1.8 Anwendung/Suchen und Finden im Dateisystem

1.8.1 Suche nach ausführbaren Dateien

Wenn man ein Programm starten will und dazu nur den Namen der Programmdatei eingibt (ohne Verzeichnis), dann muss das System wissen, wo es nach der Programmdatei suchen soll. Dazu gibt es die Variable `PATH`. Mit der Variablen `PATH` wird der Shell mitgeteilt,

- in welchen Verzeichnissen sie nach ausführbaren Dateien suchen soll
- und in welcher Reihenfolge.

Die einzelnen Einträge werden dabei durch Doppelpunkt getrennt:

Terminal
```
schueler@debian964:~$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/sbin:/usr/sbin
schueler@debian964:~$ PATH=$PATH:/usr/games
schueler@debian964:~$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/sbin:/usr/sbin:/usr/games
schueler@debian964:~$ PATH=${PATH/:\/usr\/games:/:}
schueler@debian964:~$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/sbin:/usr/sbin
```

Die Befehle `which` und `type` suchen in den Verzeichnissen der Suchpfadliste nach einem Programm, führen es aber nicht aus. Stattdessen zeigen sie an, wo sie es zuerst gefunden haben.

Terminal
```
schueler@debian964:~$ which cal
/usr/bin/cal
schueler@debian964:~$ type cal
cal ist /usr/bin/cal
```

`which` gibt nur kurz den Pfad aus oder aber nichts, wenn kein Programm dieses Namens gefunden wurde. `which` kann man also gut für Skripten verwenden, in denen eine bestimmte Programmdatei gesucht wird.

`type` dagegen wendet sich mit seiner ausführlicheren Ausgabe an den menschlichen Nutzer. Es funktioniert auch für Aliase, eingebaute Shell-Befehle und Shell-Funktionen:

Terminal
```
schueler@debian964:~$ type ll
ll ist ein Alias von 'ls -l'.
schueler@debian964:~$ type cd
cd is a shell builtin
schueler@debian964:~$ type if
if Ist ein reserviertes Schlüsselwort der Shell.
schueler@debian964:~$ type quote
quote ist eine Funktion.
```

Befehle, die auf andere Befehle gelinkt sind, findet man mit einer Kombination aus `which` und `ls`:

Terminal
```
schueler@debian964:~$ ls -l $(which vi)
lrwxrwxrwx 1 root root 20 Okt 1 2023 /usr/bin/vi->/etc/alternatives/vi
```

Die hier benutzte Konstruktion heißt Kommandosubstitution.

1.8.2 Suche nach Hilfe

Bekanntlich kann man mit dem Befehl `man` Hilfe bekommen, Hilfe zu Programmen, Dateien und Konzepten unter Linux. Hinter dem Befehl `man` befindet sich ein durchdachtes System von Hilfe-Dateien. Sie sind aufgeteilt in mehrere Abschnitte, die an die Handbuch-Kapitel der ursprünglichen UNIX-Handbücher angelehnt sind. Die Übersicht erhält man mit `man man`:

Terminal

```
schueler@debian964:~$ man man
MAN(1)           Dienstprogramme für Handbuchseiten           MAN(1)

BEZEICHNUNG
     man - eine Oberfläche für die Online-Referenzhandbücher

ÜBERSICHT
     ...

BESCHREIBUNG
     ...
     1   Ausführbare Programme oder Shell-Befehle
     2   Systemaufrufe (Kernel-Funktionen)
     3   Bibliotheksaufrufe (Funktionen in Programmbibliotheken)
     4   Spezielle Dateien (gewöhnlich in /dev)
     5   Dateiformate und Konventionen, z. B. /etc/passwd
     6   Spiele
     7   Verschiedenes (einschl. Makropaketen und Konventionen), ...
     8   Befehle für die Systemverwaltung (in der Regel nur für root)
     9   Kernel-Routinen [nicht Standard]
     ...
```

Nun kann es vorkommen, dass es zu einem Namen mehrere Handbuchseiten gibt. Dann kann man sie mit dem Befehl `whatis` auflisten lassen:

Terminal

```
schueler@debian964:~$ whatis signal
signal (7)           - Überblick über Signale (Software-Interrupts)
signal (2)           - Handhabung von Signalen in ANSI C
signal (3posix)      - signal management
```

Hier erhält man statt der ganzen Manual-Seite nur die Überschriftszeile zu sehen (entspricht dem Abschnitt BEZEICHNUNG in der Seite). Nun kann man sich aussuchen, ob man eine der Seiten ansehen möchte: Mit `man 2 signal` bekommt man nur die Seite aus Kapitel 2 zu sehen. Ist man sich nicht sicher, wählt man `man -a signal` und kann nacheinander jede der drei Seiten anschauen.

Während `whatis` genau nach einem Befehlsnamen sucht[1], kann man mit `apropos` in den Überschriftszeilen aller Seiten nach einem Wort fahnden:

Terminal

```
schueler@debian964:~$ apropos dhcp
dhclient-script (8)  - DHCP client network configuration script
dhclient.conf (5)    - DHCP client configuration file
dhclient.leases (5)  - DHCP client lease database
dhcp-eval (5)        - ISC DHCP conditional evaluation
dhcp-options (5)     - Dynamic Host Configuration Protocol options
dnsmasq (8)          - A lightweight DHCP and caching DNS server.
```

[1] allerdings nicht case-sensitiv!

Die Hilfe-Dateien werden in der Regel nicht so gespeichert wie sie angezeigt werden. Stattdessen sind sie in einer an HTML erinnernden einfachen Auszeichnungssprache abgelegt, dazu noch komprimiert. Mit dem Befehl `whereis` erhält man die Pfadnamen der Hilfe-Dateien [2]:

Terminal
```
schueler@debian964:~$ whereis signal
signal: /usr/include/signal.h /usr/share/man/man3/signal.3posix.gz
/usr/share/man/man7/signal.7.gz /usr/share/man/man9/signal.9freebsd.gz
/usr/share/man/man2/signal.2.gz
```

Woher wissen nun `man` und die anderen Programme des Hilfesystems, wo sie nach den Seiten suchen sollen? Schließlich soll das Hilfesystem ja flexibel anpassbar und erweiterbar sein.

Anders als bei der Suche nach Programmdateien verlässt sich das Hilfesystem nicht allein auf eine Umgebungsvariable, sondern auf ein eigenes Programm `manpath`, welches eine Suchpfadliste bestimmt:

Terminal
```
schueler@debian964:~$ manpath
/usr/local/man:/usr/local/share/man:/usr/share/man
```

Dieses Programm holt sich seine Konfiguration aus der Datei `/etc/manpath.config`. Anschließend wird die Pfadliste um den Inhalt von `$MANPATH` ergänzt oder ersetzt; siehe dazu die folgenden Beispiele:

Terminal
```
schueler@debian964:~$ MANPATH=":/tmp"
schueler@debian964:~$ manpath
/usr/local/man:/usr/local/share/man:/usr/share/man:/tmp
schueler@debian964:~$ MANPATH="/tmp:"
schueler@debian964:~$ manpath
/tmp:/usr/local/man:/usr/local/share/man:/usr/share/man
schueler@debian964:~$ MANPATH="/tmp"
schueler@debian964:~$ manpath
/tmp
```

In seltenen Fällen bekommt man ein Programmpaket mit einer „rohen“ Handbuchseite. Man erkennt sie an einem Dateinamen, der mit einer Ziffer (der Kapitelnummer) endet, eventuell gefolgt von einer der Endungen `.gz` oder `.Z`. Mit der Option `-l` kann `man` so eine Datei direkt anzeigen, auch wenn sie nicht im richtigen Verzeichnis liegt:

Terminal
```
schueler@debian964:~$ whereis ls
ls: /bin/ls /usr/share/man/man1/ls.1.gz
schueler@debian964:~$ man -l /usr/share/man/man1/ls.1.gz
...
```

1.8.3 Suche nach Dateien und Verzeichnissen

1.8.3.1 `find` Mit dem `find`-Befehl steht ein Werkzeug zur Suche von Dateisystemobjekten zur Verfügung. Es hat die Syntax:

```
find Teilbaum Kriterium|Aktion ..
```

Man kann mit `find` alle Objekte in einem Teilbaum finden, die bestimmten *Kriterien* genügen. Die Kriterien werden von links nach rechts abgearbeitet und sind mit UND verknüpft:

[2] sowie der Quelltexte (falls vorhanden) und der Programmdateien

Terminal
```
schueler@debian964:~$ find /home # findet alle Objekte
schueler@debian964:~$ find /home -type f # alle Dateien (f=file)
schueler@debian964:~$ find /home -name "*.html" #alles m.Name "*.html"
schueler@debian964:~$ find /home -type f -name "*.html" #Dat. "*.html"
```

Die Anführungszeichen hindern die Shell am eigenmächtigen Interpretieren der Wildcards. Ansonsten ist hier alles zugelassen, was die Wildcard-Expansion von Pfadnamen hergibt. Man kann Kriterien auch mit ODER verknüpfen, wenn man sie mit der Option `-o` verbindet:

Terminal
```
schueler@debian964:~$ find bu -type f -name "*.html" -o -name "*.txt"
bu/fehlt/such.txt
bu/fehlt/oszi.txt
bu/fehlt/bsp.html
```

Ebenso ist es möglich, mit der Option `-not` ein Kriterium zu negieren:

Terminal
```
schueler@debian964:~$ find aktuell -name "*.html" -o -name "*.txt" \
  -not -name "such.txt"
bu/fehlt/oszi.txt
bu/fehlt/bsp.html
```

Tabelle 1 zeigt weitere wichtige Kriterien .

Name	Bedeutung
`-type f`	Dateisystemtyp (f=file, d=directory, l=link, s=socket usw.)
`-name "*.txt"`	Namensmuster des Objekts ohne Verzeichnisanteil
`-size +100`	Größe (+100 heißt mehr als, -100 heißt weniger als, 100 heißt genau 100 Bytes)
`-ctime +20`	Alter in Tagen (hier: mehr als 20 Tage)
`-cmin +20`	Alter in Minuten (hier: mehr als 20 min)
`-newer x.txt`	Alter im Vergleich zu einem anderen Objekt
`-user heinz`	Nutzer
`-nouser`	zu keinem Nutzer mehr gehörend
`-perm -g=w,o=w`	Berechtigung, **alle** in der Liste angegebenen Bedingungsbits müssen gesetzt sein (UND-Verknüpfung)
`-perm /g=w,o=w`	Berechtigung, mindestens **ein** in der Liste angegebenes Bedingungsbit muss gesetzt sein (ODER-Verknüpfung)

Tabelle 1: Kriterien für `find`

Standardmäßig werden die Namen der gefundenen Objekte auf dem Bildschirm ausgegeben. Mit der Angabe spezieller *Aktionen* kann man das aber modifizieren. Dabei werden die Aktionen ausgeführt für alle Objekte, die die Kriterien bis hier erfüllen:

Terminal
```
schueler@debian964:~$ find /etc -type f -name "*.t*" -print \
   -name "*.txt" -quit
```

Hier werden alle Dateinamen, die auf `*.t*` passen, ausgegeben, aber nur so lange, bis eine Datei gefunden wird, die auf `*.txt` passt. Das Standardverhalten ist also so, als wenn am Schluss der Befehlszeile die Aktion `-print` stünde. Tabelle 2 zeigt einige der wichtigsten Aktionen.

Name	Bedeutung
`-print`	Ausgabe der Namen, getrennt durch ein Leerzeichen
`-print0`	Ausgabe der Namen, getrennt durch ein String-Ende-Zeichen `'\0'` (besser!)
`-fprint0 neu.dat`	Ausgabe der Namen in die Datei `neu.dat`
`-quit`	Abbruch
`-exec cp {} /tmp ';'`	Ausführen des folgenden Befehls mit dem Namen der gefundenen Datei an der Stelle des Wortes `{}` (in diesem Beispiel also `cp datei /tmp`)

Tabelle 2: Aktionen für `find`

1.8.3.2 `locate` Eine schnellere Suche von Dateisystemobjekten erfolgt mit dem Befehl `locate` — hier wird eine regelmäßig aktualisierte Datenbank (locatedb) zur Suche benutzt.

Terminal
```
schueler@debian964:~$ locate '*.html' # Alles mit diesem Namensmuster
schueler@debian964:~$ locate html     # Kurzform
```

1.8.3.3 Sonstige Für das Finden und Wiederherstellen gelöschter Dateien gibt es noch das Programm `photorec` aus dem Paket `testdisk`.

1.8.4 Suche von Texten *in* Dateien

Zur Suche von Texten *innerhalb* von Textdateien bietet sich der Befehl `grep` an (Abkürzung von *get regular expression*). `grep` gibt die gefundenen Textzeilen aus: Es hat die Syntax:

Terminal
```
grep [Optionen] Suchmuster [Datei] ..
```

Ohne Angabe von Dateinamen liest `grep` von der Standardeingabe. Hier ein Beispiel:

Terminal
```
schueler@debian964:~$ grep nfs /etc/services
nfs             2049/tcp                        # Network File System
nfs             2049/udp                        # Network File System
schueler@debian964:~$ cat /etc/services | grep nfs
nfs             2049/tcp                        # Network File System
nfs             2049/udp                        # Network File System
```

Es werden alle Zeilen gefunden, die an irgendeiner Stelle die Buchstabenkombination `nfs` aufweisen.

Mit der Option `-i` (*ignore-case*) wird auf die Unterscheidung zwischen Groß- und Kleinschreibung verzichtet, und es würden auch Zeilen mit dem Wortschnipsel `NFS` gefunden.

Die Option `-v` (*inverted search*) dreht die Suche um, und es werden alle Zeilen gesucht, die das Suchwort *nicht* enthalten. Damit kann man die Suche gut eingrenzen:

Terminal
```
schueler@debian964:~$ grep nfs /etc/services | grep -v udp
nfs             2049/tcp                        # Network File System
```

Mit der Option `-n` (*numbers*) wird die Zeilennummer zusätzlich ausgegeben:

Terminal
```
schueler@debian964:~$ grep nfs /etc/services
301:nfs             2049/tcp                    # Network File System
302:nfs             2049/udp                    # Network File System
```

Mit der Option `-l` (*list file names*) wird statt des Inhalts nur der Name der Datei, in der Fundstellen auftauchen, ausgegeben:

Terminal
```
schueler@debian964:~$ grep -l nfs /etc/* 2>/dev/null
/etc/exports
/etc/insserv.conf
/etc/rpc
/etc/services
```

Man kann also – wie im obigen Beispiel – auch in mehreren Dateien suchen lassen und dafür die Wildcard-Expansion von Pfadnamen benutzen.

Symbol	Bedeutung
`x`	Buchstabe x; normale Zeichen stehen für sich selbst
`.`	Ein Punkt (und nicht das Fragezeichen) steht für ein beliebiges Zeichen. Meint man einen Punkt selbst, muss man `\.` schreiben.
`[Uu]`	ein U oder ein u, aber nicht beide
`[0-9]`	ein Zeichen zwischen 0 und 9
`[[:upper:]]`	ein Großbuchstabe (ebenso `lower`, `digit`, `alpha`)
`[^xy]`	ein Zeichen außer x oder y
`^`	steht für den Zeilenanfang; sinnvoll, wenn man Worte am Zeilenanfang sucht
`$`	steht für das Zeilenende; sinnvoll für die Suche am Zeilenende
`\b`	steht für einen Wortzwischenraum; sinnvoll, wenn man nur ganze Worte finden will
`xx*`	ein oder mehrere x nacheinander
`x*`	kein, ein oder mehrere x nacheinander
`[x]\{2,7\}`	zwei bis sieben x nacheinander
`\(..\)`	die Klammer fasst zwei Zeichen zusammen; auf diese kann anschließend mit `\1` wieder zugegriffen werden (Rückbezug).
`(ab)|(cd)`	entweder ab oder cd
`x+`	ein oder mehrere x nacheinander
`x?`	kein oder ein x

Tabelle 3: Regeln für reguläre Ausdrücke

Man kann aber außerdem noch das Suchmuster verfeinern, um damit bessere Treffer zu erzielen. Dazu gibt es unter Unix-artigen Betriebssystemen (und in vielen Programmiersprachen) so genannte reguläre Ausdrücke (*regular expressions*). Sie ähneln der Wildcard-Expansion von Pfadnamen, haben aber ihre eigenen Regeln. Sie sind in Tabelle 3 aufgelistet[3].

Z. B. soll folgende Datei mit dem Namen `bsp.txt` durchsucht werden:

```
1 Bielefeld
2 Bremen
3 Dortmund
```

Dann passt `d$` auf Bielefeld und Dortmund, nicht aber auf Bremen; `r.*n` dagegen passt auf Bremen und Dortmund:

Terminal
```
schueler@debian964:~$ grep 'd$' bsp.txt
Bielefeld
Dortmund
```

[3]Die Symbole `x+` und `x?` in den letzten beiden Zeilen der Tabelle sind nur benutzbar, wenn man mit Nennung der Option `grep -E` die erweiterten regulären Ausdrücke benutzt. Dann entfallen auch die Backslash-Zeichen vor den runden und geschweiften Klammern.

```
schueler@debian964:~$ grep 'r.*n' bsp.txt
Bremen
Dortmund
```

Hier zwei Beispiele für den Rückbezug:

Terminal

```
schueler@debian964:~$ grep '\(.\).*\1' bsp.txt
Bielefeld
Bremen
schueler@debian964:~$ grep '\(..\).*\1' bsp.txt
Bielefeld
```

In den Worten `Bielefeld` und `Bremen` wurde der Buchstabe `e` (mindestens) zweimal gefunden. Außerdem wurde im Wort `Bielefeld` die Buchstabenkombination `el` (im Muster durch `..` gekennzeichnet) zweimal gefunden.

Wie man sieht, sind reguläre Ausdrücke nicht ganz einfach zu erlernen, aber dafür vielseitig in der Anwendung (Ausfiltern von Spam-Mails, finden von Schlüsselwörtern in eigenen Texten usw.). Insbesondere im Zusammenhang mit dem Textbearbeitungsprogramm `sed` (*stream editor*, siehe dort) kann man mit Hilfe regulärer Ausdrücke seine Daten mit wenigen Befehlen zielgenau modifizieren.