER "susi". **susi.png** würde also gefunden.

- mit Klammern

```
find \( -name "susi.*" -or -name "susanne.*" \) -name "*.txt"
```

  Klammern müssen maskiert werden. Hier wird jetzt für alle Dateien erfordert, dass diese auf .txt enden.

Aktionen

- Ohne weitere Angaben gibt find die Namen der gefundenen Dateien aus:

```
find /boot/grub/ -name "he*"
/boot/grub/hexdump.mod
/boot/grub/hello.mod
/boot/grub/help.mod
```

- Wie bereits gesehen kann man mit -ls eine detailliertere Ausgabe erzeugen:

```
find /boot/grub/ -name "he*" -ls
168624    4 -rw-r--r--   1 root     root         3196 Jan 13 17:08 /boot/grub/hexdump.mod
168603    4 -rw-r--r--   1 root     root         1308 Jan 13 17:08 /boot/grub/hello.mod
168623    4 -rw-r--r--   1 root     root         2200 Jan 13 17:08 /boot/grub/help.mod
```

Mit -exec und dessen Varianten lassen sich beliebige Programme auf den Fundstellen ausführen.

- Die Anzahl der Zeilen in Textdateien findet man mit wc -l DATEI; kombiniert mit find sieht das so aus:

```
find -name "*.py" -exec wc -l {} \;
10 ./x/abc.py
6 ./x/date-form.py
102 ./x/download.py
```

Das Kommando wc -l (Anzahl der Zeilen zählen) wird auf jede gefundene Datei angewendet. Die geschweiften Klammern werden durch die von find gefundenen Namen ersetzt. Am Ende muss der ganze Befehl mit einem Semikolon abgeschlossen werden. Damit das Semikolon nicht von der Shell interpretiert wird, muss man es mit einem Backslash oder Anführungsstrichen maskieren.

- die Kombination mit -print

```
find tmp -name "a" -exec touch {} \; -print
./tmp/a
./tmp/a/a
./tmp/a/a/a
```

touch setzt das Datum der Dateien auf den Ausführungszeitpunkt. Da touch aber nicht den Dateinamen ausgibt sieht man nicht, welche Dateien nun betroffen waren. Daher schickt man ein -print hinterher.

- Anstatt -exec kann man auch -ok verwenden. Hierbei wird jedes mal gefragt, ob man die Aktion ausführen möchte.

- Meist empfiehlt sich -execdir statt -exec

```
find test -type d -exec tar -cjf archiv.bz2 {} \;
```

-execdir führt das Kommando aus dem Verzeichnis heraus aus, in dem die Datei gefunden wird. So wird also für jedes Unterverzeichnis ein **archiv.bz2** vor Ort angelegt. Mit einem einfachen -exec würde für jedes Verzeichnis ein Archiv im aktuellen Verzeichnis angelegt, d.h. das Archiv immer wieder überschrieben, so dass am Ende nur ein Archiv mit den Ergebnissen des letzten Verzeichnisses existiert.

- -okdir

```
find -name "*pdf" -okdir xpdf {} \;
```

-okdir fragt im Gegensatz zu -execdir vor jeder Datei nach, ob man wirklich die Aktion ausführen möchte.

- Parallele Ausführung mit +

```
find -name "*pdf" -execdir md5sum {} +
```

Beendet man ein Kommando mit Plus + statt mit Semikolon ;, so werden mehrere, u.U. alle Funde auf einen Rutsch an das Kommando übergeben. Dies ist dann sinnvoll, wenn das Kommando selbst mit mehreren Parametern zurechtkommt. Beispiele:

```
find test -type f -execdir md5sum {} ";"
```

ergibt:

```
md5sum a
md5sum b
md5sum c
```

Dagegen ergibt:

```
find test -type f -execdir md5sum {} +
```

```
md5sum a b c
```

Das + kann nur verwendet werden, wenn die geschweiften Klammern unmittelbar davor stehen.

- Eine etwas heikle Angelegenheit ist das Löschen mit der Option -delete.

> **Achtung!**
>
> Da find auch Unterverzeichnisse durchsucht, sollte mit dieser Option vorsichtig umgegangen werden. Mit find gelöschte Dateien landen nicht im Papierkorb und können nicht wieder hergestellt werden. Siehe auch **Die Aktion -delete steht an der falschen Stelle**

Vor der Verwendung sollte ein Test ohne -delete voraus gehen, um sicher zu gehen, nicht zu viele Dateien zu löschen. Die -delete-Option impliziert -depth, d.h. man muss zum Testen auch -depth setzen, um keine Überraschung zu erleben. Es ist auch sorgfältig darauf zu achten, an welcher Position -delete steht.

```
find test/ -name "c*" -delete
```

Löscht im Verzeichnis **test** und dessen Unterverzeichnissen alle Dateien, die mit "c" beginnen. Der Befehl löscht auch Verzeichnisse selbst, die mit "c" beginnen, diese jedoch nur, wenn sie leer sind, wie allgemein üblich bei Linux. Das ist der Grund, weshalb -delete ein -depth impliziert: Wenn erst in den Unterverzeichnissen gelöscht wird kann ein leeres Oberverzeichnis auch gelöscht werden, umgekehrt nicht.

Weitere Informationen

Alternativen

find ist fast immer das Mittel der Wahl, wenn es darum geht, auch Unterverzeichnisse zu durchsuchen. Wenn man den Dateinamen genau kennt, kann **locate** [https://wiki.ubuntuusers.de/locate/] eine bessere Wahl sein. locate arbeitet allerdings mit einem Index, der 1x täglich aktualisiert wird, und findet daher ganz frische Dateien nicht.

Für die spezielle Suche nach Programmen wird man whereis benutzen.

Im aktuellen Verzeichnis, ohne Unterverzeichnisse zu berücksichtigen, kommt man mit **automatischer Vervollständigung** [https://wiki.ubuntuusers.de/Terminal/#Vervollstaendigen-lassen-nicht-tippen] und den Jokerzeichen * und ? oft weiter, wenn man Namensbestandteile kennt. Manche Programme bieten auch von sich aus an, Unterverzeichnisse zu berücksichtigen, siehe **ls** [https://wiki.ubuntuusers.de/ls/] und **Shell/grep** [https://wiki.ubuntuusers.de/Shell/grep/].

Typische Fehler

find ohne Ende

Wenn die Suche mit find läuft und viel zu viele Ergebnisse ausspuckt und nicht aufhören will, so bricht man find mit Strg + C ab.

Der Pfad muss vor dem Suchkriterium stehen

Wenn man den Stern * nicht maskiert kommt es oft zu folgender Meldung:

```
find /tmp -name *sh
find: Der Pfad muß vor dem Suchkriterium stehen: adhoc.sh
Aufruf: find [-H] [-L] [-P] [-Olevel] [-D help|tree|search|stat|rates|opt|exec] [Pfad...]
[Suchkriterium]
```

Mit

```
find /tmp -name "*sh"
```

ist das leicht geheilt.

Seltsame Größen

Bei der Suche nach Dateigrößen kann man leicht verzweifeln, wenn man nicht dahinter kommt, dass die Vorgabemaßeinheit Blöcke zu 512 Bytes sind.

```
find -size 200c
```

sucht nach Größen, die man vom Dezimalsystem her erwartet.

Seltsames Nichtfinden bei Größen wie k, M, G

Sucht man nach Dateien, die kleiner sind als 1000k, so werden Dateien bis maximal 999k gefunden:

```
find -size -1000k
```

Das klingt zunächst plausibel, aber es wird keine Datei gefunden, die 999001 Bytes groß ist, denn es wird erst aufgerundet (auf 1000k) und dann verglichen (nicht kleiner als 1000k). Krasser noch, wenn man Dateien bis 1M suchen wollte - selbst 1 Byte ist größer als die nächstkleinere Ganzzahl in dieser Maßeinheit, also größer als 0M, und wird daher nicht gefunden. Das ist nicht sehr intuitiv, also Obacht.

Kombination von UND und ODER ergibt unerwartetes

Bei der Kombination von mehreren Optionen mit UND und ODER helfen Klammern Fehler zu vermeiden.

Positionssensitiv

Bei mehreren Optionen und Ausdrücken (options und expressions) unterscheiden sich erstere von zweiteren dadurch, dass Optionen nicht

mit 'ODER' gruppiert werden können - die Optionen werden immer für die ganze Suche verwendet. Stehen die Optionen hinter Ausdrücken, so sieht das aus, als habe der User eine andere Absicht gehabt, und man bekommt eine Warnung:

Folgende Meldung erhält man, wenn man Optionen nach Argumenten benutzt.

```
find tmp -name "a" -maxdepth 3 -mindepth 3
find: Warnung: Sie haben die Option `-maxdepth` nach dem Argument -name angegeben, aber Optionen sind nicht positionssensitiv (`-maxdepth` beeinträchtigt sowohl Tests, die vor ihr als auch nach ihr definiert sind). Diese Optionen ist vor den anderen Argumenten anzugeben.
```

Provozieren kann man die Warnung etwa so:

```
find ./suchverzeichnis -maxdepth 4 -name foo -or -maxdepth 2
```

xargs und Schleifen

Oft findet man Konstruktionen mit find ... xargs oder Shellschleifen die find bemühen. Fast immer lässt sich das Problem durch eine der **Aktionen** (-okdir, -execdir, ...) eleganter lösen.

Die Aktion -delete steht an der falschen Stelle

So löscht z.B. der folgende Aufruf den kompletten Inhalt des Ordners **/home/otto/**:

```
find /home/otto/ -delete -name Cache
```

Links

- **Dateisuche mit find** [http://www.easylinux.de/2004/01/072-find/] - Artikel von EasyLinux mit weiteren Beispielen, 01/2004
- **Shell/Befehlsübersicht** [https://wiki.ubuntuusers.de/Shell/Befehls%C3%BCbersicht/]

Diese Revision [https://wiki.ubuntuusers.de/find/a/revision/790998/] wurde am 14. Februar 2015 12:26 von **wxpte** erstellt.
Die folgenden Schlagworte wurden dem Artikel zugewiesen: **Shell** [https://wiki.ubuntuusers.de/wiki/tags/Shell/]