Linux, Knoppix, Mac OS X, Open Source: Vorteile von Unix et al. in Chemie & Biologie

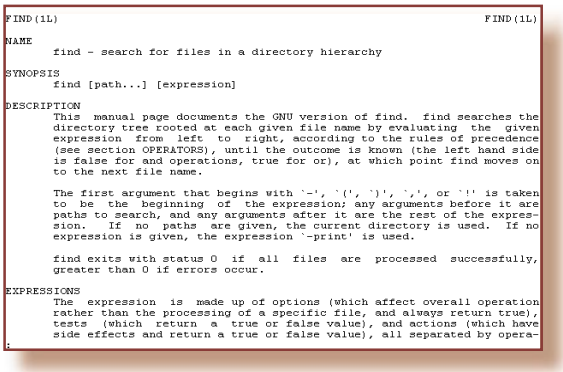
Teil 3: Hilfe & Jonglieren mit Dateien

Röbbe Wünschiers

Liebe Freunde von Unix, seiner Derivate und Leser dieser Zeilen: Bevor wir weiter in der Welt der Kommandozeile versinken, möchte ich Ihnen ein gesundes und frohes neues Jahr wünschen. Wenn Sie die Serie bis hierhin verfolgt haben, dann wissen Sie bereits wie man sich in Unix ein- und ausloggt, was Dateien, Verzeichnisse und die Shell sind und wie man mit ihnen umgeht. Sie erinnern sich, dass wir ausschließlich in der Bash Shell arbeiten werden. In dieser Ausgabe werden wir lernen, wie man zu einzelnen Befehlen Hilfe erhalten kann, wie man Dateien und Verzeichnissen sucht und was es mit Wildcards auf sich hat.

Hilfe

Viele Befehle lassen sich mit bestimmten Optionen aufrufen. Beispielsweise listet uns der Befehl `ls` (*list*) den Inhalt eines Verzeichnisses (Ordners) auf, während `ls` mit der Option `-l`, also `ls -l`, zusätzlich die Dateiattribute anzeigt (siehe Abbildung 5 in CLB 12/03). Doch dies ist nur eine Option des `ls` Befehls. Er verfügt noch über eine Vielzahl mehr, die man sich entweder merken oder aber nachschlagen muss. Wirklich? Nein, es gibt eine Hilfestellung die uns das Leben erheblich vereinfacht: die *Manual Pages*. Jedes ordentlich installierte Unix (Linux, Mac OS X, usw.) System beinhaltet das Nachschlagewerk, das mit dem Befehl `man` aktiviert wird. Mit dem folgenden Kommando können wir uns also die Hilfeseiten zu dem Befehl `ls` anzeigen lassen: `man ls`. Das Ergebnis ist in Abbildung 1 zu sehen. Natürlich wird Ihr Bildschirm nicht ausreichen, um den gesamten Inhalt der Hilfeseite darzustellen. Sie können sich jedoch mit Hilfe der Pfeil- (↑ und ↓) und Seitentasten (PgDn und PgUp) durch das Dokument bewegen. Um zurück in die Kommandozeile zu gelangen müssen Sie Q drücken. Probieren Sie es aus! Die Hilfeseiten erläutern recht ausführlich sowohl die Funktionsweise eines Befehls als auch die Bedeutung der möglichen Optionen. Doch nun zu unserem eigentlichen Thema, der Suche nach Dateien.



```
FIND(1L)                                                    FIND(1L)

NAME
       find - search for files in a directory hierarchy

SYNOPSIS
       find [path...] [expression]

DESCRIPTION
       This  manual page documents the GNU version of find.  find searches the
       directory tree rooted at each given file name by evaluating  the  given
       expression  from  left  to  right, according to the rules of precedence
       (see section OPERATORS), until the outcome is known (the left hand side
       is false for and operations, true for or), at which point find moves on
       to the next file name.

       The first argument that begins with `-', `(', `)', `,', or `!' is taken
       to  be  the  beginning  of  the expression; any arguments before it are
       paths to search, and any arguments after it are the rest of the expres-
       sion.   If  no  paths  are given, the current directory is used.  If no
       expression is given, the expression `-print' is used.

       find exits with status 0  if  all  files  are  processed  successfully,
       greater than 0 if errors occur.

EXPRESSIONS
       The  expression  is  made up of options (which affect overall operation
       rather than the processing of a specific file, and always return true),
       tests  (which  return  a  true or false value), and actions (which have
       side effects and return a true or false value), all separated by opera-
:
```

Abbildung 1: Anzeige der Hilfeseite zu dem `ls` Kommando mit `man ls`.

Wer suchet...

Wenn wir erstmal ein paar Wochen mit Unix gearbeitet haben, dann werden Sie wahrscheinlich eine Reihe von Verzeichnissen und Dateien erstellt haben. Obwohl, oder gerade weil, unter Unix Dateinamen bis zu 255 Zeichen lang sein dürfen, verliert man irgendwann die Übersicht. Dann sind Methoden gefragt, um Dateien oder Verzeichnisse effizient zu suchen. Dazu dient der Befehl **find**. **find** geht immer von einem angegebenen Verzeichnis aus und sucht in diesem und allen Unterverzeichnissen. Es würde also **find .** im aktuellen und allen Unterverzeichnissen und **find /etc** im Verzeichnis `/etc` und allen seinen Unterverzeichnissen suchen. Neben dem Suchort müssen wir noch spezifizieren wonach gesucht werden soll – ansonsten werden einfach alle Dateien des Suchortes mit ihrem Pfad aufgelistet. Als Suchkriterien bietet **find** über 10 verschiedene Optionen an. In Terminal 1 gebrauchen wir nur kleine Auswahl dieser. Die Manualseiten (**man find**) bieten Ihnen aber die Möglichkeit, sich alle Optionen einmal genauer anzusehen. Nun aber zu unserem ersten Beispiel.

Terminal 1

```
01        $ mkdir find-test
02        $ mkdir find-test/sub
03        $ date>find-test/1
04        $ date>find-test/2
05        $ date>find-test/sub/3.txt
06        $ find find-test -type f -
          size -6 -name "[0-9]"
07        find-test/1
08        find-test/2
09        $ find find-test -type f -
          size -6 -name "[0-9]*"
10        find-test/1
11        find-test/2
12        find-test/sub/3.txt
13        $ find find-test -type f -
          size -6c -name "[0-9]*"
14        $ find find-test -type f -
          name "[0-9]*" -name "*.txt"
15        find-test/sub/3.txt
16        $ mkdir find-found
17        $ find find-test -type f -
          name "[0-9]*" -name
          "*.txt" -exec cp {} find-
          found \;
```

```
18        $ ls find-found/
19        3.txt
20        $ find find-test -ctime -1
21        find-test
22        find-test/sub
23        find-test/sub/3.txt
24        find-test/1
25        find-test/2
26        $
```

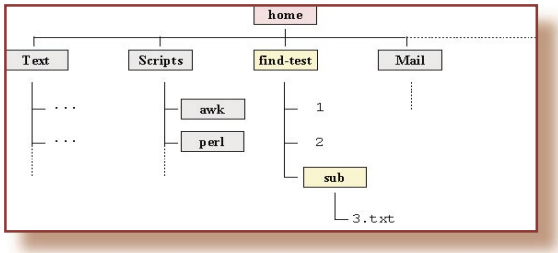


Abbildung 2: Das Suchschema von find. Unsere Testumgebung ist gelb unterlegt.

Das Beispiel in Terminal 1 ist recht umfangreich, vermittelt aber alle wesentlichen Funktionen von **find**. In den Zeilen 1-5 erstellen wir unsere „Testumgebung“ (Abbildung 2). Zunächst erstellen wir mit Hilfe des Befehls **mkdir** die Verzeichnisse *find-test* und *sub*, wobei *sub* ein Unterverzeichnis von *find-test* ist. Dann lenken wir die Ausgabe des Befehls **date** mit dem Befehl **>** (*redirect*) in eine Datei um. Auf diese Weise erzeugen wir drei Dateien, deren Inhalt die Ausgabe des Befehls **date** ist. Die Dateien mit den Namen *1* und *2* werden in dem Verzeichnis *find-test* und die Datei mit dem Namen *3.txt* in dem Verzeichnis *sub* erzeugt. Jetzt wird gesucht. In Zeile 6 suchen wir mit dem Befehl **find** in dem Verzeichnis *find-test* und allen Unterverzeichnissen nach Dateien (**-type f**), die kleiner als 3072 Byte sind (**-size -6**, 6x512=3072) und deren Name einer Zahl von 0-9 entspricht (**-name "[0-9]"**) (siehe auch Abbildung 3). Das Ergebnis ist in den Zeilen 7 und 8 zu sehen. Wieso entspricht [0-9] einer Zahl zwischen 0-9, höre ich Sie fragen. Nun, hier nutzen wir so genannte Wildcards (Joker). Die eckigen Klammern samt Inhalt stehen für genau ein Zeichen – welches Zeichen, das wird durch den Inhalt bestimmt. In unserem Fall handelt es sich um eine Zahl zwischen 0 und 9. Der Stern (*) ist ebenfalls ein Joker, oder besser gesagt: **der** Joker. Er steht für ein, kein oder beliebig viele Zeichen (mehr zu Wildcards erfahren sich weiter unten). Den Stern wenden wir in Zeile 9 in Terminal 1 an.

Hier suchen wir also nach einer Datei, die kleiner als 3072 Byte ist und deren Name mit einer Zahl beginnt und beliebig endet (**-name "[0-9]*"**). Wie wir in den Zeilen 10-12 sehen, trifft dieses Suchkriterium auf alle 3 Dateien zu. Zeile 13 unterscheidet sich von der Zeile 9 nur durch die Angabe der Dateigröße: jetzt suchen wir eine Datei, die weniger als 6 Zeichen enthält. Dazu hängen wir der 6 einfach ein **c** (*character*, Zeichen) an. Da die Ausgabe des **date** Befehls von der Form „*Tue Jan 13 21:23:35 CET 2004*“ ist, enthalten alle drei Dateien mehr als 6 Zeichen – daher erhalten wir kein Suchergebnis. Zeile 14 zeigt, dass Suchkriterien, hier **-name**, auch mehrfach auftreten dürfen. Hier suchen wir nach Dateien deren Name mit einer Zahl beginnt und auf *.txt* endet (natürlich könnten wir auch schreiben: **-name "[0-9]*.txt"**). Dies trifft in unserer Testumgebung nur auf die Datei *3.txt* zu. In Zeile 16 erstellen wir das Verzeichnis *find-found*. In Zeile 17 wird es spannend – hier werden wir sehen, wie mächtig das **find** Kommando ist. Das Suchkriterium ist das gleiche wie in Zeile 14. Anstatt aber das Suchergebnis auf den Bildschirm auszugeben, wenden wir mit der Option **-exec cp {} find-found \;** den Befehl **cp** (*copy*, kopieren) an. Die geschwungenen Klammern repräsentieren die Dateien, auf die das Suchkriterium zutrifft. Sie werden in das Verzeichnis *find-found* kopiert. Wie in Abbildung 3 zu sehen ist, beendet der Backslash (\) mit dem Semikolon die **-exec** (*execute*, ausführen) Option. In Zeile 18 lassen wir uns den Inhalt des Verzeichnisses *find-found* anzeigen – und tatsächlich, sie enthält die Datei *3.txt*. Möchte man eine Datei nicht kopieren sondern verschieben, so muss man den Befehl **mv** (*move*, verschieben) anwenden. Wie bei **cp** wird bei **mv** die Option **-r** (*recursively*, rekursiv) benötigt, wenn ein Verzeichnis samt Inhalt verschoben werden soll. Zu guter Letzt suchen wir in Zeile 20 in Terminal 1 nach Verzeichnissen oder Dateien die zuletzt vor einem Tag oder früher (**-ctime -1**) erstellt bzw. geändert wurden. Wenn Sie nicht eine ausgedehnte Pause gemacht haben, dann sollte dies auf alle Verzeichnisse und Dateien in *find-test* zutreffen. Übrigens eignet sich die Option **-ctime** (*change time*, Änderungszeit) in Verbindung mit der Option **-exec** um am Ende des Tages Sicherungskopien aller bearbeiteten Dateien zu machen.

Wildcards

Nachdem wir in Terminal 1 schon Wildcards angewandt haben, wollen wir jetzt einen detaillierten Blick auf diese überaus wichtigen Kürzel werfen. Wenn Unix ein Kartenspiel wäre, dann wären Wildcards die Joker. Allerdings gibt es unterschiedliche Joker, die in Abbildung 4 zusammengefasst sind. In Terminal 1 haben wir bereits den Stern und die eckigen Klammern kennen gelernt. Der Umgang mit Wildcards kann manchmal etwas verwirrend sein. Es gibt aber einen einfachen Weg, um die entworfene Kombination aus Wildcards zu überprü-

Abbildung 3: Einige Optionen des find Kommandos.

Option	Bedeutung
-name "*muster*"	Sucht nach Dateien und Verzeichnissen die *muster* entsprechen.
-type *zeichen*	Mit *zeichen* läßt sich die Suche auf Dateien (f) oder Verzeichnisse (d) beschränken.
-size *größe*	Mit *größe* läßt sich die Suche nach Dateien die größer (+*größe*), kleiner (–*größe*) oder gleich (*größe*) der angegebenen Größe sind einschränken. Die Größe wird in 512 Byte (0,5 kB) Blöcken angegeben und aufgerunded. Mit -size +5 werden also Dateien gesucht die größer als 2560 Byte groß sind.
-ctime *nummer*	Sucht Dateien die vor *nummer* Tagen geändert wurden. Durch voranstellen des Plus- (+) oder Minuszeichens (-) können Dateien gesucht werden die vor oder nach *nummer* Tagen geändert wurden.
-exec *befehl* {} \;	Auf alle Dateien die dem Suchkriterium entsprechen wird das Kommando *befehl* angewendet.

Wildcard	Bedeutung
?	Genau ein beliebiges Zeichen.
*	Kein, ein oder beliebig viele beliebige Zeichen.
[abc14]	Eines der angegebenen Zeichen.
[A-Za-z]	Ein beliebiges Zeichen aus dem angegebenen Bereich.
[!0-9abc]	Keines der angegebenen Zeichen.

Abbildung 4: Wildcards.

fen, bevor man irgendwelchen Schaden anrichtet. Dies geschieht mit Hilfe des Befehls `echo`.

Terminal 2

```
01        cd find-test
02        $ echo *
03        1 2 sub
04        $ echo [a-z]
05        [a-z]
06        $ echo [0-9]
07        1 2
08        $ ls *
09        1 2
10
11        sub:
12        3.txt
13        $ ls *.txt
14        ls: *.txt: No such file or
          directory
```

Wie Sie später bei der Shellprogrammierung sehen werden, dient `echo` eigentlich der Ausgabe von Text auf den Bildschirm. Werden aber Wildcards verwendet, dann gibt `echo` alle Dateien im aktuellen Verzeichnis zurück, die auf die angegebene Wildcard-Kombination passen. Passt keine Datei, dann wird einfach das Wildcardmuster zurück gegeben. Dies ist z.B. in der Zeile 5 zu sehen. Verwenden Sie den Stern zusammen mit dem `ls` Befehl, dann werden alle Dateien im aktuellen Verzeichnis und in allen Unterverzeichnissen angezeigt. Die Ausgabe kann unter Umständen also recht lang werden. Der `ls` Befehl in Zeile 13 dagegen durchsucht nicht alle Unterverzeichnisse nach Dateien, die auf *.txt* enden. Für die Suche nach Dateien oder Verzeichnissen sollten Sie daher immer das `find` Kommando verwenden.

Kopieren reloaded

Lassen Sie uns zum Schluss noch einmal einen kurzen Blick auf das Kopieren bzw. Verschieben von Dateien werfen. Mit dem Befehl `cp ../datei .` kopiert man die Datei namens *datei* aus dem Überverzeichnis (repräsentiert durch die zwei Punkte) in das aktuelle Verzeichnis (repräsentiert durch den Punkt) (siehe auch Abbildung 6 in CLB 12/03). Probieren wir das gleich aus.

Terminal 3

```
01        $ pwd
02        /Users/rw/find-test/sub
03        $ ls
04        3.txt
05        $ cp ../* .
06        cp: ../sub is a directory
          (not copied).
07        $ ls
08        1     2     3.txt
09        $ mv * ..
10        $ ls
11        $ cd ..
12        $ ls
13        1     2     3.txt sub
14        $ cd
15        $ rm -r find-test fin
16        find-found       find-test
          fink-readme.rtf
17        $ rm -r find-test find-found
```

In Terminal 3 stellen wir mit dem Befehl `pwd` (*print working directory*) zunächst sicher, dass wir uns im Verzeichnis *find-test/sub* befinden, das wir in Terminal 1 erstellt haben (Abbildung 2). Dort befindet sich nur die Datei *3.txt*. In Zeile 5 kopieren wir alle Dateien (`*`) aus dem übergeordneten Verzeichnis (`..`) in das aktuelle Verzeichnis (`.`). Dabei erhalten wir eine Fehlermeldung, denn das Verzeichnis *sub* (in dem wir uns befinden) kann nicht kopiert werden. Schauen wir einmal nach, ob alles geklappt hat. Das machen wir in Zeile 7 mit dem `ls` Befehl. Tatsächlich, jetzt sind alle Dateien im Verzeichnis *sub*. In Zeile 9 verschieben wir alle Dateien in das übergeordnete Verzeichnis. Das Ergebnis des `ls` Kommandos in Zeile 10 zeigt, dass wir erfolgreich waren: alle Dateien sind weg. Nun wechseln wir in das übergeordnete Verzeichnis (`cd ..`) und listen dessen Inhalt auf (`ls`). Da sind die Dateien wieder. In Zeile 14 wechseln wir in unser Home-Verzeichnis (`cd` ohne Angabe eines Verzeichnisses wechselt `cd` immer in das Home-Verzeichnis). Von dort aus löschen wir mit dem Befehl `rm -r` (*remove* mit der Option *recursively*) in Zeile 17 unsere Testumgebungen (die Verzeichnisse *find-test* und *find-found*), die wir in Terminal 1 erstellt haben.

Neue in dieser Ausgabe behandelte Befehle

`man`	Hilfeseiten zu Befehlen anzeigen
`find`	Dateien und Verzeichnisse suchen
`mv`	Dateien oder Verzeichnisse verschieben
`echo`	Test von Wildcards
`x > y`	Umleitung der Ausgabe des Befehls `x` in die Datei `y`.

Sie wissen nun wie mit einfachen Befehlen Dateien erzeugt, kopiert, verschoben, gesucht und gelöscht werden. Sie sollten sich die Zeit nehmen und mehrere Dateien mit verschiedenen Namen in verschiedenen Verzeichnissen generieren und Ihre Kenntnisse praktisch vertiefen. Insbesondere sollten Sie sich mit den Wildcards vertraut machen. In der kommenden Ausgabe werden wir lernen, wie Text-Dokumente erstellt, editiert, angesehen und durchsucht werden können.