

4.2 Ein Blick in /etc

Die meisten Konfigurationsdateien Ihres System befinden sich im /etc-Verzeichnis, und da einiges an Softwarepaketen für das System zusammenkommt, tummeln sich dort eine Menge Dateien. Das macht es allerdings nicht ganz einfach festzustellen, was nun für ein lauffähiges System notwendig ist.

In den vorangegangenen Kapiteln sind Ihnen für den Boot-Vorgang notwendige Dateien wie `inittab` und `fstab` begegnet. In Tabelle 4–1 finden Sie erst einmal eine Liste von wichtigen Systemkonfigurationsdateien inklusive Referenzen auf die erläuternden Abschnitte in diesem Buch.

Datei/Verzeichnis	Funktion	Abschnitt
<code>fstab, mtab</code>	Dateisysteme	2.4.6
<code>group</code>	Benutzerverwaltung	4.3.2
<code>hosts</code>	Netzkonfiguration	5.6
<code>init.d</code>	Boot-Sequenz	3.1.3
<code>inittab</code>	Boot-Sequenz	3.1.1
<code>ld.so.conf, ld.so.cache</code>	Dynamische Bibliotheken	8.1.4
<code>nsswitch.conf, resolv.conf</code>	Netzkonfiguration	5.6
<code>sysconfig/</code>	Systemkonfiguration	5.4
<code>X11/</code>	X-Window-Konfiguration	11.3.2
<code>rc?.d</code>	Boot-Sequenz	3.1.2
<code>passwd, shadow</code>	Benutzerverwaltung	4.3

Tab. 4–1 Lebenswichtige Konfigurationsdateien in /etc/

4.3 Dateien zur Benutzerverwaltung

Unix ermöglicht es mehreren Benutzern, sich anzumelden und auf dem Rechner zu arbeiten, ohne sich gegenseitig in die Quere zu kommen. Auf unterster Ebene sind Benutzer nichts anderes als Zahlen (*user IDs* – Benutzer-Identifikationsnummern). Die *Benutzernamen* bzw. Anmeldekennungen (*login names*) helfen uns Menschen ein wenig, die Benutzer-IDs besser zu handhaben und zu verstehen.

Die Textdatei `/etc/passwd` bildet Benutzernamen auf Benutzer-IDs ab. Ein Beispiel:

```
root:x:0:0:Superuser:/root:/bin/sh
bin:*:1:1:bin:/bin:/bin/sh
daemon:*:2:2:daemon:/sbin:/bin/sh
sys:*:3:3:sys:/dev:/bin/sh
nobody:*:65534:65534:nobody:/var/lib/nobody:/bin/false
otto:x:3519:1000:Otto Normalanwender:/home/otto:/bin/bash
cdata:x:543:1000:Commander Data:/home/andriod:/bin/zsh
```

Das Format ist ziemlich einfach aufgebaut: Jede Zeile hat sieben durch Doppelpunkte getrennte Felder und repräsentiert einen Benutzer. Die Felder haben im Einzelnen folgenden Inhalt:

- Benutzername (Login)
- Verschlüsseltes Passwort des Benutzers. Auf heutigen Linux-Systemen steht das Passwort nicht in `passwd`, sondern in `shadow`. Das Format von `shadow` ähnelt grob dem von `passwd`. Normale Benutzer haben jedoch keine Leseberechtigung von `shadow`. Das verschlüsselte Passwort im zweiten Feld von `shadow` oder `passwd` – unter Unix sind Systempasswörter nie unverschlüsselt – sieht ein wenig aus, als ob ein kleines Tier über die Tastatur gelaufen ist, z. B. `d1CvEWiB/oppC`. Der Rest der Felder in `shadow` hat mit Dingen wie dem Passwortverfallsdatum zu tun.
Ein »x« im zweiten Feld von `passwd` besagt, dass das Passwort in `shadow` gespeichert ist. Ein »*« oder »!« bedeutet, dass der Benutzer sich nicht anmelden kann. Ist das zweite Feld leer – sieht es also wie »:« aus –, dann heißt das, dass bei der Anmeldung dieses Benutzers kein Passwort verlangt wird. Passen Sie auf, dass so etwas nicht auf Ihrem System vorkommt. Sie sollten niemals einen Benutzer ohne Passwort haben!
- Benutzer-ID (*UID*). Die UID gibt an, wie der Benutzer dem Kernel gegenüber repräsentiert wird. Die Zahl sollte nur einmal vergeben werden. Es ist zwar prinzipiell möglich, zwei Benutzer mit derselben ID zu haben, aber es wird Sie und sicherlich einige Softwarepakete durcheinander bringen.
- Gruppen-ID (*GID*, *group ID*). Diese Zahl sollte einem der Einträge (in Feld 3) aus `/etc/group` entsprechen. Unix-Gruppen sind wichtig für Dateirechte, sonst spielen sie aber eher keine Rolle. Die Zahl in `passwd` gibt die *primäre Gruppe* des Benutzers an.
- Der richtige Name des Benutzers. Dieses Feld wird häufig *GECOS*⁴-Feld genannt. Manchmal tauchen hierin Kommata auf, die zum Trennen von Raum-, Telefon- u. a. Nummern benutzt wird.
- Heimatverzeichnis des Benutzers
- Login-Shell des Benutzers. Also das Programm, das läuft, wenn der Benutzer sich auf der Textkonsole angemeldet hat.

4. General Electric Comprehensive Operating System, ein historisches Relikt.

Warnung: Der Aufbau von `/etc/passwd` ist immer exakt wie eben beschrieben und muss strikt eingehalten werden. Die Dateistruktur erlaubt auch keine Shell- oder anderen Kommentarzeichen oder Leerzeilen. Unbedachte Experimente an dieser Datei können dazu führen, dass Sie sich aus Ihrem eigenen System aussperren.

Folgende Abbildung veranschaulicht noch einmal die verschiedenen Felder eines Benutzers aus dem vorangegangenen Beispiel:

```
otto:x:3519:1000:Otto Normalanwender:/home/otto:/bin/bash
```

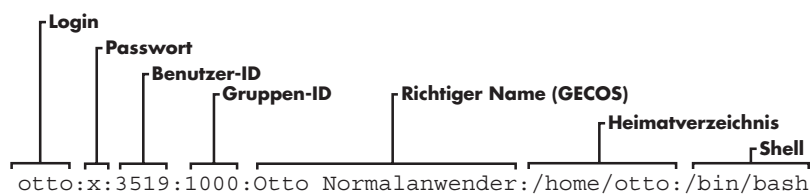


Abb. 4-1 Eintrag in `/etc/passwd`

Hinweis: Der Benutzername und das Heimatverzeichnis zusammen werden häufig als *Account* bezeichnet.

Es gibt ein paar besondere Benutzer in `/etc/passwd`. Root hat unter Linux wie in dem Beispiel oben immer die UID 0 und GID 0. Es existieren eine Reihe weiterer Benutzer ohne Anmeldeprivilegien wie z. B. `daemon` und `bin`. Diese variieren von System zu System und sollten von Ihnen nicht »angefasst« werden. `nobody` stellt einen speziellen unterprivilegierten Benutzer dar, unter dessen Account einige Prozesse vom System gestartet werden. All diese Pseudobenzutzer, die sich nicht anmelden können, existieren aus Sicherheitsgründen. Prozesse, die unter solchen Benutzer-IDs laufen, tun das nur im Kontext des Pseudobenzutzers und haben daher nur begrenzte Lese- oder Schreibberechtigung sowie andere Privilegien im System.

4.3.1 Änderung der Benutzerdaten und Passwörter

Reguläre Benutzer führen Änderungen an ihrer eigenen Zeile der `/etc/passwd` bzw. `/etc/shadow` mittels des Kommandos `passwd` durch. Ohne weitere Parameter aufgerufen, ändert `passwd` das eigene Passwort. Man kann aber auch den Parameter `-f` benutzen, um das GECOS-Feld zu verändern, oder `-s`, um die Shell zu wechseln (die gültigen Shells finden Sie in `/etc/shells`). Das System benutzt dafür `chfn` (*change finger information*) oder `chsh` (*change shell*), die Sie ebenso direkt aufrufen können. Diese Kommandos sowie `passwd` sind *suid-root*-Programme, d. h., sie laufen mit Root-Rechten, um `/etc/passwd` oder `/etc/shadow` ändern zu können.

/etc/passwd als Root ändern

Da `/etc/passwd` eine einfache Textdatei ist, kann Root prinzipiell irgendeinen Texteditor benutzen, um Änderungen vorzunehmen. Um einen Benutzer aufzunehmen, müssten Sie nichts anderes tun, als eine passende Zeile hinzuzufügen und ein Heimatverzeichnis für den Benutzer anzulegen. Das Entfernen eines Benutzers funktioniert demzufolge umgekehrt. Es gibt sogar ein spezielles Programm für Root namens `vi pw`, das die `passwd`-Datei vorübergehend sperrt (englisch: *lock*). Um als Root einem Benutzer ein anderes Passwort zuzuteilen, benutzt man einfach `passwd user`.

Die direkte Änderung von `/etc/passwd` mit einem Editor sollte jedoch den Fortgeschrittenen vorbehalten bleiben, da hierbei leicht Fehler passieren können. Die meisten Distributionen wie Debian, Suse und Red Hat Linux greifen dem Administrator mit Kommandozeilenprogrammen wie `useradd` und `userdel` unter die Arme, die gleichzeitig noch ein Heimatverzeichnis einrichten und diese von vornherein mit einigen sinnvollen Dateien, darunter Punktdateien, ausstatten. Je nach Distribution können Sie auch grafische Helferlein für die Passwort- und Gruppenverwaltung benutzen. Auch wenn all diese Werkzeuge existieren: Sie sollten trotzdem über den Aufbau von `/etc/passwd` Bescheid wissen, falls Sie später doch in die Verlegenheit geraten sollten, sie manuell zu editieren.

4.3.2 Gruppen unter Unix

Unix-Gruppen bieten eine Möglichkeit, Daten mit bestimmten anderen Benutzern gemeinsam zu verwenden und wiederum anderen Benutzern den Zugriff zu verweigern. Das wird dadurch erreicht, dass man Lese- oder Schreibrechte in den Zugriffsbits (siehe Abschnitt 1.17) für die Gruppe einer Datei setzt. In der Zeit der großen Unix-Rechner, wo sich hundert Benutzer und mehr einen Rechner geteilt haben, war dies sehr wichtig, um unter Arbeitsgruppen Dateien gemeinsam bearbeiten und darauf zugreifen zu können. Da diese Arbeitsgruppen-Dinosaurier zugunsten eher persönlicher *Workstations* nahezu ausgestorben sind, haben Gruppenrechte nicht mehr die Bedeutung, die sie einst hatten.

Wie zuvor erwähnt, gibt es in `/etc/passwd` ein Feld für die Gruppe. Die Datei `/etc/group` definiert die Gruppen-ID:

```
root:x:0:
bin:x:1:daemon
daemon:x:2:
sys:x:3:
adm:x:4:
audio:x:6:otto, cdata
nogroup:x:65534:
user:x:1000:
```

ches. Etwas mehr zu `nsswitch.conf` – der Konfigurationsdatei für den *Name Service Switch* – können Sie Abschnitt 5.6 entnehmen.

4.4 getty und login

In Abschnitt 3.1.1 ist bereits `getty` erwähnt worden, ein Programm, das sich mit einem virtuellen oder echten Terminal verknüpft und den Prompt zum Anmelden präsentiert. Unter Linux dominieren zwei `getty`-Spezies: `getty` und `mingetty`. `getty`-Spezies sind relativ unkompliziert. `mingetty` taucht z.B. in den Zeilen von `/etc/inittab`⁵ auf:

```
4:2345:respawn:/sbin/mingetty tty4
```

Sie sind wahrscheinlich gespannt zu erfahren, welche Datei `getty` Ihnen immer als Begrüßung präsentiert. Dies ist `/etc/issue`. Eine Änderung dieser Datei ist eine schnelle und zufriedenstellende Methode, Ihr System zu verunstalten, ohne ihm wirklich Schaden zuzufügen.

Nachdem Sie Ihre Benutzererkennung eingegeben haben, übernimmt das Programm `login` alle weiteren Aufgaben, wie z. B. die Frage nach dem Passwort. Wenn Sie Ihr Passwort korrekt eingegeben haben, ersetzt wiederum `login` sich selbst am Ende der Prozedur mit Ihrer Shell. Anderenfalls bekommen Sie am Ende eine Meldung wie »Login incorrect« oder »Login inkorrekt«.

Nun, da Sie wissen, was `getty` und `login` bezwecken, kann ich Ihnen auch anvertrauen, dass es äußerst selten vorkommt, dass Sie diese Programme konfigurieren oder deren Optionen in `inittab` ändern müssen.

Hinweis: Falls Sie Faxe senden und empfangen wollen oder sich auf Ihrem Rechner einwählen möchten, werden Sie `mgetty` (nicht `mingetty`) einsetzen müssen. Dies ist ein erweitertes `getty`, das Ergänzungen für den Einsatz von Fax- und Datenmodems beinhaltet.

4.5 Die Uhrzeit stellen

Ein gut funktionierendes Unix-System ist auf eine akkurate Uhrzeit angewiesen. Der Kernel ist verantwortlich für die Systemuhrzeit (*system clock*), die Ihnen angezeigt wird, wenn Sie Kommandos wie z. B. `date` benutzen. Mit `date` kann man auch die Zeit stellen, jedoch ist das keine so gute Idee, da Sie es mit `date` nie exakt hinbekommen, und schließlich wollen Sie, dass die Systemuhrzeit nur eine minimale Abweichung zur korrekten Zeit aufweist.

Jeder PC hat eine batteriegepufferte »Echtzeituhr« (*RTC: real time clock*). Dabei handelt es sich nicht gerade um das ausgefeilteste Stückchen Technik, aber

5. Unter Debian-Linux z. B. wird die Baudrate irrelevanterweise noch für die virtuellen Konsolen als Parameter spezifiziert.

es ist besser als gar nichts. Der Kernel stellt die Zeit beim Start des Systems nach dieser Uhr. Dies kann man ebenso mit dem Befehl `hwclock` während der Laufzeit erledigen. Wenn Sie Ihr System so eingerichtet haben, dass Ihre Hardware-Uhr in *UTC*⁶ (*Coordinated Universal Time = Greenwich Mean Time, GMT*) tickt, geben Sie dieses Kommando ein:

```
hwclock -hctosys -utc
```

Sie sollten versuchen, die Hardware-Uhr nach UTC-Zeit zu betreiben. Das erspart Ihnen Ärger bei den Zeitumstellungen für Sommer- und Winterzeit. Wenn Sie die »Echtzeituhr« trotz dieser Warnung in lokaler Uhrzeit ticken lassen, müssten Sie Folgendes eingeben:

```
hwclock -hctosys -localtime
```

Ebenso funktionieren die Befehle in umgekehrter Richtung: Wenn also die Hardware-Uhr nach dem Mond gehen sollte, kann die Uhrzeit mit dem Parameter `-systohc` nach der Systemzeit gestellt werden. Dies empfiehlt sich besonders, wenn Sie einen NTP-Server als Zeitreferenz benutzen (siehe Abschnitt 4.5.2).

Dummerweise ist der Kernel auch nicht fürchterlich gut darin, die Zeit korrekt zu halten. Da Unix-Maschinen nicht selten Monate, manche sogar Jahre ohne Unterbrechung laufen, tendiert die Systemzeit dazu, mehr oder weniger zu driftet. Es wäre dann etwas plump, die Systemzeit alle naselang mit `hwclock` zu korrigieren: Die Zeitsprünge würden zeitgesteuerte Abläufe durcheinander bringen oder gar Datenkonsistenz gefährden. Für graduelle Änderungen empfiehlt sich daher:

```
adjtimex --adjust
```

Dies ist jedoch nur für manuelle Korrekturen geeignet und doktert nur an der Auswirkung herum, denn die Zeit ist schon gedriftet. Um immer eine korrekte Systemzeit zu haben, sollte man am besten die Systemuhr nach einer Zeitreferenz aus dem Netz stellen (siehe übernächsten Abschnitt 4.5.2).

4.5.1 Zeitzonen

Die Systemzeit tickt in der UTC-Zeitzone (*Coordinated Universal Time*). Die Systembibliotheken konvertieren diese Zeit in die lokale Zeit und nehmen Korrekturen für Sommerzeit vor.

6. Für aufmerksame Leser: Die Abkürzung UTC hat keinen Buchstabendreher.

Um die Systemzeitzone zu ändern oder zu setzen, legen Sie einen symbolischen Link von der gewünschten Zeitzonendatei unterhalb von `/usr/share/zoneinfo` auf `/etc/localtime` an:

```
ln -sf /usr/share/zoneinfo/Europe/Berlin /etc/localtime
```

Innerhalb von Deutschland entspricht dies der richtigen Vorgehensweise. Eine ausführliche Anzeige des Links mit `ls -l` sieht dann so aus:

```
/etc/localtime -> /usr/share/zoneinfo/Europe/Berlin
```

Die wenigsten Linux-Distributionen benutzen beim Einrichten während der Installation symbolische Links, wodurch die gegenwärtige Einstellung am besten erkennbar wäre. Manche kopieren die Datei (Red Hat), manche benutzen Hardlinks (Suse-Linux).

Falls Sie eine andere Zeitzone als die vom System vorgegebene wünschen, können Sie so vorübergehend die Zeitzonevariable setzen bzw. ändern:

```
export TZ=Pacific/Marquesas  
date
```

4.5.2 NTP

Falls Ihr Computer eine permanente Verbindung zum Internet hat, empfiehlt es sich, einen *NTP*-Dämon (*Network Time Protocol*) zum Halten der Systemzeit zu verwenden. Die NTP-Webseite finden Sie unter <http://www.ntp.org>. Wenn Sie nicht die dort stapelweise vorhandene Dokumentation lesen möchten und eine pragmatischere Vorgehensweise bevorzugen, machen Sie Folgendes:

1. Fragen Sie Ihren Internet Service Provider (ISP) nach dem bzw. den nächsten NTP-Server(n).
2. Tragen Sie diese(n) in `/etc/ntp.conf` ein.
3. Stellen Sie sicher, dass während des Systemstarts das Kommando `ntpd` ausgeführt wird.
4. Anschließend starten Sie `ntpd` jedes Mal beim Booten.

Schritt 3 und 4 sollten in Form eines `init`-Skriptes geschehen (siehe Abschnitt 3.1), falls Ihre Distribution nicht schon dieses Skript dort verankert hat. Einige Distributionen erlauben auch die grafische Konfiguration des NTP-Dienstes. Suse-Linux geht sogar so weit, Ihnen einen dem Land entsprechenden NTP-Server vorzuschlagen.

4.6 Wiederkehrende Aufgaben mit Cron erledigen

Jedes Unix hat einen üblicherweise aktivierten Dämon, der sich `cron` nennt und der Programme zu vorbestimmten Zeiten nach einem bestimmten Rhythmus starten kann. Für erfahrene Systemadministratoren stellt Cron einen lebenswichtigen Bestandteil des Systems dar, weil er regelmäßige automatisierte Jobs zur Systemadministration durchführt wie z. B. die Rotation der Log-Dateien. Sie sollten also über Cron nicht nur Bescheid wissen, weil es sich um einen weiteren Dienst handelt, der in Ihrem Linux-System läuft, sondern weil er wirklich nützlich ist.

Man kann jedes Programm mittels Cron starten und das sogar zu den Zeiten, die einem genehm sind. Ein Programm, das unter der Kontrolle von Cron läuft, nennt sich *Cronjob*. Um einen Cronjob zu installieren, müssen Sie einen Eintrag in der so genannten Crontab-Datei vornehmen. Dies geschieht üblicherweise mit dem Kommando `crontab`. Folgender Eintrag beispielsweise startet einen Befehl täglich um 7:15 Uhr:

```
15 07 * * * /home/otto/bin/weckruf
```

Die fünf durch Leerraumzeichen⁷ (*whitespaces*) getrennten Felder spezifizieren den Termin, zu dem das darauffolgende Kommando gestartet wird:

- Minute (0-59)
- Stunde (0-23)
- Tag im Monat (1-31)
- Monat (1-12)
- Wochentag (0-7, wobei mit 0 und 7 der Sonntag gemeint ist)

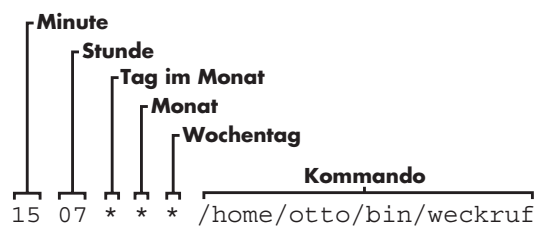


Abb. 4-3 Schema eines Eintrags in der Crontab-Datei

Das Beispiel oben startet das Kommando `weckruf` täglich, da für die Felder Tag, Monat und Wochentag Sterne spezifiziert wurden. Das »Sternzeichen« trifft so auf alle möglichen Werte zu, also jeden Tag des Monats, jeden Monat und jeden

7. Das sind Leerzeichen, Tabulatoren, verschiedene Zeilenvorschübe.

Wochentag. Wenn Otto nur am 14. jedes Monats geweckt werden soll, geschieht dies mit folgender Crontab-Zeile:

```
15 07 14 * * /home/otto/bin/weckruf
```

Man kann ebenso mehrere Werte in einem Feld eintragen, die dann voneinander mit Komma getrennt werden. Soll Otto also am 5. und 14. seinen Weckruf bekommen, kann man das mit 5, 14 im dritten Feld erreichen:

```
15 07 5,14 * * /home/otto/bin/weckruf
```

Soll er nur jeden dritten Tag im Monat geweckt werden, müsste man 5, 14 mit */3 ersetzen.

Ab dem sechsten Feld steht in der Crontab-Zeile das Kommando inklusive Parameter. Wenn cron den Job startet, kann die Befehlszeile unter Umständen eine Meldung nach stdout, stderr ausgeben oder sich nicht sauber beenden. In diesen Fällen bekommt der Eigentümer der Crontab eine E-Mail mit einer entsprechenden Ausgabe. Wenn Ihnen die E-Mails auf die Nerven gehen, empfiehlt es sich, diese Ausgabe entweder nach /dev/null umzuleiten oder in eine Datei.⁸

Die Hilfeseite mit der kompletten Formatbeschreibung ist in Abschnitt fünf zu finden. Neben dieser existiert eine für das Kommando crontab in Abschnitt eins, crontab(1). Sie müssen explizit man 5 crontab aufrufen, um die richtige zu lesen.

4.6.1 Eine Crontab installieren

Jeder Benutzer kann eine eigene Crontab-Datei besitzen. Ihr Linux-System speichert alle Crontabs unterhalb von /var/spool/cron/ – je nach Distribution entweder direkt in dem Verzeichnis oder in den Unterverzeichnissen tabs oder crontabs. Normale Benutzer haben dort weder eine Lese- noch eine Schreibberechtigung. Die Installation, das Anzeigen, Editieren oder Entfernen der Einträge wird stattdessen mit dem Kommando crontab vorgenommen.

Die einfachste Methode, eine neue Crontab zu erstellen, besteht darin, eine Datei zu erzeugen und diese dann dem crontab-Kommando mittels crontab *dateiname* zu übergeben. crontab überprüft das Dateiformat zunächst, um sicherzustellen, dass Sie nicht irgendetwas durcheinander gebracht haben. Wenn man seine eigenen Einträge angezeigt bekommen möchte, geschieht dies mittels crontab -l. Alle Einträge komplett entfernen können Sie mit crontab -r.

Mit temporären Dateien wie oben zu hantieren, um Einträge einzurichten oder zu manipulieren, kann schwierig werden. In einem Schritt lassen sich das

8. Ebenso kann über die Variable MAILTO eingestellt werden, wer die E-Mail bekommt oder ob sie überhaupt erzeugt wird, siehe Hilfeseite crontab(5).

Editieren und Installieren am besten mit `crontab -e` erledigen. Falls `crontab` dabei einen Fehler entdeckt, mahnt das Kommando dies an und erlaubt, den Fehler zu korrigieren. Über die Umgebungsvariable `VISUAL` bzw. `EDITOR` wird bestimmt, welcher Editor dabei benutzt wird.

4.6.2 Crontab-Einträge des Systems

Unter Linux stellt `Root`, auch was die `Crontab` angeht, eine Besonderheit dar: Anders als bei anderen Unix-Varianten sind bei den Distributionen von Haus aus ein paar Jobs zur Wartung und Pflege vorinstalliert. Die Verankerung dieser Tätigkeiten geschieht nicht in `/var/spool/cron`, sondern in `/etc/crontab`. Sie sollten nicht auf die Idee kommen, diese Datei mit dem `crontab`-Kommando zu editieren, da sie ein (geringfügig) anderes Format hat. Die Zeilen weisen nämlich ein Feld mehr auf. Abgesehen davon würde `crontab` diese Datei in dem Fall nach `/var/spool/cron` duplizieren. Das zusätzliche Feld enthält den Benutzer, unter dem der Job läuft. Meistens werden in den einzelnen Zeilen Skripte mit einem täglichen, wöchentlichen und monatlichen Turnus aufgerufen, die wiederum andere Skripte starten. Um die Pflegearbeiten nicht durcheinander zu bringen, empfiehlt es sich generell, von `/etc/crontab` und den Skripten die Finger zu lassen, es sei denn, Sie wissen genau, was Sie tun. Zusätzliche `Crontabs` von `Root` sollten mit den üblichen Mitteln (siehe voriges Kapitel) angelegt werden.

So könnte beispielsweise eine Zeile aussehen, die um 6.42 Uhr morgens als `Root` ein Skript zur Systempflege startet:

```
42 6 * * * root /root/bin/saubermach.sh > /dev/null 2>&1
```

Unterstützt werden auch `Crontab`-Dateien unterhalb von `/etc/cron.d` für das System. Einige Distributionen machen hiervon Gebrauch. Das Format ist das gleiche wie von `/etc/crontab`, der Name der Datei kann allerdings frei gewählt werden.

4.7 at für einmalige Aufgaben

Wenn Sie eine Aufgabe in der Zukunft starten wollen und Sie nicht `cron` benutzen wollen, da es sich um eine einmalige Angelegenheit handelt, können Sie das mit `at` erledigen. Ein Beispiel:

```
at 22:15
tagesthemem_musik.sh
```

`at` liest die Kommandos von `stdin`; beenden Sie die Eingabe mit `CTRL-D`. Das obige Beispiel führt also `tagesthemem_musik.sh` um 22.15 Uhr aus.

Um zu überprüfen, ob Jobs geplant sind, benutzt man `atq`, um Jobs zu entfernen `atrm`. Für das Starten in einer fernerer Zukunft kann hinter der Uhrzeit noch ein Datum spezifiziert werden wie `at 18:00 24.12.08`.

Ansonsten gibt es, was dieses Kommando angeht, nicht viel zu beschreiben. Im Alltag führt es ein eher stiefmütterliches Dasein. Es kann aber nützlich sein, um beispielsweise Ihr System zu einer bestimmten Zeit herunterzufahren oder Ihrer Freundin am Geburtstag ein Ständchen in Form einer MP3-Datei vorzuspielen.

4.8 Prozesse analysieren

In Abschnitt 1.16 haben Sie erfahren, wie man `ps` verwendet, um sich Prozesse zum gegenwärtigen Zeitpunkt anzeigen zu lassen. `ps` ist also prima geeignet, um eine Augenblickaufnahme eines oder mehrerer Prozesse zu erhalten. Es liefert jedoch keinerlei Informationen darüber, wie sich Prozesse mit der Zeit verändern. Durch Verwendung von `ps` alleine wird es Ihnen auch nicht gelingen, schnell herauszufinden, welcher Prozess zu viel CPU-Rechenzeit frisst oder zu viel Hauptspeicher beansprucht.

`top` kann dem abhelfen. Es zeigt Ihnen den aktuellen Systemstatus an sowie viele der Felder, die Sie auch von der Ausgabe von `ps` her kennen. Der Unterschied ist, dass die Ausgabe alle paar Sekunden – voreingestellt sind meistens drei – aktualisiert wird und Ihnen die Prozesse in einer bestimmten Reihenfolge – normalerweise nach CPU-Zeit – sortiert werden.

Sie können `top` auch noch nach dem Start mit Tastenkommandos steuern. Die wichtigsten sind:

[Leerzeichen]	Sofortige Aktualisierung
M	Sortierung nach Hauptspeicherverbrauch
T	Sortierung nach kumulativer CPU-Benutzung
P	Sortierung nach gegenwärtigem CPU-Gebrauch (Standard)
u	Auswahl der Anzeige von Prozessen bestimmter Benutzer
k	»Abschießen« von Prozessen
?	Kurzanleitung für interaktive Kommandos

Falls Sie wissen wollen, wie viel Zeit ein Kommando »auf« der CPU verbringt, stellen Sie dem Kommando einfach den Befehl `time` voran.

So würden Sie beispielsweise den CPU-Verbrauch von `ls` messen:⁹

```
time ls /usr/bin
```

9. Falls Ihre Shell über ein eingebautes Kommando `time` verfügt (dies lässt sich mit `which time` feststellen), müssten Sie stattdessen `/usr/bin/time` nehmen, um eine ähnliche Ausgabe zu bekommen.

Nach Beendigung von `ls` werden Sie eine Ausgabe erhalten wie die folgende (die Zeitangaben sind in Sekunden):

```
0.03user 0.02system 0:00.57elapsed 24%CPU (0avgtext+0avgdata 0maxresident)k
0inputs+0outputs (0major+375minor)pagefaults 0swaps
```

- *user*-Zeit ist die Zeit, die die CPU benötigt hat, den Programmcode des Programms (und nur diesen) abzuarbeiten. Steht dort eine Null, liegt das daran, dass der Prozessor das Kommando so schnell ausgeführt hat, dass die Laufzeit auf null Sekunden abgerundet wurde.
- Unter *system* wird die Systemzeit verstanden, d. h. jene Zeit, die der Kernel beispielsweise benötigt, um Dateien einzulesen oder etwas in das Konsolenfenster zu schreiben.
- Mit *elapsed* (deutsch: verstrichen) ist die gesamte Zeit gemeint, die vom Start bis zum Ende benötigt wurde. Da dies auch die Zeit beinhaltet, in der die CPU sich anderen Aufgaben widmete, ist die Angabe bei einzelnen kurzen Programmen für Performance-Messungen nicht besonders geeignet.

4.8.1 Offene Dateien finden mit `lsdf`

`lsdf` steht für *list open files* – »zeige mir geöffnete Dateien an« – und genau das tut es. Weil Unix ein sehr dateibezogenes Betriebssystem ist, stellt `lsdf` eins der wichtigsten Werkzeuge zur Fehlersuche dar. Dabei ist `lsdf` nicht auf reguläre Dateien beschränkt, sondern es kann auch für dynamische Bibliotheken, Netzverbindungen (*Sockets*), Pipes, Gerätedateien und einiges andere verwendet werden.

Startet man `lsdf` ohne Argumente, produziert es einen ganzen Schwall Zeilen. Hier ein Fragment dessen, was Ihnen dann möglicherweise entgegengeschleudert wird:

COMMAND	PID	USER	FD	TYPE	DEVICE	SIZE	NODE	NAME
init	1	root	cwd	DIR	3,1	4096	2	/
init	1	root	rtd	DIR	3,1	4096	2	/
init	1	root	txt	REG	3,1	24480	16212	/sbin/init
init	1	root	mem	REG	3,1	999542	32329	/lib/ld-2.3.3.so
init	1	root	mem	REG	3,1	1251176	32208	/lib/libc-2.3.3.so
init	1	root	10u	FIFO	3,1		97634	/dev/initctl
...								
...								
vi	3511	otto	cwd	DIR	3,4	4096	163866	/home/otto/w
vi	3511	otto	rtd	DIR	3,1	4096	2	/
vi	3511	otto	txt	REG	3,3	27048	561520	/usr/bin/vi
...								
vi	3511	otto	3rW	REG	3,4	18993	163867	/home/otto/w/c

Die einzelnen Spalten der Ausgabe haben folgende Bedeutung:

COMMAND	Name des Prozesses
PID	Prozess-ID
USER	Benutzer
FD	Dateideskriptor (englisch: <i>file descriptor</i>) Ein Dateideskriptor ist eine Zahl, die ein Prozess verwendet, um intern Dateien eindeutig zu identifizieren, was für Unix-Dateioperationen nötig ist.
TYPE	Dateityp (reguläre Datei, Verzeichnis, Pipe ...)
DEVICE	Haupt- und Nebenkennzahl der Gerätedatei, auf der die Datei zu finden ist.
SIZE	Dateigröße
NODE	Inode
NAME	Dateiname

Die Hilfeseite von `lsdf -lsdf(1)` – enthält eine vollständige Liste darüber, was in den jeweiligen Spalten für Felder auftauchen können. Normalerweise lässt sich die Bedeutung aber erraten, vorausgesetzt, Sie können ein wenig Englisch; die Ausgabe ist nicht eingedeutscht. Ein Beispiel: Schauen Sie sich die Einträge in der mit `FD` überschriebenen Spalte an. `cwd` steht für *current working directory* des Prozesses. Ein weiteres Beispiel: In der letzten Zeile sehen Sie die Datei, die gerade editiert wird.

Es gibt zwei grundsätzliche Arten, `lsdf` zu benutzen:

- Alles anzeigen lassen und die Ausgabe mittels Pipe-Symbol durch `less` oder `grep` leiten.
- Einengen der Ausgabe von `lsdf` durch die Verwendung von Kommandozeilenoptionen.

Falls Sie letztere Strategie bevorzugen: Sie können Dateinamen als Argument übergeben. `lsdf` wird dann nur die Prozesse ausgeben, die diese Datei (reguläre Datei, Verzeichnis, Gerätedatei usw.) geöffnet haben. Am Beispiel des `/tmp`-Verzeichnisses sähe das dann so aus:

```
lsdf /tmp
```

Umgekehrt kann man statt eines Dateinamens auch eine Prozess-ID übergeben und erhält alle geöffneten Dateien dieses Prozesses ausgegeben:

```
lsdf -p pid
```

`lsdf` hat jede Menge Optionen. `lsdf -h` gibt eine kurze Übersicht über die Optionen, darunter sind viele für die Formatierung der Ausgabe. `lsdf` ist auch ein nützliches Werkzeug für Netzverbindungen, vergessen Sie also nicht, Abschnitt 6.5.1 zu lesen.

Hinweis: Starten Sie `lsuf` als normaler Benutzer, wird es Ihnen – sofern es korrekt installiert ist – nur die Informationen über Prozesse und Dateien anzeigen, die Sie sehen dürfen. Das sind alle eigenen sowie einige wenige Informationen über fremde Prozesse. Dies ist ein Sicherheitsfeature. Nur Root kann mit `lsuf` herausfinden, ob Sie gerade eine Datei namens `bewerb_tolleanderefirma.txt` oder `liebesbrf.txt` geöffnet haben. (Wenn Sie den Dateinamen Ihrem Editor allerdings auf der Kommandozeile übergeben haben, kann dies jeder in der Prozessliste sehen.)

4.8.2 Abläufe mittels `trace` und `strace` verfolgen

Abgesehen von dem Befehl `time` haben alle bisher präsentierten Werkzeuge Ihnen nur etwas in die Hand gegeben, um *aktive* Prozesse unter die Lupe zu nehmen. Wenn Sie jedoch keine Ahnung haben, warum ein Programm kurz nach seinem Start stirbt, wird Ihnen auch `lsuf` nicht weiterhelfen können.

`strace` (*system call trace* – Ablaufverfolgung von Systemaufrufen) und `ltrace` (*library call trace* – Ablaufverfolgung von Bibliotheksaufrufen) können Ihnen dabei helfen herauszufinden, was ein Programm zu tun versucht. Die Werkzeuge produzieren eine recht umfangreiche Ausgabe von vielleicht für Anfänger hieroglyphisch anmutendem Text. Wenn Sie jedoch eine Idee haben, wonach Sie suchen müssen – genau das will ich Ihnen hier vermitteln –, werden Sie Problemen damit ganz gut auf die Schliche kommen können.

Ein *Systemaufruf* ist eine Operation, die der Kernel für einen Prozess durchführt, z. B. um Dateien zu öffnen und Daten daraus zu lesen. `strace` gibt all die Systemaufrufe aus, die ein Prozess ausführt. Um eine grobe Ahnung davon zu bekommen, geben Sie einfach mal folgendes Kommando ein:

```
strace cat /dev/null
```

Am Anfang sieht man lediglich ein paar Angaben zu dynamischen Bibliotheken; das soll hier aber ignoriert werden:

```
execve("/bin/cat", ["cat", "/dev/null"], [/* 37 vars */]) = 0
uname({sys="Linux", node="dama n", ...}) = 0
brk(0) = 0x804d000
open("/etc/ld.so.preload", O_RDONLY) = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY) = 3
fstat64(3, {st_mode=S_IFREG|0644, st_size=70872, ...}) = 0
old_mmap(NULL, 70872, PROT_READ, MAP_PRIVATE, 3, 0) = 0x40014000
close(3) = 0
open("/lib/libc.so.6", O_RDONLY) = 3
read(3, "\177ELF\1\1\0\0\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0\360Y\1"... , 512) = 512
```

Sie können die Ausgabe bis nach `mmap` überspringen, bis Sie etwa Folgendes zu Gesicht bekommen:

```
fstat64(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 2), ...}) = 0
open("/dev/null", O_RDONLY|O_LARGEFILE) = 3
fstat64(3, {st_mode=S_IFCHR|0666, st_rdev=makedev(1, 3), ...}) = 0
read(3, "", 4096) = 0
close(3) = 0
close(1) = 0
exit_group(0)
```

An dieser Stelle sehen Sie nun die interessanten Dinge: Schauen Sie zunächst auf den `open()`-Aufruf, der die Datei `/dev/null` öffnet. 3 als Resultat bedeutet, dass der Aufruf ein Erfolg war, die positive Zahl als Rückgabewert entspricht dem Dateideskriptor. Weiter am Ende sehen Sie, wo `cat` aus `/dev/null` liest. Beachten Sie, dass dieser `read`-Aufruf sich auf den Dateideskriptor bezieht. Da es nichts weiter zu lesen gibt, schließt das Programm die Datei mit dem Dateideskriptor 3 mittels `close` und beendet sich.

Was passiert nun, wenn es wirklich ein Problem gibt? Versuchen Sie einmal `strace cat keine_datei` einzugeben und schauen Sie sich den Aufruf `open()` an:

```
open("keine_datei", O_RDONLY|O_LARGEFILE) = -1 ENOENT (No such file or directory)
```

An dieser Stelle sehen Sie, dass `strace` Ihnen nicht nur den Fehler anzeigt, sondern auch noch eine kleine Beschreibung des Fehlers angibt (allerdings auf Englisch).

Fehlende Dateien sind die am häufigsten auftretenden Probleme unter Unix. Wenn also die Informationen im Systemlog (`syslog`) oder anderen Log-Dateien nicht besonders hilfreich sind und Sie nicht mehr weiter wissen, kann `strace` gute Dienste leisten. Man kann `strace` ebenso für bereits laufende Programme wie z. B. Dämonen anwenden. Ein Beispiel:

```
strace -o gammeld_strace.log -p 1234
```

1234 sei hier die Prozess-ID des Dämons `gammeld`. In diesem Beispiel wird der Parameter `-o` benutzt, um die Ausgabe in eine Datei umzulenken. Wenn Sie dem Dämon beim Starten auf die Finger schauen wollen, empfiehlt sich:

```
strace -o gammeld_strace -ff gammeld
```

Bei der Verwendung von `-ff` im Zusammenhang mit `-o` landet die Ausgabe in den Dateien `gammeld_strace.pid`, wobei `pid` die Prozess-ID des oder der jeweiligen (Kind-)Prozesse(s) darstellt.

`ltrace` funktioniert ähnlich, nur dass es die Aufrufe in den dynamischen Bibliotheken verfolgt. Auch die Ausgabe ist vergleichbar mit der von `strace`, mit dem Unterschied, dass normalerweise eine Menge mehr Aufrufe in die Bibliotheken erfolgen als Systemaufrufe, die Sie mit `strace` sehen. Mehr zu dynamischen

Bibliotheken steht in Abschnitt 8.1.4. Statische Bibliotheken sind Bestandteil von Programmen und werden daher von trace nicht unterstützt.

4.9 Prozessprioritäten abstimmen

Es ist möglich, die Zeitzuteilung des Kernels für Prozesse zu beeinflussen, also einem Prozess mehr oder weniger Rechenzeit (CPU-Zeit) als anderen Prozessen zu geben. Der Kernel lässt jeden Prozess mit einer *Priorität*, d. h. einer Zahl zwischen -20 und 19 ablaufen, wobei -20 vielleicht etwas irritierend die höchste Priorität ist.

ps -l gibt zwar die gegenwärtigen Prozessprioritäten aus, aber es ist einfacher, sich das Ganze mit top anzuschauen:

```

Tasks: 79 total,  1 running, 77 sleeping,  0 stopped,  1 zombie
Cpu(s):  9.4% us,  1.0% sy,  0.4% ni, 88.4% id,  0.6% wa,  0.0% hi,  0.1% si
Mem:   320364k total,  301420k used,  18944k free,  17320k buffers
Swap:  514072k total,  13260k used,  500812k free,  68260k cached

```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
23066	di rkw	16	0	1868	896	1732	R	5.6	0.3	0:00.06	top
10382	di rkw	15	0	93288	74m	28m	S	1.9	23.7	200:04.75	MozillaFirebird
1	root	16	0	1348	480	1316	S	0.0	0.1	0:04.28	init
2	root	34	19	0	0	0	S	0.0	0.0	0:00.00	ksoftirqd/0
3	root	5	-10	0	0	0	S	0.0	0.0	0:06.36	events/0
4	root	5	-10	0	0	0	S	0.0	0.0	0:08.89	kblockd/0
5	root	15	0	0	0	0	S	0.0	0.0	0:00.02	khudb
6	root	25	0	0	0	0	S	0.0	0.0	0:00.00	pdflush
7	root	15	0	0	0	0	S	0.0	0.0	1:29.48	pdflush
8	root	15	0	0	0	0	S	0.0	0.0	1:01.23	kswapd0
9	root	10	-10	0	0	0	S	0.0	0.0	0:00.00	ai o/0
10	root	19	0	0	0	0	S	0.0	0.0	0:00.00	scsi_ah_0
11	root	15	0	0	0	0	S	0.0	0.0	0:00.00	kseriod
127	root	16	0	1412	572	1360	S	0.0	0.2	0:03.24	syslogd
129	root	16	0	1344	464	1312	S	0.0	0.1	0:00.02	klogd
134	daemon	19	0	1436	404	1388	S	0.0	0.1	0:00.00	portmap
136	root	16	0	1380	516	1352	S	0.0	0.2	0:00.03	inetd

In der Ausgabe von top werden die Prozessprioritäten des Kernels (besser gesagt: des Schedulers) in der Spalte PR angezeigt. Je höher die Zahl, desto weniger wahrscheinlich wird der Kernel diesem Prozess CPU-Zeit zuteilen, wenn andere Prozesse noch Zeit benötigen. Diese Prioritäten ändern sich häufig während der Programmausführung, entsprechend des Betrags an CPU-Zeit, die der Kernel beansprucht.

Die Priorität ist nicht das einzige Kriterium, nach dem der Kernel einem Prozess CPU-Zeit zuteilt. Neben der Spalte PR befindet sich die Spalte für *nice*, die

mit `NI` überschrieben ist. Der *Nice-Wert* stellt einen Hinweis für den Kernel-Scheduler dar, und ist das, worauf Sie Ihr Augenmerk lenken sollten, wenn Sie versuchen, die Entscheidung des Kernels für die Zeitzuteilung zu beeinflussen. Der Kernel addiert den Nice-Wert zu der gegenwärtigen Priorität, um das nächste Zeitfenster für den Prozess zu bestimmen. Der Standardwert ist null. Wenn Sie möchten, dass ein Prozess sich weiter hinten anstellt, können Sie den Nice-Wert mit dem `renice`-Befehl auf 19 ändern, `pid` entspricht dabei der Prozess-ID:

```
renice 19 pid
```

Dies ist beispielsweise dann nützlich, wenn Sie rechenintensive Operationen im Hintergrund ausführen und Sie nicht möchten, dass diese Ihre interaktive Sitzung zum Stocken bringen.

Falls Sie `renice` als Root starten, dürfen Sie als Nice-Wert auch eine negative Zahl übergeben. Etwas Vorsicht ist dabei jedoch angebracht, da Systemprozesse unter Umständen zu kurz kommen können, wenn Ihre Hardware etwas älter ist. Bei den meisten Linux-Systemen, an denen nur ein Benutzer arbeitet und keinerlei Berechnungen durchführt, brauchen Sie die Nice-Werte nicht zu ändern. `renice` war viel wichtiger in der Zeit, als sich viele interaktiv arbeitende Benutzer mit samt rechenintensiven Hintergrundprogrammen auf einer Maschine tummelten.

4.10 Das Leistungsverhalten des Systems kontrollieren

Von der Performance her verhalten sich die meisten Linux-Systeme mit den Standardeinstellungen der jeweiligen Distribution unproblematisch. Man kann Stunden und Tage damit verbringen, am System »zu schrauben«, ohne irgendwelche merkbaren Erfolge zu erzielen. Manchmal ist es jedoch möglich, die Leistung des Systems zu verbessern. Dieser Abschnitt konzentriert sich primär auf Speicher- und Prozessorengpässe und zeigt, ob sich eine Hardwareaufrüstung lohnt.

Die zwei wichtigsten Dinge, die Sie wissen müssen, wenn es um die Leistung Ihres Systems geht, sind der Durchschnittswert für die Systemauslastung (englisch: *load average*) und die Anzahl der so genannten Seitenfehler (englisch: *page faults*).

Zunächst soll die Systemauslastung untersucht werden. `uptime` präsentiert Ihnen gleich drei Werte dafür (hier die vollständig deutsche Version der Ausgabe bei Red-Hat-Linux):

```
... an 91 Tagen ... Durchschnittslast: 0,09, 0,03, 0,01
```

Die drei Zahlen geben die Auslastung bzw. Last während der letzten Minute, letzten 5 Minuten und 15 Minuten an. Wie Sie sehen, ist das System nicht fürchterlich ausgelastet, da die CPU innerhalb der letzten Viertelstunde nur zu einem Prozent etwas getan hat (die letzte Zahl ist 0,01). Viele der heutigen Systeme werden

Ihnen eine Durchschnittslast von etwa null präsentieren, es sei denn, Sie übersetzen gerade ein Programm, jagen digitale Monster oder haben einen dummen Bildschirmschoner laufen. Für das System ist eine Systemauslastung von null ein gutes Zeichen, da es bedeutet, dass Ihr Prozessor (der Ihres Computers) nicht wirklich gefordert ist.

Falls die Last um 1 pendelt, stellt dies nicht notwendigerweise ein schlechtes Zeichen dar: Das bedeutet lediglich, dass die CPU – in diesem Abschnitt wird angenommen, Ihr Computer hat nur eine – die ganze Zeit etwas zu tun hat. Welcher Prozess sich viel Rechenzeit genehmigt, können Sie mittels `top` herausfinden. Wenn die Systemauslastung 2 oder mehr ist, haben Sie es wahrscheinlich mit mehreren Prozessen zu tun, die sich gegenseitig um Rechenzeit bemühen. Der Kernel versucht in so einem Fall, die Systemressourcen gleichmäßig zu verteilen. Bei zwei Prozessoren würde dieser Zustand vergleichbar mit einer Auslastung von 1 bei einem Prozessor sein.

Eine höhere Durchschnittslast bedeutet nicht notwendigerweise, dass Ihr System ein Problem hat. Ein Computer mit genügend Hauptspeicher und Reserven bei Ein- und Ausgabeoperationen (übers Netzwerk oder über die angeschlossenen Platten) kann gut mit vielen um die CPU konkurrierenden Prozessen fertig werden. Es dauert dann nur etwas länger, als wenn es nur einen Prozess gibt, der rechenintensiv ist und sich um die eine CPU bemüht. Es ist eher wichtig, wie schnell das System reagiert bzw. Anfragen jeglicher Art gerecht wird.

Wenn Sie jedoch merken, dass das System langsam und die Durchschnittslast hoch ist, sollten Sie der Sache auf den Grund gehen. Eine Ursache, die Sie als Erstes überprüfen sollten, ist, ob der Kernel evtl. unter Speichernot leidet. Dies macht sich dadurch bemerkbar, dass er häufig Speicherseiten von Applikationen in Swap – die Auslagerungsdatei bzw. -partition – hineinschiebt oder wieder hereinholt. Überprüfen Sie daher mittels `free` oder `cat /proc/meminfo`, wie viel Ihres kostbaren Hauptspeichers noch frei ist. Die entsprechenden Angaben finden Sie in der Spalte »free« bzw. »MemFree«. Wenn diese Zahl häufig niedrig ist und Sie unter »cached/buffers« auch nicht mehr viel Speicher finden, ist es gut möglich, dass ein zusätzlicher Speicherriegel die hohe Systemauslastung mindert. Dazu gleich mehr.

Bei Speicherknappheit treten häufig so genannte *page faults* (zu Deutsch etwa: Speicherseitenanforderungsfehler) auf. Damit hat es Folgendes auf sich: Der Kernel versucht, eine Speicherseite beispielsweise Ihrer Applikation zu adressieren, die sich nicht im Hauptspeicher befindet, sondern die zuvor wegen Speicherknappheit ausgelagert wurde (siehe Paging, Abschnitt 2.5). In dem Fall fordert er diese Speicherseite aus dem Swap an und Ihre Applikation muss so lange warten. Wenn Speichernot herrscht, lagert der Kernel eine andere Speicherseite dafür aus. Sie sehen an diesem vereinfachten Beispiel, dass kostbare CPU-Zeit verloren geht, da der Kernel – bevor Ihre Applikation zum Zuge kommt – erst einmal Vorarbeit leisten muss. Dies tut er auf Kosten einer anderen Anwendung, die schlimmstenfalls in einem

Sekundenbruchteil auch einen Seitenfehler bekommt. Nicht zu vergessen: Abgesehen von der reinen Verzögerung wegen der nicht im Hauptspeicher vorhandenen Speicherseite ist auch der Transfer dieser Speicherseite von der Festplatte langsam. Sie ahnen es bestimmt schon: Wenn dieser Vorgang häufig auftritt, ist der Kernel mehr mit sich selbst und der Festplatte beschäftigt als mit den Anwendungen.

Nach dieser Exkursion wird es Sie sicherlich interessieren, wie Sie dies feststellen können. `vmstat` (*virtual memory statistics*) ist hier das Mittel der Wahl. Es zeigt Ihnen nicht nur an, wie viele Speicherseiten in vorgegebener Zeit ein- und ausgelagert werden, sondern gibt noch Auskunft über die Auslastung der CPU, die Aktivität von Blockgeräten und eine Menge anderer Dinge. Es ist eins der wichtigsten Analysewerkzeuge unter Unix, Sie müssen nur die Ausgabe des Programms verstehen lernen. Hier das Resultat von `vmstat 2`, das Statistiken alle zwei Minuten ausgibt:

```
procs -----memory----- ---swap-- ----io---- --system-- ----cpu----
r b swpd free buff cache si so bi bo in cs us sy id wa
5 0 108152 5360 167732 554908 2 3 28 46 278 143 5 3 92 1
0 0 108152 5360 167732 554908 0 0 0 0 1004 1761 3 3 94 0
0 0 108152 5376 167732 554908 0 0 0 0 1003 1786 3 2 95 0
0 0 108152 5376 167732 554908 0 0 0 0 1003 1794 3 2 95 0
0 0 108152 5392 167732 554908 0 0 0 0 1003 1774 4 2 94 0
1 0 108152 5392 167732 554908 0 0 0 0 1002 1736 4 1 96 0
```

Die Ausgabe ist aufgeteilt in mehrere Kategorien. Die erste Zeile ist die Tabellenüberschrift: Prozesse (`procs`), Speicherbenutzung (`memory`), Ein- und Auslagern von Speicherseiten (`swap`), Disk (`io`), Kernel-Scheduler (`system`) und schließlich CPU-Auslastung (`cpu`), aufgeschlüsselt nach verschiedenen Teilen des Systems. Die zweite Zeile schlüsselt die Kategorien weiter auf, mehr dazu unten.

Die erste Zeile der Statistik können Sie getrost ignorieren: Sie gibt die Durchschnittswerte seit dem Systemstart wieder. Das Beispiel entspricht einem System, das nicht wirklich viel zu tun hat. In dem vorliegenden Fall sind 108.152 KByte in Swap ausgelagert (siehe Spalte `swpd`), und es sind anfangs 5360 KByte Speicher verfügbar (`free`). Dies mag den Anschein von Speicherknappheit erwecken. Sie sehen aber, dass in der Spalte `si` und `so` (Swap in/out) keine Aktivitäten zu verzeichnen sind, der Kernel also gegenwärtig weder Seiten ein- noch auslagert.¹⁰

Ganz rechts sehen Sie die Aufteilung der CPU-Zeit in vier Spalten ab `us`. Diese geben den Anteil an, den die CPU mit Anwendungen (`us`) oder Kernaufgaben (`sy`) beschäftigt ist, einfach nur Däumchen dreht (`id`) oder auf die Abarbeitung von Paketen aus dem Netz oder von Festplatte wartete (`wa`). Die letzte Spalte wird

10. Auf die Gefahr hin, Sie als Linux-Einsteiger mit zu viel Theorie zu belästigen: Anhand der Spalten `buff` und `cache` sehen Sie, dass der Kernel noch reichlich Speicher für seine Zwecke (Dateisystemoperationen) »abgezwickelt« hat. Würde Speichermangel vorherrschen, stünden hier weniger als 167.732 bzw. 554.908 KByte.

Ihnen nur bei neueren Systemen begegnen.¹¹ In diesem Beispiel gibt es auf dem Rechner nicht viel für die CPU zu tun: Gerade mal 4% der CPU sind für Anwendungen abseits des Kernels draufgegangen, bis zu 3% der CPU-Zeit hat der Kernel gebraucht, sonst hat der Rechner nur gefaulenzt (94-96%).

Nun schauen Sie, was passiert, wenn der Rechner durch ein fettes speicherfressendes Programm etwas zu tun bekommt (die zweite und dritte Zeile sind entstanden, bevor das Programm richtig loslegte):

procs		-----memory-----				---swap--		-----io----		--system--		----cpu----			
r	b	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	id	wa
0	0	122056	4136	32740	621572	1	3	28	44	340	347	20	5	74	1
1	0	122056	4120	32740	621572	0	0	0	0	1134	6511	3	3	94	0
0	0	122056	4140	32740	621572	0	0	0	0	1035	6611	4	3	93	0
0	0	122056	4140	32740	621572	0	0	0	0	1035	6611	26	14	61	0
0	5	145396	1232	25244	553916	262	11966	460	12044	1125	5759	30	44	22	4
4	1	156988	3020	10196	487192	374	5544	618	5618	1159	5702	21	45	0	34
3	2	163536	3288	3188	471788	444	3412	966	3580	1165	3722	14	59	0	28
0	4	181424	1224	3004	437740	154	8834	174	8942	1171	3349	14	59	0	26
3	6	214828	3476	2020	403836	178	8166	324	8166	1159	2046	19	40	0	41
2	6	257456	4376	1284	374016	110	9560	146	9560	1142	3506	11	37	0	53
0	26	279580	1856	1332	364956	116	9908	170	9912	1116	3059	10	21	0	69

Hier kommt die CPU ein wenig ins Schwitzen. Sie sehen verschiedene Gründe: Im Anwenderbereich geht die CPU-Last auf bis zu 30% hoch, was für sich allein genommen nicht viel ist. Der Kernel fängt jedoch an, ab der fünften Zeile kräftig Speicherseiten auszulagern (siehe Spalte *so*), wodurch die CPU auch bis zu 59% (*sy*) beschäftigt ist. Die Anzahl der freien Speicherseiten (*free*) sowie der für Dateioperationen reservierte Speicher geht zur Neige (bis auf 1332 KByte), es werden weitere Seiten ausgelagert (*so*), am Ende des Beispiels sind anstelle von 122 MByte knapp 280 MByte ausgelagert (*swpd*). In den letzten Zeilen bleiben gerade 10-20% der CPU für Anwendungen übrig (*us*), die CPU dreht zu 0% Däumchen. Sie sehen aber auch, dass der Kernel mit fortschreitender Dauer nicht viel Eigenarbeit leistet (*sy*), sondern eher mit dem Bewegen von Speicherseiten auf Festplatte (*bo*) beschäftigt ist. Er muss sogar noch lange auf die langsame Festplatte warten (*wa*), rechts unten sind 69% seiner Zeit dafür angegeben.

Dieses extreme Beispiel hat Ihnen hoffentlich ein Gefühl dafür vermittelt, wie man Speichermangel analysieren kann. Achten Sie besonders auf die Spalten *so* und *si*. Wenn sich dort häufig etwas in der Größenordnung von tausend oder mehr Speicherseiten pro Sekunde tut, sobald Sie größere Programme starten, und Ihr System sich träge verhält, dann kann eine Speichererweiterung Wunder tun.

Dieses Buch kann leider derlei Speicherproblematiken nur anreißen. Um Informationen über die restlichen Spalten von *vmstat* zu bekommen, lesen Sie am

11. Und bei älteren Kernels (2.4) steht an dieser Stelle eine Null.

besten die Hilfeseite. Wenn Sie sich näher für das Thema interessieren, sei Ihnen zunächst ein klassisches Buch über Betriebssysteme allgemein empfohlen wie *Operating System Concepts* [Silberschatz et al. 2002]. Falls Sie dann noch nicht genug haben oder Sie der Schuh wegen eines speziellen Problems drückt, schauen Sie in *System Performance Tuning* [Musumeci, Loukides 2002] nach.

4.11 Kommandos mit Root-Rechten ausführen

Bevor Sie weitere Systemkommandos kennen lernen, sollten Sie eine andere sichere Methode erlernen, Kommandos als Root abzusetzen. Sie wissen wahrscheinlich bereits, dass Sie `su` starten und das Root-Passwort eingeben können, um eine Shell für Root zu bekommen. Das funktioniert prima, hat jedoch einige Nachteile:

- Sie haben keine zuverlässige Aufzeichnung über systemverändernde Kommandos.
- Sie haben keine Angaben darüber, welcher Benutzer es war, der systemverändernde Kommandos eingegeben hat.
- Sie arbeiten nicht mit Ihrer gewohnten Shell-Umgebung [1].

Viele Installationen, besonders bei so genannten Workstations, verwenden ein Paket namens `sudo`, das Administratoren erlaubt, Kommandos als Superuser bzw. Root abzusetzen, obwohl sie als normaler Benutzer angemeldet sind. Ein Beispiel: Sie möchten `vi pw` benutzen, um `/etc/passwd` zu editieren. In diesem Fall würde das Kommando so aussehen:

```
sudo vi pw
```

Wenn Sie dies Kommando absetzen, protokolliert `sudo` dies durch den Syslog-Dienst in der Kategorie `authpriv`, so dass Sie eine Aufzeichnung von dem Aufruf des Kommando haben.

Natürlich lässt Sie `sudo` nicht jedes beliebige Kommando als Superuser bzw. Root ohne Passwort ausführen. Sie müssen die dafür vorgesehenen Benutzer und die erlaubten Kommandos in `/etc/sudoers` spezifizieren. `sudo` hat eine Reihe durchdachter Optionen, mit denen Sie z. B. selbst die zu übergebenden Parameter der Kommandos vorschreiben oder Befehle, Benutzer oder Computer zu Gruppen zusammenfassen können. Bei vielen Optionen werden Sie wahrscheinlich gar nicht in die Verlegenheit kommen, diese zu benutzen. Daraus resultiert auch der Haken an der Sache: `/etc/sudoers` hat eine nicht ganz einfache Syntax. Hier nun ein einfaches Beispiel, bei dem es dem Benutzer `data` erlaubt ist, ohne Passworteingabe Prozesse abzuschließen:

```
data ALL = NOPASSWD: /bin/kill
```

Hier steht als Erstes der Benutzername, gefolgt vom Hostnamen. ALL ist ein Platzhalter für alle Hostnamen, ebenso könnte dort ein einzelner Rechnername angegeben sein. Rechts vom Gleichheitszeichen steht NOPASSWD mit Doppelpunkt, gefolgt vom erlaubten Befehl mit vollständigem Pfad. Wird NOPASSWD: weggelassen, muss das (eigene!) Passwort beim Aufruf von sudo eingegeben werden. Aus Sicherheitsgründen sollten Sie sehr sorgsam abwägen, wer ohne Passwort welches Kommando ausführen darf, und dies aufs Allernötigste beschränken. NOPASSWD: ALL ist absolut tabu, da es dem Benutzer erlauben würde, alle Programme mit Root-Rechten aufzurufen, ohne ein Passwort eingeben zu müssen.

Ein etwas weiter gehendes Beispiel:

```
User_Alias COMMANDER = worf, data
Cmd_Alias WEGDAMT = /bin/kill, /bin/killall

riker    enterprise = NOPASSWD: WEGDAMT
COMMANDER ALL      = /usr/bin/cancel, /sbin/shutdown
```

Diese `/etc/sudoers` definiert zunächst eine Benutzergruppe namens COMMANDER und eine Kommandogruppe namens WEGDAMT, beides muss großgeschrieben werden. riker darf auf dem Rechner enterprise die Kommandos kill und killall aus /bin ohne Passwortheingabe ausführen. Die Benutzer worf und data dürfen auf allen Computern Druckjobs zurückziehen und die Computer herunterfahren. Sie müssen aber dafür ihr Passwort eingeben.

Die Syntax kann etwas heikel sein wie bei `/etc/passwd`. Daher nimmt man üblicherweise `vi sudo`, um `/etc/sudoers` zu editieren.

Das ist alles, was Sie an dieser Stelle über sudo wissen müssen. Wenn Sie mehr Wissensdurst verspüren, konsultieren Sie am besten die Hilfeseiten `sudoers(5)` und `sudo(8)`.

Echte und effektive Benutzer-ID

Wenn Sie Programme mit *suid*-Bit aufrufen, müssen Sie sich darüber im Klaren sein, dass es mehrere Benutzer-IDs innerhalb des laufenden Programms gibt. Die ID, die die Zugriffsrechte zu dem Programm bestimmt, ist die echte bzw. reale Benutzer-ID. Beim Start des `setuid`-Programms setzt der Kernel jedoch die *effektive Benutzer-ID (euid)* gleich dem Eigentümer des Programms, was Ihre Zugriffsrechte von da an bestimmt; es behält jedoch Ihre eigentliche Benutzer-ID als *reale Benutzer-ID*.

sudo ist fürchterlich gründlich: Es wechselt die reale wie effektive Benutzer-ID für Sie. In seltenen Fällen kann es vorkommen, dass Programme Probleme damit haben, unter der realen Benutzer-ID von root zu laufen. In dem Fall lässt sich das Problem dadurch lösen, dass man in `/etc/sudoers` eine Zeile einfügt wie:

```
Default ts:                                stay_setuid
```
