

# Mac OS X für Profis

Michael Kofler

Terminal ■ sudo ■ 32/64-Bit-Kernel  
Mac-OS-X-spezifische Kommandos  
HFS ■ X11 ■ Gimp ■ Emacs ■ LaTeX  
MacPorts ■ Fink ■ VirtualBox ■ Tastatur

ebooks.kofler

## **Mac OS X für Profis**

Autor: Michael Kofler

Verlag: ebooks.kofler, Schönbrunnngasse 54c, 8010 Graz, Österreich

<http://ebooks.kofler.info>

[ebooks@kofler.info](mailto:ebooks@kofler.info)

**ISBN:** 978-3-902643-00-1

Korrektorat: Friederike Daenecke, Zülpich

© 2010 ebooks.kofler / Michael Kofler

Viele in diesem ebook genannten Hard- und Software-Bezeichnungen sind geschützte Markennamen.

Dieses ebook wurde mit großer Sorgfalt verfasst. Dennoch sind Fehler nicht ganz auszuschließen. Für allfällige Fehler kann keine Verantwortung oder Haftung übernommen werden. Verbesserungsvorschläge oder Korrekturen sind selbstverständlich willkommen ([ebooks@kofler.info](mailto:ebooks@kofler.info)). Vielen Dank dafür!

Dieses ebook ist durch das österreichische Urheberrecht geschützt. Sie dürfen das ebook für den persönlichen Gebrauch kopieren und ausdrucken, aber nicht an andere Personen weitergeben, weder in elektronischer noch in anderer Form.

## Inhaltsverzeichnis

<b>1</b>	<b>Das Terminal</b>	<b>5</b>
1.1	Tastatur und Maus	7
1.2	Farben	11
<b>2</b>	<b>Mac-OS-X-Grundlagen</b>	<b>14</b>
2.1	sudo (Administratorrechte im Terminal)	14
2.2	Apple- und BSD-spezifische Kommandos	16
2.3	32- versus 64-Bit-Kernel	25
2.4	Das Apple-Dateisystem HFS	27
2.5	Organisation des Verzeichnisbaums	33
2.6	X11	37
2.7	Xcode	41
<b>3</b>	<b>Open-Source-Software</b>	<b>45</b>
3.1	Firefox und Thunderbird	45
3.2	OpenOffice	47
3.3	Gimp	47
3.4	Inkscape	50
3.5	Scribus	51
3.6	Emacs	51
3.7	LaTeX (MacTeX)	57
<b>4</b>	<b>MacPorts und Fink</b>	<b>62</b>
4.1	MacPorts installieren	63
4.2	Das port-Kommando	65
4.3	MacPorts-Internia	69
4.4	MacPorts-Erfahrungen	70
4.5	Fink	73
<b>5</b>	<b>VirtualBox</b>	<b>78</b>
5.1	VirtualBox installieren	78
5.2	Windows installieren	79
5.3	Linux installieren	87
5.4	VirtualBox-Internia und Konfigurationstipps	89
<b>A</b>	<b>Die Tastatur</b>	<b>95</b>
<b>B</b>	<b>Tools und Utilities</b>	<b>106</b>

## Vorwort

Dieses Buch bzw. ebook richtet sich an fortgeschrittene Mac-OS-X-Anwender. Es beginnt nicht bei Adam und Eva, sondern dort, wo es interessant wird: beim Terminal! Schlüsselthemen in diesem ebooks sind:

- die effiziente Nutzung des Terminals
- Mac-OS-X-Grundlagen (Dateisystem HFS, Apple- bzw. BSD-spezifische Kommandos wie `defaults`)
- die Installation wichtiger Open-Source-Programme (Firefox, Thunderbird, OpenOffice, Gimp, Inkscape, Scribus, Emacs und LaTeX)
- die Verwendung von MacPorts und Fink
- die Ausführung virtueller Maschinen mit VirtualBox (Linux, Windows)
- der Umgang mit der Tastatur (PC-Tastatur verwenden, Tastatur im Terminal, in Emacs, in VirtualBox etc.)
- Tools und Utilities, die die Arbeit erleichtern bzw. effizienter machen

Die Benutzeroberfläche von Mac OS X ist in diesem ebook dagegen nur ein Randthema: Nach ein paar Stunden Probieren und Herumspielen haben Sie alle wesentlichen Funktionen entdeckt. Wozu also lange darüber schreiben?

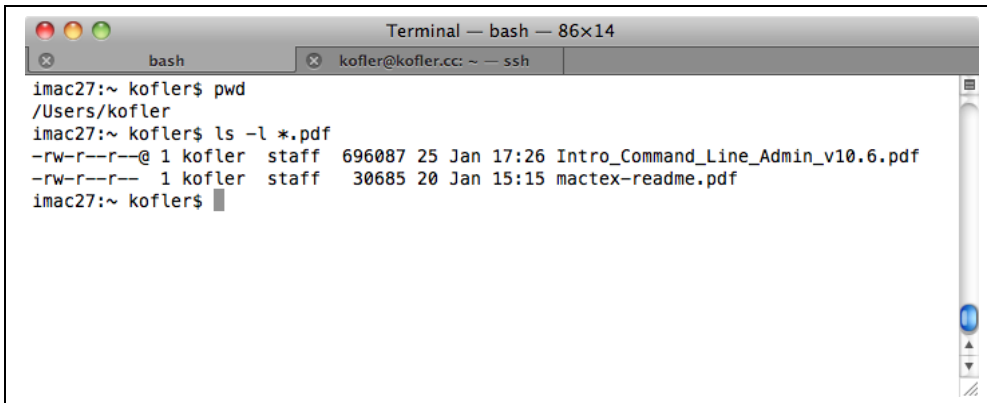
Dieses ebook ist aktuell zu Mac OS X 10.6 (*Snow Leopard*). Viele der beschriebenen Funktionen und Programme stehen aber auch in älteren Versionen des Apple-Betriebssystems zur Verfügung.

Lernen Sie Ihren Mac *richtig* kennen, werfen Sie einen Blick hinter die schöne Benutzeroberfläche, und nutzen Sie das Potenzial kostenloser Open-Source-Programme! Viel Spaß dabei wünscht Ihnen

Michael Kofler (Februar 2010)  
<http://kofler.info/>

# 1 Das Terminal

Jedes Unix-ähnliche Betriebssystem lässt sich über ein Terminal bedienen – und das gilt auch für Mac OS X. Am einfachsten starten Sie das Terminal-Programm über das Dock (Verzeichnis Programme/Dienstprogramme, siehe Abbildung 1.1). Im Terminal läuft standardmäßig die auch unter Unix/Linux populäre Shell bash. Verwunderlich ist nur, dass mit Mac OS X 10.6 noch die bash-Version 3.2 ausgeliefert wird, während schon lange Version 4.0 und mittlerweile sogar schon Version 4.1 verfügbar ist.

The image shows a screenshot of a Mac OS X Terminal window. The title bar reads "Terminal — bash — 86x14". The window has a standard Mac OS X title bar with red, yellow, and green buttons. The terminal content shows the following commands and output:

```
imac27:~ kofler$ pwd
/Users/kofler
imac27:~ kofler$ ls -l *.pdf
-rw-r--r--@ 1 kofler  staff  696087 25 Jan 17:26 Intro_Command_Line_Admin_v10.6.pdf
-rw-r--r--  1 kofler  staff   30685 20 Jan 15:15 mactex-readme.pdf
imac27:~ kofler$
```

Abbildung 1.1: Das Terminal

## Hinweis

Das mit Mac OS X mitgelieferte Terminal erfüllt alle Anforderungen, die ich an ein Terminal stelle. Wer noch intensiver mit dem Terminal arbeitet und nach noch mehr Konfigurationsmöglichkeiten und Zusatzfunktionen sucht, der sollte einen Blick auf die Open-Source-Alternative iTerm werfen:

<http://iterm.sourceforge.net/>

## Kommandos

Im Terminal stehen standardmäßig fast alle gängigen Unix/Linux-Kommandos sowie diverse Apple- bzw. BSD-spezifische Kommandos zur Verfügung (siehe Tabelle 1.1).

Umgang mit Dateien und Verzeichnissen	cd, cp, df, du, ln, ls, mkdir, rm, rmdir
Dateien komprimieren und archivieren	bzip, bunzip, ditto, gzip, gunzip, tar, unzip, zip
Benutzerverwaltung, Zugriffsrechte	chgrp, chmod, chown, passwd
Umgang mit Textdateien	cat, cut, grep, less, sort, tail
Suchen	find, grep, locate
Verwaltung des Dateisystems	asr, df, diskutil, hdiutil, mount
Prozessverwaltung	bg, fg, kill, killall, launchctl, nice, top
Systemverwaltung	nvrn, otool, shutdown, systemsetup
Hilfe	help, info, man
Kommandos mit Administratorrechten ausführen (siehe Abschnitt 2.1)	su, sudo
Netzwerk-Tools	ifconfig, ping, rsync, scp, ssh
Editoren	emacs (siehe Abschnitt 3.6), nano, vim
Apple- bzw. BSD-spezifische Kommandos (siehe Abschnitt 2.2)	asr, defaults, diskutil, ditto, hdiutil, launchctl, nvrn, open, otool, systemsetup
MacPorts, Fink	port, fink (siehe Kapitel 4)

Tabelle 1.1: Wichtige Kommandos

## Hinweis

Syntaxdetails zu nahezu jedem Kommando können Sie mit `man kommandoname` nachlesen. Beachten Sie, dass bei manchen Kommandos einzelne Optionen anders implementiert sind als unter Linux! Eine gute Einführung in das Arbeiten im Terminal gibt die folgende PDF-Datei von Apple:

[http://images.apple.com/server/macosx/docs/Intro\\_Command\\_Line\\_Admin\\_v10.6.pdf](http://images.apple.com/server/macosx/docs/Intro_Command_Line_Admin_v10.6.pdf)

## X-Programme

Sofern X11 installiert ist (siehe Abschnitt 2.6), können Sie im Terminal auch X-Programme starten. Um das auszuprobieren, führen Sie das Kommando `xeyes &` aus. (Das gleichnamige Programm zeigt in einem kleinen Fenster zwei Augen an, die immer in die Richtung des Mausursors sehen.)

Mit `ssh` können Sie sogar X-Programme auf einem anderen Rechner ausführen. Dazu loggen Sie sich mit `ssh -X` auf dem externen Unix- oder Linux-Rechner ein. Wenn Sie nun ein Programm mit grafischer Benutzeroberfläche starten, erscheint das Programm am Mac-OS-X-Desktop.

X11 stellt mit dem Programm `xterm` ein eigenes Terminal zur Verfügung. Es bietet gegenüber dem gewöhnlichen Terminal-Programm keine nennenswerten Vorteile, dafür aber viele Nachteile und wesentlich weniger Konfigurationsmöglichkeiten.

### 1.1 Tastatur und Maus

Im Terminal irritiert zuerst einmal die Tastatur – insbesondere dann, wenn man bereits unter Linux oder einem anderen Unix-System mit Terminals gearbeitet hat und daher entsprechende Erwartungshaltungen hat. Die zwei größten Ärgernisse bestehen darin, dass `alt`-Tastenkürzel nicht funktionieren und dass es einiger Fingerakrobatik bedarf, um den Cursor um eine Seite nach oben bzw. nach unten zu bewegen. Tabelle 1.2 fasst die wichtigsten Tastenkürzel zusammen, die standardmäßig funktionieren.

Das `alt`-Problem ist auf Terminalebene kaum lösbar: Unter Mac OS X dient `alt` zur Eingabe diverser Sonderzeichen – selbstverständlich auch im Terminal. Unter Unix/Linux in Kombination mit einer PC-Tastatur gilt `Alt` dagegen als Metataste, mit der diverse Steuerungskommandos eingegeben werden können. Beispielsweise bewegen `Alt+B` bzw. `Alt+F` den Cursor um ein Wort vor bzw. zurück, und `Alt+←` löscht ein Wort zurück.

Sie können im Einstellungsdialog die Option WAHLTASTE ALS META-TASTE VERWENDEN aktivieren (siehe Abbildung 1.2) – dann funktionieren diese Tastenkürzel auch im Mac-OS-X-Terminal. Allerdings können Sie dann Sonderzeichen wie @[]{}~€ nicht mehr eingeben. Deswegen ist es zweckmäßiger, häufig benötigte `alt`-Tastenkürzel im Terminal neu zu definieren. Diese Tastenkürzel funktionieren dann auch in diversen Programmen und Editoren, die direkt im Terminal ausgeführt werden.

<code>⇧</code> + <code>fn</code> + <code>↑</code> / + <code>↓</code>	den Cursor eine Seite nach oben bzw. unten bewegen (funktioniert z. B. für <code>man</code> , <code>less</code> und in diversen Editoren)
<code>ctrl</code> + <code>A</code> / + <code>E</code>	Cursor an den Beginn bzw. das Ende der Zeile stellen
<code>ctrl</code> + <code>C</code>	das laufende Kommando bzw. Programm abbrechen
<code>ctrl</code> + <code>K</code>	bis zum Ende der Zeile löschen
<code>ctrl</code> + <code>L</code>	Terminalinhalt löschen ( <i>clear screen</i> )
<code>ctrl</code> + <code>R</code>	nach zuletzt eingegebenen Kommandos suchen
<code>ctrl</code> + <code>Y</code>	zuletzt gelöschten Text wieder einfügen
<code>ctrl</code> + <code>Z</code>	das laufende Kommando bzw. Programm unterbrechen (es kann später mit den Kommandos <code>fg</code> oder <code>bg</code> als Vorder- bzw. Hintergrundprozess fortgesetzt werden)
<code>&lt;</code> / <code>&gt;</code>	an den Anfang bzw. das Ende des Dokuments springen (funktioniert während der Ausführung von <code>man</code> und <code>less</code> )
<code>↩</code>	Expansion unvollständiger Kommando- und Dateinamen
<code>cmd</code> + <code>T</code>	öffnet ein weiteres Terminal-Dialogblatt ( <i>tab</i> )
<code>cmd</code> + <code>⌘</code> / + <code>⌘</code>	wechselt in das vorige bzw. nächste Dialogblatt

Tabelle 1.2: Wichtige Tastenkombinationen zur Bedienung des Terminals

## Eigene Tastenkürzel definieren

Im Dialogblatt `TASTATUR` der Terminaleinstellungen können Sie eigene Tastenkürzel definieren. Das setzt freilich voraus, dass Sie wissen, welche Funktionen der *readline*-Bibliothek mit welchen ANSI-Escape-Codes verbunden sind (siehe Tabelle 1.3).

## Hinweis

Innerhalb des Terminals läuft die Shell *bash*, die wiederum auf die *readline*-Bibliothek zurückgreift, um Tastatureingaben zu verarbeiten. Im Detail können Sie das hier nachlesen:

[http://www.gnu.org/software/bash/manual/html\\_node/Readline-Init-File.html](http://www.gnu.org/software/bash/manual/html_node/Readline-Init-File.html)

Für das ESC-Zeichen gibt es unterschiedliche Notationen, z. B. `\033`, `\e` oder `\M.\033` ist die oktale Schreibweise. 33 ist der oktale Code für 27, also für das ESC-Zeichen.

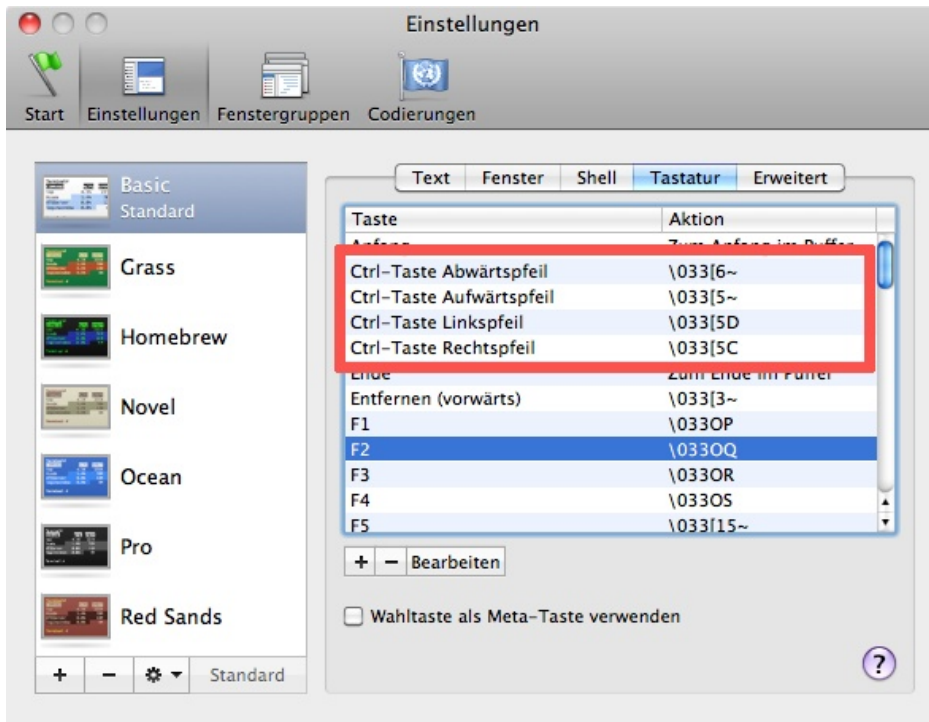


Abbildung 1.2: Tastaturkonfiguration für das Terminal

\033[6~	Cursor um eine Seite nach oben bewegen
\033[5~	Cursor um eine Seite nach unten bewegen
\033[5D	Cursor an den Beginn der Zeile bewegen
\033[5C	Cursor an das Ende der Zeile bewegen
\033b	Cursor um ein Wort zurück bewegen
\033f	Cursos um ein Wort vor bewegen
\033\177	Wort rückwärts löschen

Tabelle 1.3: Einige ANSI-Escape-Codes

Auf meinem Rechner habe ich die in Tabelle 1.3 zusammengefassten Escape-Sequenzen mit leicht erreichbaren `ctrl`- und `alt`-Tastenkürzeln verbunden (siehe Abbildung 1.2).

## bash-completions

Wenn Sie im Terminal `⌘` drücken, ergänzt die Shell Bash nach Möglichkeit den schon begonnenen Kommando- oder Dateinamen. Unter Linux funktioniert dieser Mechanismus noch wesentlich besser. Beispielsweise liefert man `l⌘` eine Liste aller man-Seiten, die mit dem Buchstaben L beginnen.

Um diese Funktion auch unter Mac OS X zu nutzen, installieren Sie mit MacPorts das Paket `bash-completions` (also `sudo port install bash-completions`, siehe Kapitel 4). Anschließend ergänzen Sie `.bash_profile` um die folgenden drei Zeilen und starten das Terminal-Programm neu.

```
if [ -f /opt/local/etc/bash_completion ]; then
    . /opt/local/etc/bash_completion
fi
```

## Maus

Die Maus spielt im Terminal nur eine eingeschränkte Rolle. Bei manchen Editoren besteht immerhin die Möglichkeit, die Cursorposition mit `alt`+Mausklick zu verändern. Wenn Sie das folgende Kommando ausführen, erhält in Zukunft jenes Terminalfenster den Eingabefokus, über das zuletzt der Mauscursor bewegt wurde. Beim Arbeiten mit vielen Terminalfenstern erspart das so manchen Mausclick.

```
$ defaults write com.apple.Terminal FocusFollowsMouse true
```

## Tipp

Unter Unix/Linux ist es üblich, dass Sie zuvor mit der Maus markierten Text ohne weitere Umstände mit der mittleren Maustaste einfügen können. Im Terminal (nicht aber in anderen Apple-Programmen) funktioniert das Einfügen ebenfalls, sofern Sie eine Maus mit drei Tasten bzw. mit einem Mousrad einsetzen *und* beim Klicken der mittleren Taste bzw. des Mousrads zugleich `cmd` drücken. Wer eine Apple-Maus oder ein Notebook mit Trackpad verwendet, muss mangels mittlerer Maustaste auf diese praktische Tastenkombination leider verzichten.

## 1.2 Farben

Im Einstellungsdialog können Sie zwischen verschiedenen Farbschemata für das Terminal auswählen. Die so eingestellten Farben gelten für den Hintergrund, die Standardtextfarbe, die Cursorfarbe etc.

Darüber hinaus kann das Terminal selbstverständlich auch Textausgaben in verschiedenen Farben darstellen. Unbegreiflicherweise sind Standardkommandos wie `ls` oder `grep` so vorkonfiguriert, dass sie von dieser Möglichkeit keinen Gebrauch machen. Abhilfe schafft beim Kommando `ls` die Option `-G`. Damit werden Verzeichnisse, ausführbare Dateien etc. farbig markiert. Bei `grep` müssen Sie die Option `--color` angeben, damit die Suchergebnisse farbig hervorgehoben werden.

Auf die Dauer ist die Eingabe dieser Optionen natürlich mühsam. Abhilfe: Erstellen Sie mit einem beliebigen Texteditor die Datei `.bash_profile` in Ihrem Heimatverzeichnis, und geben Sie darin die folgenden Kommandos an:

```
alias ls='ls -G'  
alias grep='grep --color'  
if [ -e .profile ]; then  
    source .profile  
fi
```

Die Datei wird erst bei einem Neustart des Terminals berücksichtigt. Mit den `alias`-Kommandos definieren Sie Abkürzungen für `ls` und `grep`, die die Farboptionen inkludieren. Die `if`-Anweisung verarbeitet außerdem die Datei `.profile`, falls diese existiert.

Es geht aber noch bunter :-). Die Umgebungsvariable `PS1` definiert, welche Zeichenkette vor der Kommandoeingabe angezeigt wird. Standardmäßig besteht diese Zeichenkette aus drei Teilen: dem Rechnernamen (im folgenden Beispiel `imac27`), dem aktuellen Verzeichnis (wobei nur der letzte Teil des Pfads angegeben wird) und dem Login-Namen. Und natürlich werden alle drei Teile in derselben Farbe angezeigt:

```
imac27:Documents kofler$
```

Dieser Standardformatierung liegt die Einstellung `\h:\W \u\ $` zugrunde. Mein Lieblingsprompt hat aber die Form `user@host:directory$`. Das entspricht der `PS1`-Einstellung `\u@\h:\W\ $`. Außerdem will ich, dass der Prompt in blauer Farbe angezeigt wird. Um das zu erreichen, fügen Sie die folgende Zeile in `.bash_profile` hinzu. (Probieren Sie das Kommando vorher im Terminal aus!)

```
export PS1='\[\e[0;34m\]\u@\h:\W\$\[\e[0;39m\] '
```

Das ergibt dann den folgenden Prompt in blauer Farbe:

```
kofler@imac27:Documents$
```

## Hinweis

`\[\e[0;34m\]` schaltet die blaue Farbe ein, `\[\e[0;39m\]` aktiviert wieder die Standardtextfarbe. Eine Menge weiterer Hintergrundinformationen und Tipps zur individuellen Gestaltung Ihres Prompts finden Sie auf der folgenden Website. Dort sind auch die Codes für andere Farben aufgelistet.

<http://sos.blog-city.com/>

[mac\\_os\\_x\\_\\_bash\\_customize\\_your\\_terminal\\_prompt\\_a\\_little\\_color.htm](http://mac_os_x__bash_customize_your_terminal_prompt_a_little_color.htm)

## Transparenz

Es ist total unpraktisch, aber es sieht cool aus: ein transparentes Terminal, durch dessen Hintergrund Sie die dahinter laufenden Programme bzw. den Desktop sehen. Wenn Sie es ausprobieren möchten, öffnen Sie den Einstellungsdialog des Terminals, wechseln in das Dialogblatt `EINSTELLUNGEN|FENSTER`, klicken dort den Button `FARBE` an und stellen den Regler `DECKKRAFT` nach Bedarf ein.

# 2 Mac-OS-X-Grundlagen

Hinter der auf Hochglanz polierten Benutzeroberfläche von Mac OS X verbirgt sich ein Unix-ähnliches Betriebssystem auf der Basis von NeXTStep und BSD. NeXTStep war ein Betriebssystem der Firma NeXT, die später von Apple gekauft wurde. Die *Berkeley Software Distribution* (BSD) ist eine freie Unix-Variante. Als Apple 2000 mit den Arbeiten an Mac OS X begann, kombinierte es Elemente aus NextStep und BSD. Der eigentliche Kernel von Mac OS X steht deswegen auch im Quellcode zur Verfügung (ganz im Gegensatz zu den anderen Komponenten von Mac OS X).

Dieser Abschnitt beschreibt einige Grundlagen von Mac OS X: die Kommandos `sudo` und `defaults`, die Struktur des Verzeichnisbaums, das Apple-Dateisystem HFS etc. Dabei setze ich Unix/Linux-Vorkenntnisse voraus und konzentriere mich auf die Details, in denen sich Mac OS X von Unix/Linux unterscheidet.

## 2.1 `sudo` (Administratorrechte im Terminal)

Als gewöhnlicher Benutzer können Sie im Terminal keine Systemdateien verändern, keine Systemprozesse starten oder stoppen und generell kaum administrative Aufgaben erledigen. Dazu müssen Sie als `root` arbeiten, also mit Administratorrechten. Dazu sieht Mac OS X – übrigens genauso wie Ubuntu – das Kommando `sudo` vor. Es kann auf zwei Arten verwendet werden:

- `sudo` kommando führt das angegebene Kommando mit `root`-Rechten aus.
- `sudo -s` wechselt in den `root`-Modus. Alle weiteren Kommandos werden nun so ausgeführt, als würden sie von `root` gestartet. `ctrl` + `D` beendet diesen Modus.

In beiden Fällen müssen Sie vorher einmal Ihr eigenes Passwort angeben. `sudo` merkt sich die Authentifizierung für fünf Minuten. In dieser Zeitspanne können Sie also weiterer `sudo`-Kommandos ausführen, ohne das Passwort neuerlich angeben zu müssen.

### **`/private/etc/sudoers`**

Die Datei `/private/etc/sudoers` steuert, welche Benutzer `sudo` benutzen dürfen. Standardmäßig sind das alle Benutzer, die der Gruppe `admin` angehören. Mit dem Kommando `visudo` können Sie die Datei `sudoers` ändern (siehe auch `man sudo`). Verändern Sie `sudoers` nicht mit einem anderen Editor! Wenn Ihnen dabei ein Syntaxfehler passiert, funktioniert `sudo` womöglich nicht mehr, und das kann fatale Folgen haben!

### **root-Login**

Grundsätzlich ist es unter Mac OS X aus Sicherheitsgründen unmöglich, sich als `root` einzuloggen. Wenn Sie den `root`-Login unter Mac OS X 10.6 Snow Leopard aktivieren möchten, starten Sie im Finder das Programm `System/Library/CoreServices/Verzeichnisdienste`. (Das Programm kann standardmäßig nicht über das Dock gestartet werden. Falls Sie die englische Lokalisierung von Mac OS X verwenden, lautet der Dateiname `Directory Utility`.) Im Programm *Verzeichnisdienste* klicken Sie auf das Schloss-Symbol und geben Ihr Passwort an. Anschließend führen Sie das Menükommando `BEARBEITEN|ROOT-BENUTZER AKTIVIEREN AUS`.

Die Vorgehensweise zur Aktivierung des `root`-Logins variiert stark je nach Mac-OS-X-Version. Auf der folgenden Website finden Sie eine Anleitung für die Mac-OS-X-Versionen 10.5 und 10.4:

<http://support.apple.com/kb/HT1528>

## 2.2 Apple- und BSD-spezifische Kommandos

Dieser Abschnitt stellt einige wichtige Apple- oder BSD-spezifische Kommandos kurz vor. Eine vollständige Beschreibung aller derartigen Kommandos ist hier aber aus Platzgründen nicht möglich. Weitere Details zu den hier vorgestellten Kommandos liefert man `kommandoname`. Und eine gute Aufzählung (aber keine Beschreibung) vieler Apple-spezifischer Kommandos enthält der Anhang der folgenden PDF-Datei:

[http://images.apple.com/server/macosx/docs/Intro\\_Command\\_Line\\_Admin\\_v10.6.pdf](http://images.apple.com/server/macosx/docs/Intro_Command_Line_Admin_v10.6.pdf)

### defaults

Windows speichert unzählige Einstellungen in der Registrierdatenbank. Der Gnome-Desktop (Linux) verwendet hierfür das `gconf`-System. Bei Apple übernehmen *property lists* (\*.plist-Dateien) diese Aufgabe, wobei jedes Programm bzw. jeder Dienst seine eigene Konfigurationsdatei hat. Die Dateien werden in den folgenden Verzeichnissen gespeichert:

<code>/Library/Preferences/*.plist</code>	(globale Einstellungen)
<code>/Network/Library/Preferences/*.plist</code>	(Einstellungen von Netzwerk-(Server-)Dienstern)
<code>/Users/name/Library/Preferences/*.plist</code>	(benutzerspezifische Einstellungen)

Um Namenskonflikte zu vermeiden, ist für die Dateinamen eine umgekehrte DNS-Notation üblich, wobei sich die Namesbestandteile aus der Firma bzw. Organisation plus dem eigentlichen Programmnamen ergeben: Beispielsweise hat die Konfigurationsdatei für die *Time Machine* den Namen `com.apple.TimeMachine.plist`. Die Konfigurationsdatei für das Open-Source-Programm *VirtualBox* lautet `org.virtualbox.app.VirtualBox.plist`.

\*.plist-Dateien werden wahlweise im XML-Format oder in binärer Form gespeichert. Beide Formen sind inhaltlich gleichwertig, die binäre Variante ist aber naturgemäß effizienter und platzsparender. Mit dem Kommando `plutil -convert` können Sie eine Umwandlung von bzw. in die XML-Form durchführen. Wenn Sie eine binäre \*.plist-Datei nur ansehen (aber nicht bleibend auf der Festplatte ändern) möchten, führen Sie das Kommando so aus:

```
$ cd /Library/Preferences/  
$ plutil -convert xml1 -o - com.apple.TimeMachine.plist | less
```

Egal, ob die Datei nun binär oder im XML-Format vorliegt – manuelle Änderungen an \*.plist-Dateien sind fehleranfällig und nicht zu empfehlen. Soweit Sie nicht eine grafische Benutzeroberfläche verwenden (z. B. den Einstellungsdialog des jeweiligen Programms oder die Mac-OS-X-Systemeinstellungen), können Sie einzelne Einstellungen mit dem Kommando `defaults` auslesen und verändern.

`defaults read` liefert eine schier endlose Liste aller Einstellungen aller Programme des aktiven Benutzers. `defaults read name` zeigt die Einstellungen eines bestimmten Programms an:

```
$ defaults read com.apple.x11  
{  
    "wm_click_through" = 1;  
    "wm_ffm" = 0;  
}
```

`defaults write` verändert eine vorhandene Einstellung bzw. legt einen neuen Eintrag an. Sie müssen drei Parameter angeben: den Programmnamen in der umgekehrten DNS-Notation, den Namen des Schlüssels und den gewünschten Wert. Die folgende Einstellung bewirkt, dass das Terminal, über dessen Fenster zuletzt der Mauscursor bewegt wurde, automatisch den Eingabefokus erhält. Beim Arbeiten mit mehreren Terminals kann das praktisch sein.

```
$ defaults write com.apple.Terminal FocusFollowsMouse true
```

## Hinweis

Das `defaults`-Handbuch (`man defaults`) warnt davor, die Einstellungen laufender Programme zu verändern: Zum einen nehmen die Programme die Änderungen erst nach einem Neustart wahr. Zum anderen besteht die Gefahr, dass die Programme die Änderungen unwissentlich wieder überschreiben.

**Tipp**

Mit `defaults` können viele Optionen verändert werden, für die Apple keine Konfigurationsdialoge vorgesehen hat. Aus der Suche nach unbekanntem bzw. »geheimen« Einstellungen hat sich ein eigener Sport entwickelt. Eine umfassende Sammlung von `defaults`-Kommandos für jeden erdenklichen Zweck finden Sie auf der folgenden Website:

<http://secrets.blacktree.com/>

**diskutil**

`diskutil` hilft bei der Low-Level-Administration von Festplatten und deren Partitionen. Mit `diskutil` können Sie auch ein RAID-System verwalten. Die Anwendung dieses Kommandos ist naturgemäß gefährlich – wenn Sie einen Fehler machen, können Sie den gesamten Inhalt Ihrer Festplatte(n) verlieren! Vollkommen gefahrlos ist hingegen das folgende Kommando, das die Partitionen der angeschlossenen Festplatten auflistet:

**\$ diskutil list**

```
/dev/disk0
#:                TYPE NAME                SIZE          IDENTIFIER
0:    GUID_partition_scheme      *2.0 TB       disk0
1:                EFI                209.7 MB      disk0s1
2:                Apple_HFS Macintosh HD    2.0 TB        disk0s2

/dev/disk1
#:                TYPE NAME                SIZE          IDENTIFIER
0:    FDisk_partition_scheme     *250.1 GB     disk1
1:                Apple_HFS disk1s1        250.1 GB     disk1s1
```

Alle Größenangaben von `diskutil` erfolgen in aktuellen Mac-OS-X-Versionen im Dezimalsystem. Ein Terabyte (TB) ist also  $10^{12}$  Byte, nicht  $2^{40}$  Byte.

**Hinweise**

Zu `diskutil` gibt es eine grafische Benutzeroberfläche (`PROGRAMME|DIENSTPROGRAMME|FESTPLATTEN-DIENSTPROGRAMM`).

`diskutil` eignet sich nur für Apple-Rechner mit Intel-Prozessoren. Diese Rechner verwenden GUIDs (weltweit eindeutige ID-Nummern) zur Kennzeichnung der Partitionen (siehe [http://de.wikipedia.org/wiki/GUID\\_Partition\\_Table](http://de.wikipedia.org/wiki/GUID_Partition_Table)).

Ältere Apple-Rechner auf der Basis von PPC-Prozessoren verwenden ein anderes Partitionsformat. Zur Administration müssen Sie das Kommando `pdisk` verwenden.

## ditto

Das Kommando `ditto` kopiert einen Verzeichnisbaum in ein anderes Verzeichnis. Wenn das Zielverzeichnis noch nicht existiert, wird es erzeugt. Beim Kopieren bleiben alle besonderen HFS-Eigenschaften der Dateien erhalten (Forks, EAs, ACLs etc., siehe auch Abschnitt 2.4). Optional kann `ditto` den Verzeichnisbaum auch in ein CPIO- oder ZIP-Archiv schreiben. Auch eine Übertragung des Verzeichnisses auf einen anderen Rechner via `ssh` ist möglich.

```
$ ditto verz1 verz2           (kopiert Verzeichnis 1 in Verzeichnis 2)
$ ditto -c -j verz1 bak.cpio.bz2 (speichert verz1 in einem komprimierten CPIO-Archiv)
$ ditto -x bak.cpio.bz2 verz3   (packt das CPIO-Archiv in Verzeichnis 3 aus)
$ ditto -c verz1 - | ssh hostname ditto -x - verz2   (kopiert verz1 via ssh)
```

## hdiutil

`hdiutil` ermöglicht es, Disk Images (\*.dmg-Dateien, siehe Abschnitt 2.4) zu erzeugen, in das Dateisystem einzubinden, in ein anderes Format umzuwandeln, auf eine CD/DVD zu brennen etc.

```
$ hdiutil create -srcfolder verz1 image.dmg (erzeugt das Image mit den Dateien
                                             aus verz1)
$ hdiutil verify image.dmg                 (kontrolliert die Prüfsumme)
$ hdiutil burn image.dmg                   (brennt das Image auf eine CD/DVD)
$ hdiutil attach image.dmg                 (bindet das Image in das Dateisystem ein)
$ hdiutil detach /dev/disk<n>              (löst das Image aus dem Dateisystem)
```

## launchd

Unter Unix und bei vielen Linux-Distributionen ist das sogenannte Init-V-System dafür verantwortlich, während des Rechnerstarts alle Funktionen und Netzwerkdienste zu initialisieren. Bei Mac OS X ist dafür das Programm `launchd` verantwortlich, das beim Einschalten des Rechners als Erstes gestartet wird. (Mit `ps ax` können Sie sich davon überzeugen, dass `launchd` die Prozessnummer 1 hat.) Teilweise übernimmt `launchd` auch Aufgaben, für die auf traditionellen Unix-Systemen `xinitd` und `cron` zuständig sind.

Das Ziel bei der Entwicklung von `launchd` war es, ein einheitliches System zu entwerfen, das sich um die Ausführung aller Programme bzw. Netzwerkdienste kümmert, die automatisch gestartet werden sollen. Eine Besonderheit von `launchd` besteht darin, dass es nicht nur mit `root`-Rechten für den Start bzw. Stopp von Systemdiensten verantwortlich ist; es kann vielmehr auch von einem gewöhnlichen Benutzer gestartet werden, um eigene Prozesse zu steuern. `launchd` und die zugehörigen Werkzeuge sind übrigens als Open-Source-Software lizenziert, werden aber bisher außerhalb von Mac OS X nicht oder höchstens ganz vereinzelt eingesetzt. Eine gute Einführung zu `launchd` finden Sie in der englischen Wikipedia sowie unter:

<http://developer.apple.com/MacOsX/launchd.html>

<http://www.afp548.com/article.php?story=20050620071558293>

Die Konfiguration von `launchd` erfolgt durch `*.plist`-Dateien, die sich an verschiedenen Orten im Dateisystem befinden:

<code>/System/Library/LaunchAgent/*.plist</code>	(Betriebssystemfunktionen)
<code>/System/Library/LaunchDaemons/*.plist</code>	
<code>/Library/LaunchAgent/*.plist</code>	(optionale Netzwerkdienste etc.)
<code>/Library/LaunchDaemons/*.plist</code>	
<code>~/Library/LaunchAgent/*.plist</code>	(benutzerspezifische Prozesse)

Die `*.plist`-Dateien in den `LaunchDaemon`-Verzeichnissen sind für Prozesse verantwortlich, die beim Rechnerstart automatisch ausgeführt werden sollen. Die

LaunchAgent-Verzeichnisse enthalten dagegen Konfigurationsdateien für Prozesse, die nur nach einem Login gestartet werden sollen.

Um Namenskonflikte zu vermeiden, ist für die Namen der Konfigurationsdateien eine umgekehrte DNS-Notation üblich: Beispielsweise hat die launchd-Konfigurationsdatei für den NFS-Dämon von Mac OS X den Dateinamen `com.apple.nfsd.plist`. Die launchd-Konfigurationsdatei für den Update-Dienst von Adobe befindet sich dagegen in der Datei `com.adobe.ARM.*.plist`.

## launchctl

Der Start der durch launchd verwalteten Prozesse erfolgt automatisch bzw. bei Bedarf – das ist ja gerade der Sinn von launchd. Wenn Sie neue Konfigurationsdateien einrichten, vorhandene Konfigurationsdateien deaktivieren oder den Status von Prozessen ermitteln möchten, nehmen Sie das Kommando `launchctl` zu Hilfe.

`launchctl list` listet alle launchd bekannten Prozesse auf. Wenn in der ersten Spalte des Ergebnisses eine Prozessnummer angegeben wird, läuft dieser Prozess. Bei nicht laufenden Prozessen gibt die zweite Spalte an, welchen Wert der Prozess bei seinem letzten Ende zurückgegeben hat. Das hilft mitunter bei der Fehlersuche.

`launchctl list name` liefert detaillierte Informationen zu einem bestimmten Prozess. Mit der zusätzlichen Option `-x` liefert `launchctl` diese Informationen im XML-Format der `*.plist`-Konfigurationsdatei.

`launchctl stop name` bzw. `launchctl start name` beenden bzw. starten einen Prozess manuell. Das sollte eigentlich nie erforderlich sein, weil sich launchd ja automatisch um die Prozessverwaltung kümmert. Bisweilen sind diese `launchctl`-Kommandos aber ein gutes Hilfsmittel, um einen hängen gebliebenen Prozess gewissermaßen »geordnet« neu zu starten. Die beiden folgenden Kommandos starten den für die TimeMachine verantwortlichen Backup-Dämon neu:

```
$ sudo launchctl stop com.apple.backupd
```

```
$ sudo launchctl start com.apple.backupd
```

`launchctl unload name` deaktiviert die angegebene Konfigurationsdatei und beendet den dazugehörigen Prozess. Die umgekehrte Wirkung hat `launchctl load name`: Die betreffende Konfigurationsdatei wird aktiviert und der darin beschriebene Prozess gestartet.

## Hinweise

In der Ergebnisliste von `launchctl list` tauchen auch Prozesse auf, die *nicht* von `launchd` gestartet wurden, die aber zu irgendeinem Zeitpunkt mit `launchd` kommuniziert haben (Details dazu finden Sie mit `man launchd`).

Es macht einen Unterschied, ob Sie `launchctl` als gewöhnlicher Benutzer oder mit `sudo` aufrufen! Im ersten Fall bezieht sich `launchctl` auf Benutzerprozesse, im zweiten Fall auf Systemprozesse.

`launchctl`-Kommandos können auch in `/etc/launchd.conf` und in `~/ .launchd.conf` angegeben werden. Diese Dateien werden beim Start von `launchd` ausgewertet.

## nvrnm

Einige Systemeinstellungen Ihres Apple-Rechners werden im *Non-Volatile Random-Access Memory* (kurz NVRAM) gespeichert. Dort enthaltene Informationen bleiben auch dann erhalten, wenn Ihr Computer vollkommen ausgeschaltet ist. Mit dem Kommando `nvrnm` können Sie das NVRAM auslesen bzw. darin enthaltene Einstellungen ändern. Beispielsweise aktiviert das folgende Kommando den 64-Bit-Kernel (siehe Abschnitt 2.3):

```
$ sudo nvrnm boot-args='arch=x86_64'
```

## open

`open` öffnet vom Terminal aus ein Programm mit grafischer Benutzeroberfläche (Option `-a`) bzw. eine Datei. Beim Start von Programmen können auch Parameter

übergeben werden, beispielsweise, um in einem Editor eine bestimmte Datei zu öffnen.

```
$ open -a /Applications/Aquamacs\ Emacs.app (startet den Editor Aquamacs Emacs)
$ open name.pdf (zeigt die PDF-Datei an)
```

## otool

otool liefert Informationen über Objektdateien und Bibliotheken. Das folgende Kommando verrät, welche Bibliotheken der Webbrowser Firefox nutzt. Die Ausgabe wurde aus Platzgründen gekürzt.

```
$ otool -L /Applications/Firefox.app/Contents/MacOS/firefox-bin
/Applications/Firefox.app/Contents/MacOS/firefox-bin:
    /System/Library/Frameworks/Cocoa.framework/Versions/A/Cocoa (...)
    @executable_path/XUL (compatibility version 1.0.0, current version 1.0.0)
    @executable_path/libmozjs.dylib (compatibility version 1.0.0, ...)
    ...
    /System/Library/Frameworks/Carbon.framework/Versions/A/Carbon (...)
    /System/Library/Frameworks/CoreAudio.framework/Versions/A/CoreAudio (...)
    /usr/lib/libstdc++.6.dylib (compatibility version 7.0.0, ...)
    /usr/lib/libgcc_s.1.dylib (compatibility version 1.0.0, ...)
    /usr/lib/libSystem.B.dylib (compatibility version 1.0.0, ...)
```

## periodic

Unter Unix bzw. Linux kümmert sich der cron-Dämon darum, regelmäßig erforderliche Wartungs- und Aufräumarbeiten durchzuführen. Unter Mac OS X übernimmt das Kommando periodic diese Aufgabe. Dieses Kommando wird von launchd automatisch gestartet. Dafür sind die folgenden Konfigurationsdateien verantwortlich:

```
/System/Library/LaunchDaemons/com.apple.periodic-daily.plist
/System/Library/LaunchDaemons/com.apple.periodic-weekly.plist
/System/Library/LaunchDaemons/com.apple.periodic-monthly.plist
```

Beispielsweise bestimmt `com.apple.periodic-weekly.plist`, dass das Kommando `periodic weekly` jeden Samstag um 3:15 ausgeführt wird. Je nachdem, welcher Parameter an `periodic` übergeben wird (`daily`, `weekly` oder `monthly`), führt `periodic` alle Shell-Scripts in den folgenden Verzeichnissen aus:

```
/private/etc/periodic/daily/  
/private/etc/periodic/weekly/  
/private/etc/periodic/monthly/
```

Wenn Sie selbst Wartungsaufgaben haben, die einmal pro Tag, Woche oder Monat erledigt werden sollen, erstellen Sie einfach ein entsprechendes Script in den angegebenen Verzeichnissen. Dabei können Sie sich am Muster `999.local` orientieren. (Die dem Script-Namen vorangestellte dreistellige Nummer bestimmt die Reihenfolge, in der die Scripts ausgeführt werden.)

Wenn sich der Rechner zum geplanten Zeitpunkt für die Ausführung von `periodic` gerade im Ruhezustand befindet, verschiebt sich die Ausführung der Kommandos um die Zeitspanne des Ruhezustands. Wenn der Rechner vor der geplanten bzw. verschobenen Ausführungszeit aus- und nach der geplanten Ausführungszeit wieder eingeschaltet wird, entfällt der Aufruf von `periodic` ersatzlos. Mit anderen Worten: Der Ruhezustand verzögert die Ausführung der Wartungs-Scripts, und das Ausschalten kann mit etwas Pech dazu führen, dass manche Wartungs-Scripts nie oder erst sehr viel später ausgeführt werden.

Um zu kontrollieren, wann die jeweiligen Scripts das letzte Mal ausgeführt wurden, sehen Sie sich das Änderungsdatum der Dateien `/var/log/*.out` an:

```
$ ls -l /var/log/*.out  
-rw-r--r--  1 root  wheel  30102  3 Feb 07:50 /var/log/daily.out  
-rw-r--r--  1 root  wheel   166  1 Feb 07:58 /var/log/monthly.out
```

Das obige Ergebnis bedeutet, dass die wöchentlichen Wartungs-Scripts bisher noch nie ausgeführt wurden – die Datei `weekly.out` existiert noch gar nicht. Zu diesem Ergebnis kam es zwei Monate, nachdem der Rechner in Betrieb ging! Aus den Textdateien `*.out` geht hervor, welche Arbeiten bei der letzten Ausführung der Scripts erledigt wurden.

Die Wartungs-Scripts kümmern sich unter anderem darum, dass Logging-Dateien nicht ins Unermessliche wachsen, dass temporäre Dateien gelöscht werden und dass diverse Datenbanken (z. B. für `whatis` und `locate`) regelmäßig aktualisiert werden.

Um die Wartungs-Scripts sofort manuell auszuführen, führen Sie das folgende Kommando aus:

```
$ sudo periodic daily weekly monthly
```

### systemsetup

Mit dem Kommando `systemsetup` können Sie einige Systemparameter auslesen bzw. verändern. Dazu zählen die aktuelle Uhrzeit und das Datum, die Zeitzone, die Zeitspanne, nach der der Rechner, die Festplatte bzw. der Bildschirm den Energiesparmodus aktiviert, der Hostname etc. Mit `systemsetup` können Sie auch den SSH-Server des Rechners aktivieren bzw. deaktivieren und zwischen dem 32- und 64-Bit-Kernel umschalten.

## 2.3 32- versus 64-Bit-Kernel

Apple preist Mac OS X als 64-Bit-Betriebssystem an. Und tatsächlich laufen unter Mac OS X 10.6 (Snow Leopard) viele Programme als 64-Bit-Programme, wovon Sie sich im Programm `AKTIVITÄTSANZEIGE` überzeugen können. Dennoch ist das nur die halbe Wahrheit: Der Kernel, also die Basis des Betriebssystems, läuft standardmäßig als 32-Bit-Programm! `uname -a` schafft Klarheit:

```
$ uname -a
```

```
Darwin imac27.lan 10.2.0 Darwin Kernel Version 10.2.0:
```

```
Tue Nov 3 23:08:29 PST 2009; root:xnu-1486.2.11~3/RELEASE_I386 i386
```

Auch die meisten »großen« Anwendungsprogramme, z. B. Firefox, iPhoto, iTunes, OpenOffice, die iWork-09-Komponenten, Thunderbird und VirtualBox laufen als 32-Bit-Programm. Zu den wenigen positiven Ausnahmen zählen Finder und Safari.