IN DIESEM KAPITEL

- Dateien, Namen und Verzeichnisse
- Dateien, ihre Besitzer und deren Rechte
- Mit Links auf die gleiche Datei verweisen
- Verzeichnisbaum statt Laufwerke
- Prozesse
- Benutzerverwaltung

Kapitel 1
Linux

Linux hat mehrere Besonderheiten, die es weitgehend von UNIX geerbt hat. Diese unterscheiden sich häufig von anderen Betriebssystemen.

Ein Blick in die Geschichte

Warum die Wahl bei Servern so oft auf Linux fällt, ließe sich durch einen intensiven Blick in die Betriebssystemarchitektur leicht erklären. Linux ist für den Mehrbenutzerbetrieb gebaut, hat also von Anbeginn an Konzepte, Benutzer anzulegen, deren Dateien von denen anderer zu trennen und im Zweifelsfall einen Fremdzugriff zu verteidigen. Anschaulicher wird es, wenn man in die Historie zurückblickt.

1970 Das Betriebssystem UNIX entstand aus der Anregung von MULTICS, einem Versuch, über Firmengrenzen hinaus ein Universalbetriebssystem zu erstellen, also eine Eier legende Wollmilchsau. Wie das immer wieder bei diesen Tieren ist, lassen sie sich letztlich doch nicht umsetzen. Die damals verfügbare Hardware schaffte es nicht. Ken Thompson und Dennis Ritchie von AT&T erstellten aus den Konzepten von MULTICS eine schlankere Version und nannten sie UNIX. Seinerzeit wurden Betriebssysteme in Maschinensprache geschrieben und waren damit abhängig von der Hardware, insbesondere der CPU. Um UNIX portabel zu machen, entwickelten sie die Programmiersprache C.

AT&T durfte zu jenem Zeitpunkt aus kartellrechtlichen Gründen keine Software verkaufen. So wurde UNIX inklusive C-Compiler der Berkeley-Universität

zur Verfügung gestellt, die auf dieser Basis eine eigene Variante von UNIX namens BSD erstellte.

Auf diesem Wege entstanden auch viele kommerzielle UNIX-Varianten. Um einen Standard zu erhalten, definierte man IEEE 1003, der später als POSIX bezeichnet wurde.

In den frühen 1980er-Jahren durfte AT&T wieder Software verkaufen und das inzwischen populäre UNIX wurde kommerzialisiert. Der Code durfte nicht mehr veröffentlicht werden. Das verärgerte viele Mitprogrammierer an UNIX, die zur Verbreitung beigetragen hatten, aber die Ergebnisse selbst nicht nutzen durften. Es entstand die GNU-Bewegung (GNU's Not Unix), die schrittweise ein freies POSIX-konformes Betriebssystem auf eigener Code-Basis entwickelte und vor allem dafür sorgte, dass die Sourcen durch eine entsprechende Open-Source-Lizenz immer frei zugänglich bleiben sollten.

1973 1973 begannen die ersten Entwicklungen zu TCP/IP. Ziel war es, eine Netzwerkverbindung über mehrere Netzwerke mit diversen Übertragungstechniken zu schaffen. Das Verteidigungsministerium der USA stieß diese Entwicklung entscheidend an. Das führte dazu, dass die Sourcen öffentlich finanziert waren und damit nach amerikanischem Recht auch öffentlich zugänglich sein mussten. 1975 wurde erstmals ein Netzwerk zwischen Stanford und der University College London eingerichtet.

1980 Für den Prozessor 8086 erstellte Tim Paterson ein Basisbetriebssystem unter dem Namen QDOS – Quick and Dirty Operating System, weil eine CP/M-Version für diesen Prozessor nicht verfügbar war.

QDOS wurde von Microsoft aufgekauft und 1981 zu MS-DOS. Es wurde Standardbetriebssystem auf dem IBM PC und Geräten, die zum IBM PC kompatibel waren. MS-DOS war ein Einbenutzerbetriebssystem, das kein Multitasking erlaubte und zunächst nicht einmal Verzeichnisse, da es für den Diskettenbetrieb geschaffen war.

1984 Der Mac erschien 1984 als erster kommerziell erfolgreicher Computer mit einer grafischen Oberfläche. Das Betriebssystem des Mac war zu diesem Zeitpunkt ein Einbenutzersystem ohne Multitasking-Fähigkeiten.

1985 Bill Gates ließ Windows als ersten Versuch einer Mac-Kopie für MS-DOS entwickeln. Windows hatte in dieser Version allerdings keinen Erfolg. Erst 1990 konnte Windows 3.0 ernsthaft am Markt Fuß fassen.

1991 Linus Torvalds veröffentlichte seinen Kernel für 80386-PCs, der später Linux genannt werden sollte und schließlich unter GNU-Lizenz stand.

Der Kernel wurde schnell mit der bereits entwickelten GNU-Umgebung kombiniert und so stand damit bald ein freies POSIX-konformes System zur Verfügung.

1992 Berkeley befreit sein BSD von lizenzrechtlich problematischen Quelltexten und stellt dieses frei zur Verfügung.

TCP/IP konnte sowohl für BSD als auch für Linux frei implementiert werden.

1993 Microsoft positioniert Windows NT als Konkurrenz zu Novell NetWare. NetWare stellte einen Dateiserver auf der Basis eines eigenen Netzwerk-Protokolls dar. NT lieferte außerdem erstmalig TCP/IP standardmäßig für eine Windows-Version aus.

1998 Steve Jobs kehrte zu Apple zurück. Er hatte nach seinem Ausscheiden bei Apple 1985 die Firma NeXT gegründet. NeXT stellte High-Level-Computer mit einem hochmodernen Betriebssystem auf der Basis einer UNIX-Variante her. Nachdem NeXT von Apple aufgekauft wurde, wurde Mac OS X als Betriebssystem des Macs herausgegeben, welches auf NeXTStep und BSD-UNIX basierte.

Aufgrund der freien Lizenz von Linux findet man dieses inzwischen in allen möglichen Geräten wie Routern, Fernsehern, Autos, Smartphones oder Kleingeräten des Internet of Things (IoT).

Dateien und Verzeichnisse

Wie andere Betriebssysteme auch verwendet Linux Dateien und ordnet sie in Verzeichnissen an. Interessant sind die Eigenschaften, die bereits vollständig auf ein Mehrbenutzersystem vorbereitet sind.

Als Administrator werden Sie meist vom Terminal aus operieren. Darum lernen Sie hier gleichzeitig die grundlegenden Befehle und die Besonderheiten des Dateisystems kennen.

Damit Sie die Beispiele ausführen können, benötigen Sie also ein Terminal. Dazu gibt es mehrere Wege:

- ✔ Wenn Ihr Linux keine grafische Oberfläche hat, werden Sie nach dem Einloggen automatisch im Terminal landen.
- ✔ Die grafischen Oberflächen bieten Programme an, die eine Terminalsitzung in einem Fenster starten können. Diese Programme tragen das Wort »Terminal« im Namen und sind meist der Gruppe Systemwerkzeuge zugeordnet.
- ✔ Mit der Tastenkombination Strg+Alt+L verschwindet Ihre grafische Oberfläche. Sie können sich unter Ihrer Kennung an einem Terminal anmelden. Wenn Sie fertig sind, loggen Sie sich mit dem Befehl `exit` wieder aus und schalten mit der Tastenkombination Strg+Alt+F7 wieder zurück in Ihren grafischen Desktop.

Betrachten der Dateien

Mit dem Befehl `ls` können Sie sich Dateien und Verzeichnisse im aktuellen Verzeichnis anschauen.

Optionen

Optionen beginnen immer mit einem Minuszeichen und bestehen aus einem Buchstaben. Es gibt allerdings auch Optionen mit zwei Minuszeichen. Dahinter dürfen dann auch ganze Wörter als Option stehen. Das ist anschaulicher, aber auch anstrengender zu tippen, steigert also den Kalorienverbrauch.

Argumente und Wildcards

Sie können neben den Optionen auch Argumente angeben. Das wären dann beispielsweise die Dateien, die Sie einzeln betrachten wollen. Wenn Sie mehrere Dateien anschauen wollen, können Sie einen Stern als Platzhalter verwenden. `M*us` steht für alle Namen, die mit einem M beginnen und mit us enden, also beispielsweise Mus, Maus oder auch Markenstatus. Ein Fragezeichen steht für genau einen Buchstaben. `Ma?s` passt also auf Mais oder Maus, aber nicht auf Markenstatus. Diese Platzhalter werden als *Wildcards* bezeichnet.

Die Eigenschaften der Dateien anzeigen

Der Befehl `ls` gibt nur die nackten Dateinamen aus. Wird der Befehl `ls -l` abgesetzt, erscheint eine Liste von Dateien, die weitere Informationen enthält. Dazu gehören Eigenschaften, Rechte, Besitzer, Größe, letzter Änderungszeitpunkt und eben der Name der Datei.

```
-rw-rw-r--  1 arnold arnold    6813 Mär  6 20:39 out.txt
-rwxr-xr-x  1 arnold arnold  127014 Apr 12 02:06 control
-rw-rw-r--  1 arnold arnold   21997 Apr  2 23:49 datasync.tex
drwxr-xr-x 11 arnold arnold   20480 Aug 17 06:41 Videos
-rw-------  1 arnold arnold 6922652 Aug 11  2010 org.img.gz
lrwxrwxrwx  1 arnold arnold      18 Aug 17 07:05 img.gz -> org.img.gz
```

1. Das erste Zeichen der Ausgabezeile von `ls -l` verrät die Art der Datei. Eine normale Datei hat ein Minuszeichen, ein `d` steht für ein Verzeichnis und ein `l` für einen symbolischen Link.

2. Die neun Zeichen dahinter beschreiben die Zugriffsrechte. `r` steht für Lesen, `w` für Schreiben und `x` für Ausführen. Die Rechte werden dreimal aufgeführt. Die erste Rechtekombination gilt für den Besitzer, die zweite für die Gruppe und die letzte für den Rest der Anwender. Das Thema wird im Zusammenhang mit den Dateirechten und dem Befehl `chmod` weiter ausgeführt.

3. Die Spalte danach zeigt die Verweise an, die es für diese Datei gibt. Bei Verzeichnissen sind das die Dateien und Verzeichnisse, die sich im Verzeichnis befinden. Bei Dateien ist es die Anzahl der harten Links auf die Datei.

4. Die nächste Spalte nennt den Besitzer der Datei.

5. Es folgt die Gruppe, der die Datei gehört.
6. Es folgt die Größe der Datei in Byte.
7. Datum und Uhrzeit der letzten Änderung. Die erste Spalte gibt den Monatsnamen und die zweite den Tag an. Die dritte Spalte enthält das Jahr oder, wenn die Datei noch nicht älter als ein Jahr ist, die Uhrzeit.
8. Zuletzt wird der Dateiname angezeigt. Bei symbolischen Links sehen Sie auch nach dem angedeuteten Pfeil, wohin der Link zeigt.

Der Befehl `ls` hat eine ganze Stange von Optionen. Mit dem Befehl `man ls` erhalten Sie den passenden Handbucheintrag. Oft ist eine Kombination die optimale Lösung. Meine Lieblingsoption ist übrigens `ls –lrt`. Bei einbuchstabigen Optionen können diese zusammengestellt werden und sind so leichter zu tippen als `ls –l –r –t`. Der Befehl zeigt die Langdarstellung (l). Alle Dateien werden zeitlich sortiert (t), aber in umgekehrter Reihenfolge (r). Damit sind die zuletzt geänderten Dateien ganz unten, also direkt oberhalb der neuen Eingabe. Denn meistens sind es genau diese, die wirklich interessant sind.

Mit Verzeichnissen arbeiten

Verzeichnisse sind Sammelbehältnisse für Dateien. Sie werden durch das d am Anfang der Zeile bei einem Aufruf von `ls –l` gekennzeichnet. Gibt man den Verzeichnisnamen als Argument an, dann schaut der `ls` allerdings in das Verzeichnis hinein. Will man das verhindern, um beispielsweise die Rechte des Verzeichnisses zu sehen, muss man die Option `–d` angeben.

Der Verzeichnistrenner ist der einfache Schrägstrich, nicht der Backslash \, wie er bei MS-DOS eingeführt wurde, weil der Schrägstrich dort schon als Optionszeichen verwendet wurde, als MS-DOS noch keine Verzeichnisse kannte.

Erzeugen

Der Befehl `mkdir` erzeugt die Verzeichnisse, die ihm als Argument übergeben werden. Mit der Option `–p` kann eine komplette Hierarchie angelegt werden. Beispiel:

```
$ mkdir –p eins/zwei/drei/vier
```

Gab es noch kein Verzeichnis namens *eins*, wird die komplette Verzeichnislinie angelegt. Falls es schon das Verzeichnis *eins/zwei* gab, werden nur *drei* und *vier* erzeugt. Ohne `–p` müsste der Pfad *eins/zwei/drei* existieren, damit der Befehl fehlerfrei arbeitet.

Löschen

Der Befehl `rmdir` löscht leere Verzeichnisse. Befindet sich eine Datei im Verzeichnis, weigert sich `rmdir`, diese und auch das Verzeichnis zu löschen. Sie müssten in diesem Fall den Befehl `rm –r` verwenden. Dieser löscht alle angegebenen Verzeichnisse und auch die darin enthaltenen Verzeichnisse und Dateien, also rekursiv.

Gerade wenn Sie rekursiv löschen, kann es Ihnen passieren, dass sich ein Dateirecht querstellt. Mit der Option `-f` oder `--force` zwingen Sie `rm` dazu, keine Rückfragen zu stellen.

Wechseln

Mit dem Befehl `cd` wechseln Sie in ein Verzeichnis, das damit zum Arbeitsverzeichnis wird. Die Eingabe von `cd` ohne Argument führt zur Wurzel des Heimatverzeichnisses zurück. Auch die Tilde (~) bezeichnet das Heimatverzeichnis.

Mit dem Befehl `pwd` erhalten Sie das Verzeichnis, in dem Sie sich gerade befinden. Allerdings erscheint das Verzeichnis immer im sogenannten Prompt links neben dem Cursor.

Mit Dateien arbeiten

Sie können Dateien auch auf dem Terminal kopieren, umbenennen, verschieben und löschen.

Dateien kopieren

Der Befehl `cp` kopiert Dateien. Als Argumente erwartet der Befehl zunächst die zu kopierenden Dateien und als Letztes den Namen der Zieldatei oder das Zielverzeichnis. Sie können also durch Aufzählung oder durch Verwendung von Wildcards mehrere Dateien auf einen Schlag kopieren, müssen dann aber ein Verzeichnis als Ziel angeben.

- ✔ Dateirechte übernehmen: `-p`

  Der Befehl `cp` erzeugt immer eine neue Datei mit dem aktuellen Datum. Der Aufrufer des Befehls wird automatisch zum Besitzer der Datei. Sollen allerdings die Eigenschaften der Originaldatei übernommen werden, geben Sie die Option `-p` an.

- ✔ Rekursive Kopie: `-r`

  Mit der Option `-r` können komplette Verzeichnisbäume kopiert werden. So wird der folgende Befehl das Verzeichnis *verzeichnis* und alle darunterliegenden Dateien und Verzeichnisse in das Verzeichnis */tmp* kopieren:

  ```
  $ cp -r verzeichnis /tmp
  ```

- ✔ Update: `-u`

  Mit der Option `-u` können Sie erreichen, dass nur ältere Dateien durch neuere ersetzt werden.

Dateien verschieben oder umbenennen: mv

Mit dem Befehl `mv` können Sie Dateien verschieben. Als Argument erwartet er beliebig viele Dateien. Das letzte Argument muss ein Verzeichnis sein, in das die Dateien verschoben werden. Hier werden alle Dateien, die auf **.txt* enden, in das Verzeichnis *eins/zwei* verschoben.

```
$ mv *.txt eins/zwei
```

Wird als Argument nur eine Datei oder ein Verzeichnis angegeben und das Ziel ist kein existierendes Verzeichnis, wird der Befehl den Namen ändern.

```
$ mv saulus paulus
```

Das Verschieben von Dateien geschieht oft erstaunlich schnell. Linux kann erkennen, wenn Quelle und Ziel auf dem gleichen Dateisystem sind, und wird dann die Einträge in den Verzeichnissen umschieben und die dahinterstehenden Daten dort lassen, wo sie sind.

Da verschobene Dateien nicht neu angelegt werden, behalten sie auch ihre ursprünglichen Eigenschaften bezüglich Eigentümern, Rechten und dem letzten Änderungsdatum.

Dateien löschen

Mit dem Befehl `rm` löschen Sie alle Dateien, die Sie als Argumente angeben. Da das Löschen von Dateien so endgültig ist, gibt es die Möglichkeit, durch die Option `–i` den Anwender bei jeder einzelnen Datei noch einmal zu fragen, ob es wirklich sein Ernst ist.

Datei- und Verzeichnisnamen

Unter Linux kann ein Dateiname fast jedes beliebige Zeichen enthalten. Die folgenden Zeichen sollten Sie allerdings in Dateinamen nicht verwenden:

- ✔ **Schrägstrich** Der Schrägstrich dient unter Linux als Verzeichnistrenner.
- ✔ **Doppelpunkt** Der Doppelpunkt wird im Netzwerk dazu verwendet, einen fremden Rechner anzusprechen.
- ✔ **Minuszeichen am Namensanfang** Das Minuszeichen wird von den Kommandozeilenprogrammen als Zeichen für eine Option verwendet. Diesen Programmen wird es also schwerfallen, eine solche Datei zu bearbeiten, weil sie den Dateinamen nicht erkennen und ihn für eine Option halten.
- ✔ **Stern und Fragezeichen** Beide Zeichen werden von der Kommandozeile aus als Wildcard verwendet. Es ist zwar möglich, diese Zeichen durch Anführungszeichen in ihrer Sonderfunktion »auszublenden«, es wird aber in jedem Fall eine gewisse Verwirrung auftreten.
- ✔ **Internationale Sonderzeichen** Im deutschen Sprachraum ist die Versuchung des Anwenders groß, Dateinamen auch aus Umlauten zu bilden. Andererseits möchten Sie vermutlich auch nicht nach den skandinavischen Sonderzeichen auf Ihrer Tastatur suchen, wenn ein Däne auf die gleiche Weise vorgeht.

Groß- und Kleinschreibung

Linux unterscheidet bei den Dateinamen zwischen großen und kleinen Buchstaben. Im selben Verzeichnis kann die Datei *Makefile* neben der Datei *makefile* stehen. Windows sieht das leider vollkommen anders.

Verborgene Dateien mit Punkt

Dateien und Verzeichnisse, die mit einem Punkt beginnen, werden von den Standardbefehlen nur dann zur Kenntnis genommen, wenn die Option `-a` eingesetzt wird oder der Name direkt angesprochen wird. Auf diese Weise sind Dateien und Verzeichnisse, die mit einem Punkt beginnen, quasi unsichtbar. Dieses Verhalten wird teilweise auch von grafischen Dateimanagern übernommen. Beim Dateimanager Caja beispielsweise können Sie mit der Tastenkombination Strg+H hin- und herschalten.

Dieser Effekt wird gern in den Benutzerverzeichnissen verwendet, um Konfigurationsdateien etwas aus dem Sichtfeld zu schaffen.

Fortgeschrittene Dateibefehle

Unter Linux werden Konfigurationen gern in Textdateien gehalten. Auch Fehlermeldung werden sorgfältig in Textdateien abgelegt, Programmierer verwenden solche Dateien, weil Compiler an hübscher Gestaltung der Texte kein Interesse haben. Passend dazu liefert Linux ein Arsenal von Befehlen, mit denen man Textdateien durchsuchen, analysieren und verändern kann.

Dateiinhalte anzeigen

Der Befehl `cat` liest eine oder mehrere Dateien aus und befördert die Inhalte in die Standardausgabe. Das ist in der Regel der Bildschirm. Der folgende Befehl stellt den Inhalt der Benutzerdatei */etc/passwd* auf dem Bildschirm dar:

```
$ cat /etc/passwd
```

Seitenweise blättern: more und less

Um Textinhalte anzusehen, die größer sind, macht `cat` keinen Spaß. Da ist das Programm `more` deutlich nützlicher, denn es zeigt immer eine Bildschirmseite an und wartet, bis Sie mit der Leertaste weiterblättern.

```
$ more /etc/passwd
```

Neben der Leertaste beherrscht `more` noch ein paar Tasten mehr:

- ✔ Die Leertaste blättert seitenweise.
- ✔ Die Taste ← blättert zeilenweise.
- ✔ Mit der Taste Q können Sie die Ausgabe abbrechen.

Die GNU-Version von `more` heißt `less` und getreu dem Motto »less is more than more« kennt `less` noch ein paar Tricks mehr. So kann `less` die normalen Cursortasten einsetzen, aber vor allem kann `less` auch rückwärts blättern.

Der Befehl `less` erlaubt sogar das Suchen im Datenstrom. Dazu verwenden Sie wie im `vi` den Schrägstrich, und geben den Suchbegriff ein und schließen ihn mit ⏎ ab.

Wozu benötigen Sie `more`, wenn `less` doch alles besser kann? Um vor- und zurückgehen zu können, muss `less` die Datei in den Hauptspeicher holen. Das kann bei extrem großen Dateien schwierig werden. Hier ist `more` also im Vorteil.

Wollen Sie nur die ersten zehn Zeilen einer Datei betrachten, hilft der Befehl `head`. Nach der Option `–n` können Sie auch eine andere Anzahl von Zeilen sehen.

```
$ head /etc/passwd
```

Was vorn geht, geht auch hinten. Mit dem Befehl `tail` können Sie sich die letzten zehn Zeilen einer Datei ansehen. Gerade bei der Fehlerdatei in */var/log/syslog* sind die neuesten Ereignisse logischerweise am Schluss.

```
$ tail /var/log/syslog
```

Richtig spannend aber wird es, wenn man eine Live-Übertragung der Fehlerentstehung mitverfolgen kann. Wenn Sie dem Befehl `tail` die Option `–f` spendieren, führt es dazu, dass der Befehl mit der Datei verbunden bleibt. Werden neue Zeilen in die Datei angehängt, werden diese zeitgleich ausgegeben.

```
# tail –f /var/log/syslog
```

Wenn Sie nun einen USB-Stick einstecken, werden Sie sehen, wie die neue Hardware erkannt und behandelt wird. Beim Start eines Servers können Sie mitverfolgen, wenn dieser scheitert, und bekommen darüber hinaus eine Ahnung, warum das passiert. Die Übertragung mit `tail –f` wird durch die Tastenkombination Strg+C beendet.

Durchsuchungsbefehl: grep

Der Befehl `grep` durchsucht Textdateien nach Textmustern. Findet er eine Zeile, in der das Muster auftaucht, zeigt er die Zeile an. Das folgende Beispiel zeigt, wie `grep` die Zeichenkette »href« in allen Dateien des Verzeichnisses sucht, die auf *.htm* enden.

```
$ grep href *.htm
```

Da in HTML der `href` sowohl klein- als auch großgeschrieben werden kann, können Sie mit der Option `–i` nach beiden Fällen suchen.

```
$ grep href –i *.htm
```

Der Suchbegriff sollte in Anführungszeichen gesetzt werden, wenn er Leerzeichen oder auch Sonderzeichen enthält, die von der Shell (siehe Kapitel 2 interpretiert werden könnten.

Der Befehl `grep` kennt einige Optionen, von denen folgende häufiger eingesetzt werden:

-i ignoriert Groß- und Kleinschreibung.

-l erstellt eine Liste der Dateien mit einem Treffer.

-v zeigt die Zeilen, in denen der Begriff nicht vorkommt. In Kombination mit `-l` werden nur die Namen der Dateien angezeigt, die den Suchbegriff nicht enthalten.

Sie können nach zwei Begriffen in einer Zeile suchen, indem Sie das Ergebnis eines `grep`-Befehls mit einem senkrechten Strich in einen zweiten `grep`-Befehl weiterleiten. Wenn Sie in Ihren Dateien alle Links auf Wikipedia suchen, könnte der Befehl so ergänzt werden:

```
$ grep href *.htm | grep wikipedia
```

Suchen und Agieren: find

Der Befehl `find` sucht nach Dateien in einem Verzeichnis und dessen Unterverzeichnissen. Er spürt die Dateien nicht nur anhand des Namens auf, sondern auch aufgrund des Alters, der Größe oder anderer Eigenschaften. Sie können `find` dazu bewegen, auf die gefunden Dateien Befehle einwirken zu lassen.

Durch geschickte Kombinationen und Aktionen können Sie veraltete Datensicherungen entsorgen, prüfen, ob irgendwo übermäßig große Dateien auftreten, oder beispielsweise Dateien eines bestimmten Besitzers archivieren, die ein gewisses Alter überschritten haben.

Der Aufbau eines `find`-Befehls:

- ✔ Direkt auf den Befehl `find` folgt der Pfadname des Verzeichnisses, das das Kommando durchsuchen soll. Oft ist es genau das Verzeichnis, in dem Sie sich gerade befinden, sodass hier ein Punkt angegeben wird.
- ✔ Sie können mit Optionen die gewünschten Dateien ausfiltern, beispielsweise nach deren Dateiname oder Alter.
- ✔ Mit der Option `-exec` können Sie Befehle auf die gefundenen Dateien loslassen.

Filter

Mit verschiedenen Optionen können Sie die Ergebnisse von `find` filtern. Durch die Kombination mehrerer Optionen schränken Sie die Ergebnismenge von `find` immer weiter ein.

Filtern nach Dateinamen: -name und -iname

Nach Eingabe des Befehls `find` und des Ausgangsverzeichnisses wird für die Suche nach Dateinamen die Option `-name` gefolgt von dem gesuchten Namen angegeben.

```
$ find . -name 11exceptions.tex
./tex/java/11exceptions.tex
```

Falls Sie sich nicht mehr genau an den Namen der Datei erinnern, aber noch wissen, dass der Dateiname das Wort *exception* enthält, können Sie einen Stern als Dateimaske einsetzen. Allerdings sollten Sie in dem Fall den Dateinamen mit Anführungszeichen umgeben:

```
$ find . -name "*exception*"
./tex/cpp/listings/exception.cpp
./tex/java/11exceptions.tex
```

Nun hat `find` schon zwei Dateien gefunden. Wenn Sie die Option `–iname` verwenden, wird auch noch Groß- und Kleinschreibung ignoriert.

Dateityp: -type

Die Option `–type` sucht nach Dateien eines bestimmten Typs. Welchen Typ Sie suchen, geben Sie als weiteren Parameter an.

- **-type d** Verzeichnisse
- **-type f** Dateien
- **-type l** symbolische Links
- **-type b** Blockgerätedateien
- **-type c** Zeichengerätedateien
- **-type s** Socket-Kommunikationspunkte
- **-type p** Pipe-Kommunikationspunkte

Das folgende Beispiel zeigt alle Verzeichnisse unterhalb des aktuellen Verzeichnisses an.

```
$ find . –type d
.
./pic
./example
```

Wenn Sie die Suche darüber hinaus auf spezielle Dateisysteme (siehe Abschnitt 7) einschränken wollen, hilft Ihnen die Option `–fstype`. Als Parameter benötigt sie einen Dateisystemtyp, wie er auch bei `mount` (siehe Abschnitt 7) verwendet wird.

Berechtigungen: -perm

Mit der Option `–perm` (für »permission«) können Sie nach Berechtigungen filtern. Die Berechtigung geben Sie in der Zahlendarstellung an, die auch bei dem Befehl `chmod` üblich ist.

```
$ find . –perm 644
```

Dieser Befehl zeigt alle Dateien und Verzeichnisse an, die vom Besitzer schreib- und lesbar sind und von der Gruppe und der Welt nur lesbar sind.

Wollen Sie alle Dateien haben, die mindestens die angegebenen Rechte besitzen, müssen Sie der Zahl ein Minuszeichen voranstellen.

```
$ find . –perm –644
```

Besitzfragen: -user, -uid, -group und -gid

Sie können die Ergebnisse auch nach dem Besitzer oder der Gruppe filtern. Je nach Art der Frage ist die Kennung oder die ID als Parameter zu übergeben.

-user sucht nach dem Benutzernamen

-uid sucht nach der Benutzernummer

-group sucht nach dem Gruppennamen

-gid sucht nach der Gruppennummer

Filtern nach Zeitstempel

Sie können auch über das Alter der Dateien filtern. Damit können Dateien selektiert werden, die seit einem bestimmten Zeitpunkt verändert wurden oder schon älter und vielleicht nicht mehr wichtig sind.

Bei der Option `–mmin` geben Sie an, wie viele Minuten höchstens seit der letzten Änderung vergangen sein dürfen. Die Option `mtime` verwendet ebenfalls eine Zahl. Diese ist dann allerdings die Anzahl der Tage.

Hat die Zahl, die als Parameter für `–mmin` oder `mtime` angegeben wird, kein Vorzeichen, muss diese Zeit exakt auf die Dateizeit passen. Bei `–mmin +10` werden die Dateien angezeigt, die mindestens vor 10 Minuten geändert wurden. `–mmin –10` gilt für diejenigen Dateien, deren Änderung höchstens zehn Minuten her ist.

Sollen beispielsweise alle Dateien gelöscht werden, die auf *backup.tar* enden und älter als eine Woche sind, würde der Befehl lauten:

```
find . –mtime +7 –name "*backup.tar" –exec rm {} \;
```

Lassen Sie sich nicht von der Option `–exec` erschrecken. Diese wird weiter unten beschrieben. Im Augenblick müssen Sie mir einfach glauben, dass die Dateien gelöscht werden.

Die Option `–newer` vergleicht nicht mit einem Datum, sondern erwartet als weiteren Parameter eine Datei und vergleicht, ob die gefundene Datei neuer ist als die angegebene.

Die Größe der Datei

Sie können mit der Option `–empty` nach leeren Dateien suchen. Über die Option `–size` können Sie die Größe der gesuchten Datei angeben. Dabei wird an die Zahl ein Buchstabe für die Größeneinheit angehängt:

Zeichen	Einheit
`b`	Anzahl der Blöcke
`c`	Byte
`k`	Kilobyte
`M`	Megabyte
`G`	Gigabyte

Tabelle 1.1: Einheiten der Option `–size`

Ein + als Vorzeichen der Parameterzahl bewirkt, dass die Datei mindestens, und ein –, dass sie höchstens so groß sein darf. Der folgende Befehl sucht alle Dateien unterhalb des aktuellen Verzeichnisses, die kleiner als 700 Byte sind.

```
$ find . –size –700c
```

Ausführung von Befehlen

Die Option –exec ermöglicht es, Befehle auf die gefundenen Dateien anzuwenden. Den Befehl, den Sie anwenden wollen, schreiben Sie direkt hinter die Option –exec.

Der Befehl erwartet als Argument vermutlich eine Datei. Die Idee ist natürlich, dass der Befehl jeweils auf die gefundene Datei angewendet wird. Als Stellvertreter für die jeweils gefundene Datei verwendet find ein Paar geschweifter Klammern.

Der Befehl der Option –exec muss mit einem Semikolon abgeschlossen werden. Damit dieses Semikolon nicht von der Shell ausgewertet wird und so verschwindet, müssen Sie einen Backslash davorstellen. So würde der Befehl, der alle Dateien löscht, deren Name *exception* enthält, folgendermaßen lauten:

```
$ find . –name "*exception*" –exec rm {} \;
```

Verknüpfungen von Optionen

Sie müssen sich den Ablauf eines find-Befehls so vorstellen, dass alle Dateien des angegebenen Pfades nacheinander betrachtet werden. Dabei werden alle Filteroptionen geprüft, ob sie auf die gefundene Datei jeweils zutreffen. Sind mehrere Optionen angegeben, müssen alle erfüllt sein. Es handelt sich nach der Aussagenlogik also um eine UND-Verknüpfung.

Wenn Sie beispielsweise feststellen, dass in den letzten drei Tagen die bis vor Kurzem noch fast leere Festplatte voll ist, werden Sie wissen wollen, wo sich große Dateien befinden, die in den letzten drei Tagen entstanden sind. Dazu gehen Sie an das Wurzelverzeichnis der Festplatte und geben den folgenden Befehl ein:

```
$ find . –size +1G –mtime –3
```

Die angezeigten Dateien sind über ein Gigabyte groß und wurden in den letzten drei Tagen verändert.

Neben der UND-Verknüpfung, die nur wahr ist, wenn alle Teilbedingungen erfüllt sind, können Sie auch eine ODER-Verknüpfung anwenden, die wahr ist, wenn wenigstens eine der Bedingungen erfüllt ist. Um zwei Optionen per ODER zu verknüpfen, müssen Sie zwischen die Optionen die Option –o stellen.

Sie können sehr komplexe boolesche Ausdrücke basteln. Dazu stehen Ihnen die folgenden Möglichkeiten zur Verfügung:

- ✔ ODER können Sie mit –o definieren.
- ✔ AND ist die Vorgabe, kann aber durch –a explizit genannt werden.

- ✔ Eine Negierung ist über ein Ausrufezeichen zu realisieren.
- ✔ Um die Ausdrücke zusammenzufassen, können Sie Klammern verwenden, müssen diese aber jeweils mit einem Backslash versehen, um Missverständnisse mit der Shell zu vermeiden.

Das folgende Beispiel löscht alle Dateien namens *.rhosts*, die unterhalb des Verzeichnisses */home* liegen und nicht die Rechtekombination 0600 haben:

```
$ find /home -name .rhosts -a ! -perm 0600 -exec rm {} \;
```

Verweis auf andere Dateien: Links

Mit Links ist es nicht nur möglich, dass mehrere Dateinamen auf dieselbe Datei verweisen. Es gibt sogar zwei Arten solcher Verweise, die gern als Link bezeichnet werden.

- ✔ Der sogenannte »harte« Link entsteht, wenn zwei oder mehr Einträge in Verzeichnissen auf ein und denselben Datenblock verweisen. Dazu muss der Verweis auf dem gleichen Dateisystem stehen. Das Dateisystem zählt die Anzahl der Verweise mit und löscht den Datenblock erst, wenn der letzte Verweis gelöscht wird.
- ✔ Der symbolische Link enthält dagegen den kompletten Dateinamen inklusive der Verzeichnisse und kann auf beliebige Dateien oder Verzeichnisse im Dateisystem zeigen, allerdings auch auf solche, die inzwischen vielleicht entfernt worden sind.

Der Befehl für das Erzeugen von Verweisen lautet `ln`.

Anlegen eines harten Links

Der Befehl `ln` ähnelt syntaktisch dem Kopierbefehl `cp` und erwartet als erstes Argument eine existierende Datei und als zweites Argument einen neuen Dateinamen für diese Datei.

Das spielen wir gleich mal durch: Mit dem Befehl `ls -l` sehen wir uns das aktuelle Verzeichniss genau an. Die Ausgangsdatei heißt hier *original*. Wir erzeugen dann mit dem Befehl `ln` einen Link mit dem Namen *dublette* und schauen dann noch einmal hin.

```
$ ls -l original
-rw-r--r-- 1 arnold arnold 5267 Okt 18 17:37 original
$ ln original dublette
$ ls -l original dublette
-rw-r--r-- 2 arnold arnold 5267 Okt 18 17:37 dublette
-rw-r--r-- 2 arnold arnold 5267 Okt 18 17:37 original
$ diff original dublette
$
```

Nach der Ausführung des Befehls sehen Sie, dass die Zahl zwischen den Dateirechten und dem Besitzernamen erhöht wurde. Dies ist der Linkzähler, der normalerweise immer 1 ist. Weiter sehen Sie, dass die Dateien bezüglich Datum, Uhrzeit, Rechten und Größe gleich aussehen.

Auch der Befehl `diff`, der jegliche Unterschiede zwischen Dateien aufzeigt, bestätigt, dass die Dateien gleich sind. Was Sie nicht auf Anhieb sehen können, ist, dass der zusätzliche Link keinen zusätzlichen Festplattenplatz für die Daten benötigt.

Die Verweise müssen nicht im gleichen Verzeichnis liegen, aber aus technischen Gründen auf dem gleichen Dateisystem.

Der symbolische Link

Während nach dem Setzen eines harten Links das Original vom Duplikat nicht mehr zu unterscheiden ist, ist ein symbolischer Link lediglich ein textueller Verweis auf eine Datei. Ein symbolischer Link wird ebenfalls mit dem Befehl `ln` angelegt, allerdings wird die Option `–s` hinzugefügt.

```
$ ln –s original link
$ ls –l
–rw–r––r–– 2 arnold arnold 5267 Okt 18 17:37 dublette
lrwxrwxrwx 1 arnold arnold    8 Okt 18 17:47 link –> original
–rw–r––r–– 2 arnold arnold 5267 Okt 18 17:37 original
$ diff original link
$
```

Der Linkzähler verändert sich beim Setzen eines symbolischen Links nicht. Der Befehl `ls –l` zeigt einerseits ein kleines »L« ganz links beim symbolischen Link. Darüber hinaus wird auch der Zielort des symbolischen Links rechts hinter einem stilisierten Pfeil angezeigt. Tatsächlich enthält ein symbolischer Link auch nur den Dateinamen, auf den er verweist. Darum ist die Datei *link* auch nur acht Byte groß. Der Vergleich über den Befehl `diff` zeigt, dass die Dateien *link* und *original* identisch sind. Tatsächlich betrachten also die Linux-Befehle symbolische Links fast immer genauso wie die Datei, auf die sie verweisen.

Der symbolische Link ist sehr flexibel.

- ✔ Er kann quer über alle Dateisysteme und Festplatten hinweg verweisen, weil er ja nur den Ort der Datei bezeichnet.
- ✔ Der symbolische Link ist nicht auf Dateien beschränkt. Sie können auch einen symbolischen Link auf ein Verzeichnis setzen und er wird als Verzeichnis behandelt.

Die Einschränkung ist, dass das Dateisystem den symbolischen Link nicht kontrolliert. Wird das Original gelöscht, verschwindet der symbolische Link nicht automatisch, sondern verweist ins Leere. Sie müssen wohl selbst für Ordnung sorgen.

Spezielle Dateien: Sockets, Pipes und Gerätedateien

Für die Kommunikation verwendet Linux sowohl Socket- als auch Pipe-Dateien. In den meisten Fällen haben Sie damit wenig zu tun, außer dass Sie Dateien finden, die ein kleines `s` oder ein `p` in der ersten Spalte einer Ausgabe von `ls –l` haben.

Diese Dateien dienen als Kommunikationsendpunkte und manche Programme verwenden sie für interne Zwecke. Das Hauptproblem für den Administrator besteht darin, dass sie nicht einfach durch Kopieren zu sichern sind.

Gerätedateien

Ganz besondere Dateien finden Sie im Verzeichnis */dev*. Diese Dateien sind keine echten Dateien, sondern dienen als Zugriffspunkte für Hardware. Beispielsweise werden wir damit Festplatten und Partitionen bearbeiten. Sie können diese Dateien in vieler Hinsicht mit üblichen Dateibefehlen behandeln.

Sockets und Pipes

Sockets sind vor allem als Netzwerkverbindungen bekannt. Es ist aber auch möglich, Sockets im Dateisystem anzulegen. Dabei wird eine Datei angelegt, die als Kommunikationsknotenpunkt zwischen zwei Prozessen angelegt wird.

Ganz ähnlich verhält es sich mit Pipes. Dies sind Daten, die von einem Prozess an einen anderen weitergereicht werden und dazu das Dateisystem verwenden.

In beiden Fällen entstehen diese Dateien durch eine Kommunikation zwischen Prozessen, verschwinden aber auch wieder, wenn sie nicht mehr benötigt werden. Aus Sicht des Administrators sind diese nicht so interessant. Sie müssen und können auch gar nicht gesichert werden.

Der Linux-Verzeichnisbaum

Ein vollständig qualifizierter Dateiname besteht aus dem Namen der Datei und dem vollständigen Verzeichnispfad. Dieser Pfad muss dann mit einem Schrägstrich beginnen. Damit ist der Pfad positionsunabhängig und eindeutig. Einige Programme fordern den vollständig qualifizierten Dateinamen.

- ✔ Ein Verzeichnispfad, der mit einem Schrägstrich beginnt, wird *absolut* genannt, weil er positionsunabhängig ist.
- ✔ Fehlt am Anfang des Pfades der Schrägstrich, wird der Pfad *relativ* zum aktuellen Verzeichnis gesehen.
- ✔ Das aktuelle Verzeichnis nennt man *Arbeitsverzeichnis*

Der Standard-Verzeichnisbaum

Es gibt ein paar Verzeichnisse, die bestimmte Aufgaben haben. Diese Verzeichnisstrukturen haben sich bereits unter UNIX über viele Jahre entwickelt und sind inzwischen durch den FHS (File Hierarchy Standard) vorgegeben.

/etc Das Verzeichnis */etc* enthält die Konfigurationsdateien der Programme und des Systems. Insbesondere bei Server-Programmen ist dies oft ein Verzeichnis, in dem dann mehrere Konfigurationsdateien abgelegt sind.

/tmp Das Verzeichnis */tmp* ist für temporäre Dateien gedacht und kann von jedem Benutzer des Systems gelesen und geschrieben werden. Nach einem Neustart ist dieses Verzeichnis häufig leer, da es oft als RAM-Disk realisiert wird. Darum ist es kein Ort, um dort wichtige Daten zu parken.

/usr Das Verzeichnis */usr* enthält die Dateien, die für alle Benutzer des Systems zur Verfügung gestellt werden. Sie können gelesen oder ausgeführt, aber nicht geschrieben werden.

/usr/bin Hier liegen fast alle ausführbaren Programme. Da nur der Administrator in diesem Bereich Schreibrecht hat, ist gesichert, dass keine unerwünschte Software installiert werden kann.

/usr/lib Hier lagern die dynamischen Bibliotheken und andere wichtige Dateien wie Plugins für die Programme.

/usr/sbin Für die Tools des Administrators gibt es das Verzeichnis */usr/sbin*.

/var Wenn Systemprogramme oder Serverprogramme Daten ablegen wollen, verwenden sie dazu Unterverzeichnisse des Verzeichnisses */var*. Als Administrator interessiert Sie dort besonders das Verzeichnis */var/log*, in dem die meisten Programme ihre Schmerzen dokumentieren. Der Webserver legt seine Daten beispielsweise standardmäßig unter */var/www* ab.

/home Für jeden Benutzer wird im Verzeichnis */home* ein Unterverzeichnis angelegt. Hier und nur hier können Anwender Daten hinterlassen. Das ist für die Datensicherung interessant, weil Sie durch die Sicherung dieses Verzeichnis alle Benutzerdaten erwischt haben dürften.

/opt Das Verzeichnis *opt* wird teilweise von größeren Programmpaketen verwendet, damit deren Dateien an einem Ort versammelt sind. Dies wird insbesondere oft von Herstellern verwendet, die sich nicht gern mit den Besonderheiten von Linux beschäftigen mögen.

/mnt Das Verzeichnis */mnt* steht für das kurzfristige Einhängen (mount) von Dateisystemen zur Verfügung. Darum sollte es auch nicht dauerhaft belegt sein. Für längerfristiges Einhängen sollten Sie */media* verwenden.

/media Wenn Sie beispielsweise einen USB-Stick in Ihr Gerät einstecken, wird dieser unterhalb des Pfades */media* eingehängt. Um genau zu sein, wird unter */media* ein Verzeichnis mit der Benutzerkennung und dann ein Verzeichnis mit dem Namen des Sticks angelegt, also beispielsweise */media/arnold/MEINUSB*.

Das Verzeichnis ist auch ein guter Ort, um darin Verzeichnisse für das längerfristige Einhängen externer Festplatten oder Netzwerkverzeichnisse anzulegen.

/proc Das Pseudoverzeichnis */proc* enthält keine echten Dateien, sondern dient als Schnittstelle zum Betriebssystem und den Prozessen. Hier können verschiedene Informationen über das System gewonnen werden. So liefert der Befehl `cat /proc/meminfo` beispielsweise eine Übersicht über die aktuelle Belegung des Hauptspeichers.

/dev Das Verzeichnis */dev* enthält Pseudodateien, die den Zugriff auf die Hardware ermöglichen, beispielsweise auf Festplatten und ihre Partitionen.

Besitzer, Gruppen und Rechte

Dateien haben Besitzer. Üblicherweise hat der Besitzer das Recht, die Datei zu lesen und zu schreiben. Eine Besonderheit sind Programmdateien, die auch ausgeführt werden können. Diese drei Rechte werden mit den Buchstaben r, w und x gekennzeichnet. Intern werden sie mit 4, 2 und 1 gekennzeichnet. Durch diese Wahl ist es möglich, eine beliebige Kombination zu erstellen, indem man die Zahlen addiert. Heraus kommt ein Wert zwischen 0 und 7. Das Recht 5 besteht aus 4 und 1, also r und x und bedeutet, dass die Datei les- und ausführbar, aber nicht schreibbar ist. Tabelle 1.2 stellt dies übersichtlich zusammen.

Recht	Zahl	Bedeutung für Dateien
`r`	4	Die Datei darf gelesen werden.
`w`	2	Die Datei darf geändert, also geschrieben werden.
`x`	1	Die Datei darf ausgeführt werden.

Tabelle 1.2: Die Rechte für Dateien

Anders als unter Windows wird ein Programm also nicht durch seine Endung wie etwa *.exe* ausführbar, sondern durch die Dateieigenschaft.

Wenn Sie in einem Verzeichnis den Befehl `ls -l` aufrufen, sehen Sie die dort liegenden Dateien mit ihren Rechten. Diese bestehen aus drei Gruppen, die jeweils aus drei Buchstaben bestehen. Die ersten drei Zeichen stehen für die Rechte des Besitzers, die zweite für die der Gruppe und die dritte Kombination für den Rest der Welt.

```
$ ls -l moin
-rwxr-x--x    1 arnold   users   13489 2010-07-06 21:34 moin
```

Die Datei gehört dem Benutzer arnold und der Gruppe users.

- ✔ Die Datei *moin* darf vom Besitzer arnold gelesen, geschrieben und ausgeführt werden.
- ✔ Die Gruppe users darf sie lesen und ausführen, aber nicht verändern.
- ✔ Der Rest der Welt darf die Datei ausführen, aber weder hineinschauen noch sie verändern.

Dateiberechtigungen ändern

Diese Rechtecodierung durch 4, 2 und 1 verwendet der Befehl `chmod` als ersten Parameter, um die Berechtigungen einer Datei zu ändern. Im folgenden Beispiel erhält die Datei *moin* zweimal neue Rechte.

```
$ ls -l moin
-rwxr-xr-x    1 arnold   users   13489 2010-07-06 21:34 moin
$ chmod 641 moin
$ ls -l moin
-rw-r----x    1 arnold   users   13489 2010-07-06 21:34 moin
$ chmod 755 moin
$ ls -l moin
-rwxr-xr-x    1 arnold   users   13489 2010-07-06 21:34 moin
```

Die Rechtslage bei Verzeichnissen

Ein Verzeichnis ist eine besondere Art einer Datei, in der die darin liegenden Dateien aufgelistet sind.

- ✔ Nur mit dem Schreibrecht (w) dürfen Sie darin befindlichen Dateien löschen oder umbenennen, und Sie können natürlich neue Dateien anlegen.
- ✔ Das Leserecht (r) erlaubt Ihnen, in das Verzeichnis hineinzuschauen.
- ✔ Das Ausführungsrecht (x) ist nicht so anschaulich. Dies berechtigt Sie zum Wechseln in das Verzeichnis.

Rechte ändern über Buchstaben statt Zahlen

Neben der Zahlenkombination erlaubt der Befehl `chmod` auch Buchstabenordnungen. Diese ersetzen nicht die Zahlen, sondern ermöglichen vor allem das flexible Hinzufügen spezieller Rechte und Eigenschaften.

Auf der linken Seite der Zuordnung stehen die Zielgruppen. Dabei stehen »u« (user) für den Benutzer, »g« (group) für die Gruppe, »o« (other) für die Welt und »a« (all) für alle miteinander. Die Rechte sind wie gehabt als »r«, »w«, »x« und »s« codiert.

Mit einem Pluszeichen werden Rechte hinzugefügt. Das Minuszeichen löscht Eigenschaften und das Gleichheitszeichen weist Eigenschaften zu. Alle nicht explizit veränderten Eigenschaften bleiben unverändert.

Im folgenden Befehl wird der Welt das Leserecht für die Klausur entzogen:

```
$ chmod o-r klausur
```

Der nächste Befehl sorgt dafür, dass der Besitzer der Datei *skript* sein Skript ausführen darf:

```
$ chmod u+x skript
```

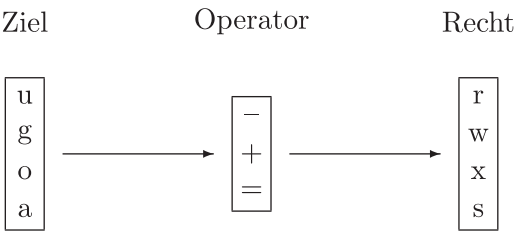

Abbildung 1.1: Bildungsregel für chmod-Optionen

Prozesse

Ein Programm ist eine ausführbare Datei. Als *Prozess* bezeichnet man ein gestartetes Programm. Da ein Programm mehrfach gestartet werden kann, kann es mehrere Prozesse desselben Programms geben.

Zunächst startet das Betriebssystem mit seinem Init-Prozess. Der erhält die Prozessnummer 1. Alle weiteren Prozesse werden irgendwann von diesem oder einem seiner Kindprozesse gestartet. Somit ergibt sich ein Baum von Prozessen.

Prozesse haben wie Dateien Besitzer. Üblicherweise gehören Prozesse den Benutzern, die sie gestartet haben.

Die Prozessliste

Die Prozesse vermehren sich insbesondere beim Booten wie die Kaninchen. Um die Übersicht zu bewahren, sammelt das Betriebssystem sie in einer langen Liste und erlaubt seinen Nutzern einen Blick darauf. Der Befehl dazu lautet `ps`.

Wenn Sie den Befehl `ps` direkt absetzen, sehen Sie nur die eigenen Prozesse, die Sie in diesem Terminal gestartet haben.

```
$ ps
    PID TTY          TIME CMD
  17151 pts/1    00:00:00 bash
  18723 pts/1    00:00:00 ps
$
```

Das ist sehr übersichtlich. Sie sehen die Shell namens `bash` und den eben abgesetzten Befehl `ps`. Die jeweilige PID ist die Prozessnummer und die Zahlen lassen erahnen, wie viele Prozesse seit dem letzten Boot bereits gestartet worden sind. Wenn Sie die anderen Prozesse auch alle sehen wollen, oder wenigstens einen Teil davon, müssen Sie Optionen angeben. Davon hat `ps` reichlich und die können Sie alle mit dem Befehl `man ps` kennenlernen.

Zwei Kombinationen von Optionen sind unter Administratoren besonders beliebt: `ps -ef` und `ps -aux`. Beide zeigen alle Prozesse, die auf dem System laufen, darum ist es sinnvoll, deren Ergebnisse nach `less` umzuleiten. Schauen wir uns mal die ersten vier Zeilen an.

```
$ ps –ef | less
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 Mär24 ?       00:00:02 /sbin/init splash
root           2       0  0 Mär24 ?       00:00:00 [kthreadd]
root           3       2  0 Mär24 ?       00:00:00 [rcu_gp]
root           4       2  0 Mär24 ?       00:00:00 [rcu_par_gp]
```

Sie sehen den ersten Prozess `init` mit der PID 1. Die Kopfzeile zeigt unter anderem die folgenden Informationen an:

UID der Benutzer, dem der Prozess gehört.

PID die Prozessnummer, unter der der Prozess erreichbar ist.

PPID die »Parent«-PID, also die Prozessnummer des Elternprozesses.

TIME die Anzahl der CPU-Sekunden, die der Prozess verbraten hat.

CMD der Befehl, mit dem der Prozess gestartet wurde. Da diesen jeder sehen kann, sollte kein Passwort in einem Befehl erscheinen

Auch die Option `–aux` zeigt alle Prozesse, allerdings fehlt vor allem die PPID. Dafür zeigt diese Option Informationen über den Speicherverbrauch und wird konkreter bei der CPU-Auslastung.

```
$ ps –aux | less
USER  PID %CPU %MEM    VSZ   RSS TTY STAT START  TIME COMMAND
root    1  0.0  0.0 174832 11920 ?   Ss   Mär24  0:02 /sbin/init splash
root    2  0.0  0.0      0     0 ?   S    Mär24  0:00 [kthreadd]
root    3  0.0  0.0      0     0 ?   I<   Mär24  0:00 [rcu_gp]
```

Die Prozentzahlen bei CPU und auch bei MEM, also dem Hauptspeicher, sind sehr viel aussagekräftiger bezüglich der Systemlast als TIME, die ja eine Summe über die Laufzeit darstellt.

Zusätzlich liefert `ps –aux` VSZ für den momentanen virtuellen und RSS für den derzeitigen realen Speicher des Prozesses, also dem im Hauptspeicher.

Nicht immer mit Tötungsabsicht: kill

Als Administrator möchten Sie vielleicht hin und wieder mit Ihren Prozessen sprechen. Das geschieht über den etwas martialisch klingenden Befehl `kill`. Trotz des Namens sendet der Befehl `kill` eigentlich nur ein Signal.

Nun gibt es sehr unterschiedliche Signale. Programme können einige Signale fangen und darauf reagieren. Tun sie es nicht, werden sie terminieren. Hier nun die wichtigsten Signale:

- ✔ SIGTERM: Das ist das Standardsignal, das `kill` sendet, wenn nichts anderes angegeben wurde. Der Prozess wird damit zur Terminierung aufgefordert. Der Prozess darf den Aufruf fangen, um noch schnell sein Testament zu machen und wichtige Daten zu retten.

- ✔ SIGINT: Dieses Signal senden Sie einem gerade gestarteten Prozess, wenn Sie die Tastenkombination Strg+C drücken.
- ✔ SIGKILL: Dies ist ein definitiver Tötungsaufruf, den der Prozess auch nicht verweigern kann.
- ✔ SIGHUP: Dieses Signal erhält der Prozess, wenn das Terminal abgeschaltet wurde oder wenn die Verbindung per `ssh` abgebrochen wurde. Es kann durch Setzen des Befehls `nohup` abgefangen werden. Dasselbe Signal wird von Servern oft dahin gehend interpretiert, dass sie ihre Konfiguration neu einlesen sollen.
- ✔ SIGSTOP und SIGTSTP: Der Prozess wird angehalten und eingefroren, aber nicht beendet. Dieses Signal wird auch durch die Tastenkombination Strg+Z für den aktuell laufenden Prozess ausgelöst. Ein Prozess kann sich dem nicht entziehen.
- ✔ SIGCONT: Ein mit SIGSTOP angehaltener Prozess darf fortfahren. Sie können dieses Signal von der Konsole an durch Strg+Z angehaltene Prozesse vom Terminal aus senden. Der Befehl `bg` setzt den Prozess dann im Hintergrund fort, der Befehl `fg` im Vordergrund.

Die Signalbezeichnung kann dem `kill`-Befehl durch ein vorangestelltes Minuszeichen als Option mitgegeben werden. Das Argument ist die PID des Prozesses, dem das Signal gesendet werden soll.

Im Beispiel wird der Prozess mit PIS 55908 zunächst gestoppt. Anschließend lässt man ihn wieder laufen, um ihn zum Schluss dazu aufzufordern zu terminieren.

```
$ kill –SIGSTOP 55908
$ kill –SIGCONT 55908
$ kill –SIGTERM 55908
```

Benutzerkonten

Linux ist durch seine UNIX-Wurzeln ein Mehrbenutzersystem. Die Rechte des einzelnen Benutzers sind beschränkt. Nur der Administrator ist allmächtig und heißt *root*. Aber darauf kommen wir noch zurück.

Die Benutzer eines Linux-Systems werden in der Datei */etc/passwd* gepflegt, die so heißt, weil sie ursprünglich auch die Passwörter enthielt. Die Passwörter sind inzwischen in die Datei */etc/shadow* ausgelagert worden, die nicht für jedermann zugänglich ist. Die Datei */etc/passwd* ist für jeden Benutzer des Systems lesbar, aber nur für root veränderbar.

Aufbau der Datei /etc/passwd

In den alten Tagen, als Administratoren noch harte Kerle waren, wurde die Datei */etc/passwd* noch mit dem Editor gepflegt und die Umgebung für die Benutzer in

liebevoller Handarbeit angelegt. Heute gibt es dafür Werkzeuge. Das ist zwar nicht so romantisch, aber dafür vergisst man nichts. Dennoch hilft ein Blick in die *passwd*-Datei, um zu erfahren, was alles für einen Benutzer eingetragen wird. Eine Zeile hat den folgenden Aufbau:

```
Name:Passwort:User-ID:Group-ID:Kommentar:Verzeichnis:Shell
```

Auf meinem Linux-Notebook steht dort für mich folgender Eintrag:

```
arnold:x:1000:1000:Arnold Willemer,,,:/home/arnold:/bin/bash
```

Die Einträge dieser Zeile sind durch Doppelpunkte getrennt.

- ✔ Mein Benutzername ist arnold. Man könnte auch willemer, arwi, awillemer oder welche Art der Namensgebung am besten gefällt nehmen. Groß- und Kleinschreibung sind relevant. Darum ist es sinnvoll, nur kleine Buchstaben zu nehmen. Das schreibt sich einfach schneller.
- ✔ Das x an der Stelle des Passworts deutet schon an, dass es hier nicht steht. Tatsächlich gibt es dafür die Datei */etc/shadow*.
- ✔ Die erste Zahl ist die Benutzer-ID. Für die normalen Anwender beginnt diese bei 1000. Diese Nummer wird beispielsweise dazu verwendet, um zu codieren, welchem Benutzer eine Datei gehört. Darum kann eine Datei, die ich auf einem USB-Stick abgelegt habe, auf Ihrem Computer plötzlich unter Ihrem Benutzernamen auftauchen, wenn Sie dort auch die Benutzer-ID 1000 haben. Eine Benutzer-ID ist also nur auf der Computerebene eindeutig.
- ✔ Die zweite Zahl ist die Gruppe, zu der der Benutzer gehört. Prinzipiell kann ein Benutzer mehreren Gruppen angehören. Aber die Gruppe, die hier steht, ist seine Hauptgruppe. Dateien, die dieser Benutzer anlegt, werden dieser Gruppe zugeordnet. Debian und Ubuntu legen für jeden Benutzer zunächst eine eigene Gruppe an, die den gleichen Namen und die gleiche ID wie der Benutzer hat.
- ✔ Es folgt ein Kommentarfeld, in dem oft der korrekte Name des Benutzers eingetragen wird. Zwischen die Kommata können weitere Daten eingegeben werden.
- ✔ Es folgt der Pfad des Heimatverzeichnisses des Benutzers. Gern wird dafür ein Verzeichnis unterhalb des Pfades */home* verwendet.
- ✔ Der letzte Eintrag beschreibt das Programm, das gestartet wird, wenn der Benutzer sich anmeldet. Bei mir ist es die Linux-Shell bash. Steht hier */bin/false*, wird sich der Benutzer nicht einloggen können. Dies ist bei Dienstprogrammen üblich.

Benutzer anlegen und löschen

Für das Anlegen eines Benutzers gibt es die Programme `useradd` und `adduser`. Allerdings empfiehlt die Manpage von `useradd` die Verwendung von `adduser`, und das zu Recht. Denn tatsächlich erledigt `adduser` alles auf einmal. Es bearbeitet die Konfigurationsdateien, erzeugt das Benutzerverzeichnis, legt die Standarddateien dort an und erfragt das Passwort und ein paar Informationen für das Kommentarfeld in der Datei */etc/passwd*.

Als Beispiel legen wir einen Benutzer namens `paul` an:

```
# adduser paul
Lege Benutzer »paul« an ...
Lege neue Gruppe »paul« (1002) an ...
Lege neuen Benutzer »paul« (1002) mit Gruppe »paul« an ...
Erstelle Home-Verzeichnis »/home/paul« ...
Kopiere Dateien aus »/etc/skel« ...
Geben Sie ein neues UNIX-Passwort ein:
Geben Sie das neue UNIX-Passwort erneut ein:
passwd: Passwort erfolgreich geändert
Benutzerinformationen für paul werden geändert.
Geben Sie einen neuen Wert an oder drücken Sie ENTER für den Standardwert
        Vollständiger Name []: Paul
        Zimmernummer []: 00
        Telefon geschäftlich []: 112
        Telefon privat []: 110
        Sonstiges []: netter Kerl
Sind diese Informationen korrekt? [J/n]
```

Sein Eintrag sieht dann so aus:

```
paul:x:1002:1002:Paul,00,112,110,netter Kerl:/home/paul:/bin/bash
```

Auch das Verzeichnis */home/paul* wurde so angelegt, dass sich Paul sofort anmelden kann.

Wenn die Benutzerverwaltungsprogramme ein neues Benutzerverzeichnis anlegen, werden auch bestimmte Dateien oder Verzeichnisse vorgegeben. Das Muster dafür finden die Programme im Verzeichnis */etc/skel*. Mit einem einfachen `ls` werden Sie darin nicht viel sehen. Erst bei Verwendung von `ls -a` sehen Sie die Konfigurationsverzeichnisse, die mit einem Punkt beginnen.

Analog existiert auch der Befehl `deluser` zum Löschen des Benutzerkontos:

```
# deluser paul
Entferne Benutzer »paul« ...
Warnung: Die Gruppe »paul« hat keine Mitglieder mehr.
Fertig.
```

Auch die Gruppe `paul` wurde gelöscht, weil sie außer `paul` keine weiteren Mitglieder mehr hatte. Das Benutzerverzeichnis und die von Paul zwischenzeitlich vielleicht angelegten Dateien muss der Administrator von Hand aufräumen.

Passwörter verwalten

Jeder Benutzer kann sein Passwort mit dem Befehl `passwd` selbst ändern, und das sollte er auch tun.

```
$ passwd
Ändern des Passworts für paul.
(aktuelles) UNIX-Passwort:
Geben Sie ein neues UNIX-Passwort ein:
```

```
Geben Sie das neue UNIX-Passwort erneut ein:
passwd: Passwort erfolgreich geändert
$
```

Allerdings soll es schon vorgekommen sein, dass ein Benutzer sein Passwort vergessen hat. Dann wird er hoffen, dass der Administrator ihm helfen kann. Und welch ein Glück: Er kann.

Dazu benutzt der Administrator als root den gleichen Befehl `passwd`, gibt aber als Argument den Benutzernamen an. root wird nicht nach dem bisherigen Passwort gefragt. Er kann einfach ein neues Passwort einsetzen.

```
# passwd paul
Geben Sie ein neues UNIX-Passwort ein:
Geben Sie das neue UNIX-Passwort erneut ein:
passwd: Passwort erfolgreich geändert
#
```

Die Datei /etc/shadow

Wie schon erwähnt wird das Passwort nicht in der Datei */etc/passwd* gespeichert. Stattdessen befindet es sich in der Datei */etc/shadow*, allerdings nicht im Klartext, sondern verschlüsselt. Als `root` dürfen Sie hineinschauen, als normaler Benutzer nicht.

```
paul:$6$Tx...Vtwt:16021:0:99999:7:::
```

Da, wo oben die drei Punkte stehen, steht in der Datei etwa eine Zeile wilder Zeichen. Bei aller Liebe zu komplizierten Passwörtern: So lang war das Passwort von paul nun auch wieder nicht.

Sie sehen noch viele Doppelpunkte in der Zeile. Tatsächlich kann mit der Shadow-Datei das Benutzerkonto noch weiter eingeschränkt werden. Die Struktur einer Zeile in der Datei */etc/shadow* hat folgenden Aufbau:

```
Name:Passwort:Zuletzt:Erstmalig:Wechselfrist:Warnfrist:Ablauf:Sperrung
```

Sie können in den Einträgen noch weitere Einstellungen durchführen.

- ✔ Der `Name` stimmt mit dem Benutzernamen in der Datei */etc/passwd* überein.
- ✔ Den zweiten Eintrag habe ich ja schon als verschlüsseltes Passwort vorgestellt. Übrigens ist es ein sogenannter Falltüralgorithmus. Das heißt, Sie können aus dem verschlüsselten Passwort nicht mehr das Passwort im Klartext ermitteln. Bei der Anmeldung wird das eingegebene Passwort in gleicher Weise verschlüsselt und geprüft, ob das Ergebnis übereinstimmt.
- ✔ Die Zahl *Zuletzt* gibt an, wann das letzte Mal das Passwort geändert wurde. Codiert wird dieser Zeitpunkt in der Anzahl der Tage seit dem 1.1.1970.
- ✔ Die Zahl *Erstmalig* gibt an, wie viele Tage das Passwort mindestens unverändert bleiben muss.

- ✔ In der Spalte *Wechselfrist* kann festgelegt werden, nach wie vielen Tagen das Passwort mindestens geändert werden muss.
- ✔ Damit der Benutzer nicht überraschend mit einem nicht mehr funktionierenden Passwort unterwegs ist, wird mit der Zahl *Warnfrist* angegeben, nach wie vielen Tagen der Benutzer gewarnt wird, dass sein Passwort abläuft.
- ✔ Das nächste Feld *Ablauf* gibt an, wann das Passwort dann endgültig ungültig wird.
- ✔ Das letzte Feld gibt den Zeitpunkt an, wann das Benutzerkonto ausläuft. Dann hilft dem Benutzer auch das Wechseln des Passwortes nicht mehr.

Gruppen verwalten

Linux ermöglicht mit dem Gruppenkonzept eine Kooperation mehrerer Benutzer mit gleichen Dateien und Verzeichnissen. Die Gruppen erlauben es aber auch, bestimmte Privilegien der Benutzer zu verwalten.

Die Gruppen werden in der Datei */etc/group* verwaltet. Jede Zeile definiert eine Gruppe und beginnt mit dem Gruppennamen. Es folgen ein Passwort, das meist leer bleibt, die eindeutige Group-ID und dann die Liste der Mitglieder, jeweils durch ein Komma getrennt:

```
Gruppenname:Passwort:Gruppen-ID:Liste der Mitglieder
```

Wenn Sie einer Gruppe ein weiteres Mitglied gönnen wollen, hängen Sie dessen Benutzernamen einfach hinten an. Bei mehreren Mitgliedern achten Sie bitte darauf, dass die Mitglieder durch Kommata getrennt sind.

Eine neue Gruppe erzeugen Sie einfach durch Kopieren einer Zeile. Ändern Sie den Gruppennamen in den gewünschten. Die Gruppen-ID muss eindeutig sein.

Gruppenpasswort: gpasswd

Normalerweise werden keine Passwörter für Gruppen vergeben. Sollten Sie dennoch eines einrichten wollen, hilft Ihnen der Befehl `gpasswd`. Mit diesem Befehl lassen sich aber auch Mitglieder zu einer Gruppe hinzufügen. Zunächst kann root über die Option `-A` einem Mitglied die Administrationsrechte für die Gruppe zuteilen:

```
# gpasswd -A georg testgruppe
#
```

Nun kann georg die Gruppe testgruppe verwalten. So kann er beispielsweise seinen Freund paul zur Gruppe hinzufügen:

```
$ gpasswd -a paul testgruppe
Benutzer paul wird zur Gruppe testgruppe hinzugefügt.
$
```

Mit der Option `-d` kann er seinen Freund wieder austragen.

Gruppenverwaltung: chgrp

Mit dem Befehl `chgrp` kann paul Dateien, die ihm gehören, nun einer anderen Gruppe zuteilen, in der er Mitglied ist:

```
$ chgrp testgruppe datei.txt
```

Welche Rechte die Mitglieder der Gruppe an seiner Datei bekommen, steuert paul mit dem Befehl `chmod`, wie das in Kapitel 1 beschrieben ist.

Der Benutzer ist normalerweise immer in seiner Hauptgruppe aktiv, die in seinem Eintrag der Datei */etc/passwd* eingetragen ist. Will paul ab sofort Dateien anlegen, die der Gruppe testgruppe zugehören, kann er den Befehl `newgrp` verwenden:

```
$ newgrp testgruppe
```

Kurzfristig den Benutzer wechseln: su

Mit dem Befehl `su` (set user) wechseln Sie den Benutzer. In Kapitel 1 haben Sie gesehen, wie Sie damit zum Administrator werden. Aber tatsächlich kann der Befehl Ihre Identität auf jeden Benutzer wechseln. Dazu geben Sie den Benutzernamen der gewünschten Identität als Argument. Natürlich wird der Wechselwillige nach dem Passwort der Zielperson gefragt. Es könnte ja sonst jeder kommen.

```
$ su paul
Passwort:
$
```

Allerdings gibt es eine Ausnahme. Wenn `root` wechseln will, braucht er kein Passwort. Dies bietet die Möglichkeit, Benutzer anzulegen, die gar kein zulässiges Passwort besitzen. Das ist praktisch, wenn ein Server einen eigenen Benutzer verwenden soll. Der Server wird unter diesem Benutzer gestartet, aber niemand kann dessen Passwort ausspionieren. Ein Passwort gibt es nicht und die Shell in der Datei */etc/passwd* ist auf `/bin/false` gesetzt.

Sollte man doch einmal unter diesem Benutzer Wartungsarbeiten machen müssen, kann der Administrator in dessen Rolle schlüpfen.

Wird der Befehl `su` ausgeführt, wechselt der Benutzer zwar die Identität, erhält also dessen Rechte. Er durchläuft aber nicht den kompletten Login, der beispielsweise dessen Login-Shell startet und die Startskripte ausführt. Will man dies erreichen, so muss man zusätzlich ein einzelnes Minuszeichen angeben.

```
$ su - paul
Passwort:
$
```

Geben Sie keine Benutzerkennung als Argument, wird der Benutzer zum `root`. Ohne Minuszeichen erhält er dessen Rechte, mit Minuszeichen erhält er die komplette Umgebung

des Administrators. Dazu gehören beispielsweise auch die Pfade zu dessen Werkzeugen in den Verzeichnissen */sbin* und */usr/sbin*.

Sie können durch den Befehl `exit` wieder zu Ihrer alten Identität zurückkehren.

Administrationsaufgaben starten: sudo

Der Befehl `sudo` wird einem Befehl vorangestellt, der unter Administratorrechten laufen soll. Das darf nicht jeder Benutzer.

Bei einer Standardinstallation von Ubuntu ist es beispielsweise nur der erste Benutzer, der angelegt wird. Er darf jedem beliebigen Befehl ein `sudo` voranstellen und dieser wird unter Administratorrechten ausgeführt. Vorher wird das Passwort des Benutzers abgefragt, der sich die Administrationsrechte wünscht.

```
arnold@server ~ $ sudo apt update
[sudo] Passwort für arnold:
```

Wenn dies ein anderer Benutzer versucht, wird er abgewiesen.

```
gast@server:~$ sudo apt update
[sudo] Passwort für gast:
Leider darf der Benutzer gast »/usr/local/bin/apt update«
als root auf server nicht ausführen.
```

Wollen Sie weiteren Benutzern das Recht auf `sudo` zugestehen, müssen Sie diesen in der Gruppe `sudo` eintragen.

```
$ sudo usermod –aG sudo gast
```

Alternativ können Sie den Benutzer `gast` auch direkt in der Datei */etc/group* eintragen. Das Ergebnis ist dasselbe:

```
sudo:x:27:arnold,gast
```

Sobald sich der Benutzer `gast` das nächste Mal anmeldet, hat er das Recht, Befehle als Administrator aufzurufen, indem er den Befehl `sudo` voranstellt.

Hintergrund

Ursprünglich diente der Befehl `sudo` dazu, einzelne Administratoraufgaben delegieren zu können. In der Datei */etc/sudoers* kann dazu genau festgelegt werden, welche Kommandos welchen Benutzern erlaubt sind.

Die dazu berechtigten Benutzer können dann die festgelegten Kommandos durch Voranstellen des Befehls `sudo` ausführen. Zur Authentifizierung müssen Sie Ihr eigenes Passwort eingeben. Stimmt die Berechtigung, wird die Zeile anschließend so ausgeführt, als hätte sie der Administrator aufgerufen. Der ganze Vorgang wird protokolliert, sodass ein Missbrauch nachweisbar ist.

Auf diese Weise kann der Administrator die Datensicherung an einen Kollegen delegieren und die Installation von Software an einen anderen.

Der Befehl `visudo` editiert speziell die Datei */etc/sudoers* und sorgt gleichzeitig dafür, dass die Datei nicht mehrfach parallel geöffnet wird. Falls Sie jetzt Bauchschmerzen bekommen, weil Sie den Editor `vi` nicht leiden können, kann ich Sie beruhigen. Trotz des Namens ruft `visudo` den Editor auf, den Sie in der Umgebungsvariablen `EDITOR` hinterlegt haben. Steht dort nichts, verwendet Linux `nano` in der Annahme, dass damit auch Anfänger klarkommen.

Eine Zeile in der Datei beginnt mit dem Benutzer oder einer Gruppe. Dann wird der Rechnername genannt, auf dem die Zeile gelten soll. Es folgen ein Gleichheitszeichen und dann das erlaubte Kommando:

```
paul  server = /home/paul/bin/dasi
```

Diese Zeile besagt, dass der Benutzer `paul` berechtigt ist, das Kommando `/home/paul/bin/dasi` mit vorangestelltem Befehl `sudo` aufzurufen, sodass es mit den Rechten von root läuft. Anders ausgedrückt: `paul` darf die Datensicherung ausführen. Und wenn er das tut, sieht das so aus:

```
$ sudo /home/paul/bin/dasi
[sudo] password for paul:
```

Sie könnten aber auch eine Gruppe `dasi` für diese Aufgabe einrichten. Wenn Sie dann `dasi` statt `paul` eintragen, können Sie durch Hinzufügen von `georg` und `inga` dafür sorgen, dass beide die Datensicherung starten dürfen.

In jeder *sudoers*-Datei finden Sie eine Zeile, die festlegt, dass `root` auf allen Rechnern alles darf:

```
root    ALL=(ALL) ALL
```

Privilegierung durch Dateirechte

Es gibt die Möglichkeit, einem Benutzer die Rechte eines anderen Benutzers zuzuordnen, indem der Programmdatei zugewiesen wird, unter welchem Benutzer sie laufen soll, ganz gleich, von wem sie aufgerufen wurde.

1. Zunächst muss die Programmdatei natürlich ausführbar sein, also ein x in den Rechten tragen.
2. Dann muss die Programmdatei demjenigen gehören, unter dessen Rechten sie laufen soll.
3. Schließlich muss der Datei das Set-User-ID-Bit zugewiesen werden, das noch vor die drei Rechtegruppen mit `chmod` gesetzt wird.

Im folgenden Beispiel soll ein Vokabelprogramm unter dem Benutzer `vokuser` laufen, der zuvor natürlich angelegt worden sein muss.

```
$ ls -l vokabel
-rwxrwxr-x 1 arnold arnold 239232 Okt  7  2019 vokabel
$ sudo chown vokuser vokabel
[sudo] Passwort für arnold:
$ sudo chmod 4755 vokabel
$ ls -l vokabel
-rwsr-xr-x 1 vokuser arnold 239232 Okt  7  2019 vokabel
```

Durch den Befehl `chmod 4755` wird das Set-User-ID-Bit gesetzt, das in der Rechtegruppe durch ein `s` anstelle des ersten `x` in den Rechten für die Welt erscheint.

Wenn nun der Benutzer `paul` das Programm startet, wird es dennoch als `vokuser` laufen. Erstellt das Programm Dateien, werden diese dann `vokuser` gehören und nicht `paul`.

Diese Art, Rechte zu vergeben, ist durchaus riskant. Darum ist es nicht möglich, Skripte mit dem Set-User-ID-Bit auszuführen.