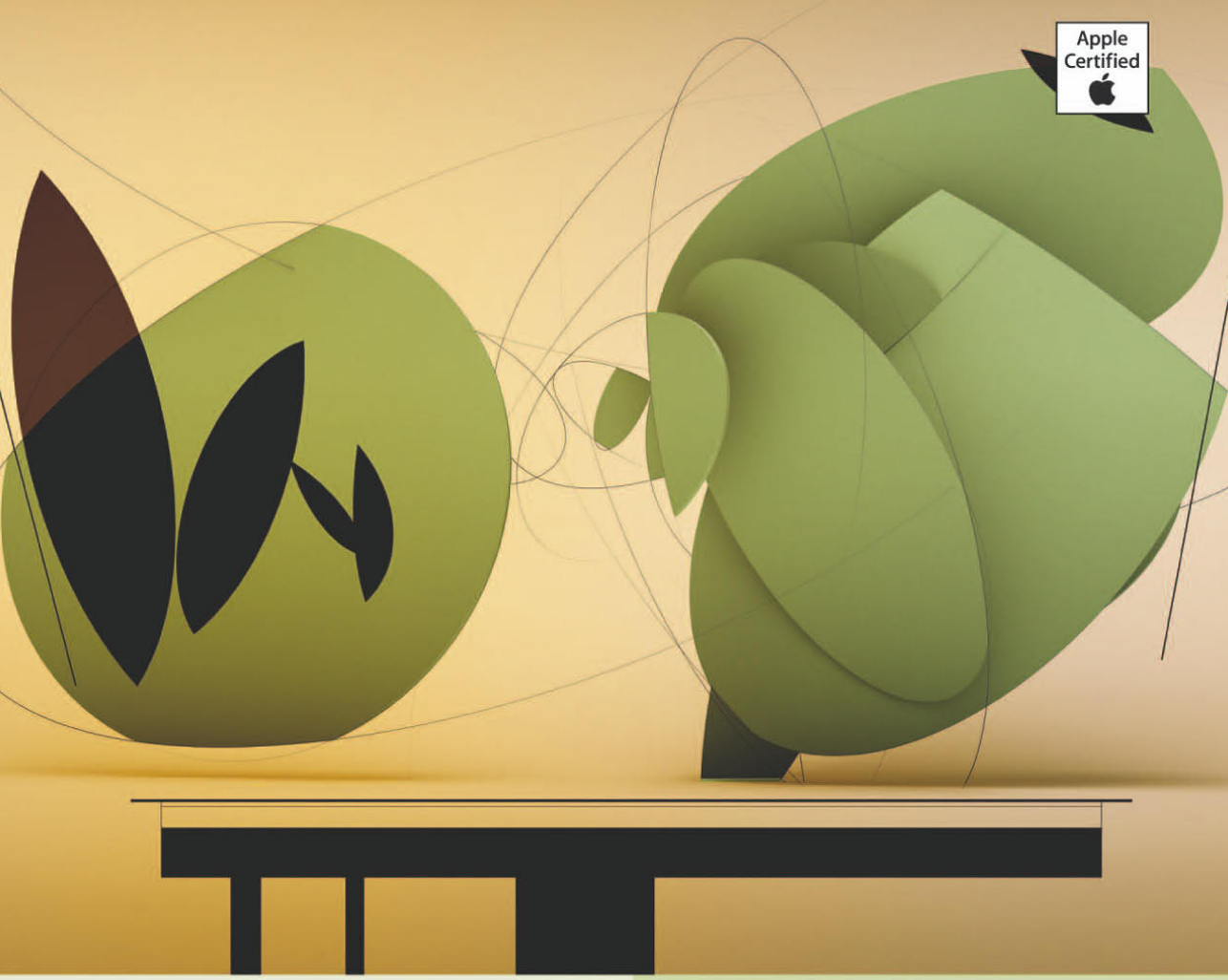


Apple Training Series

Mac OS X Support Essentials v10.6

Das offizielle Handbuch zu Mac OS X 10.6 für Administratoren, Help Desk und Support

Kevin M. White

ADDISON-WESLEY

3

Dauer

Diese Lektion dauert etwa 3 Stunden.

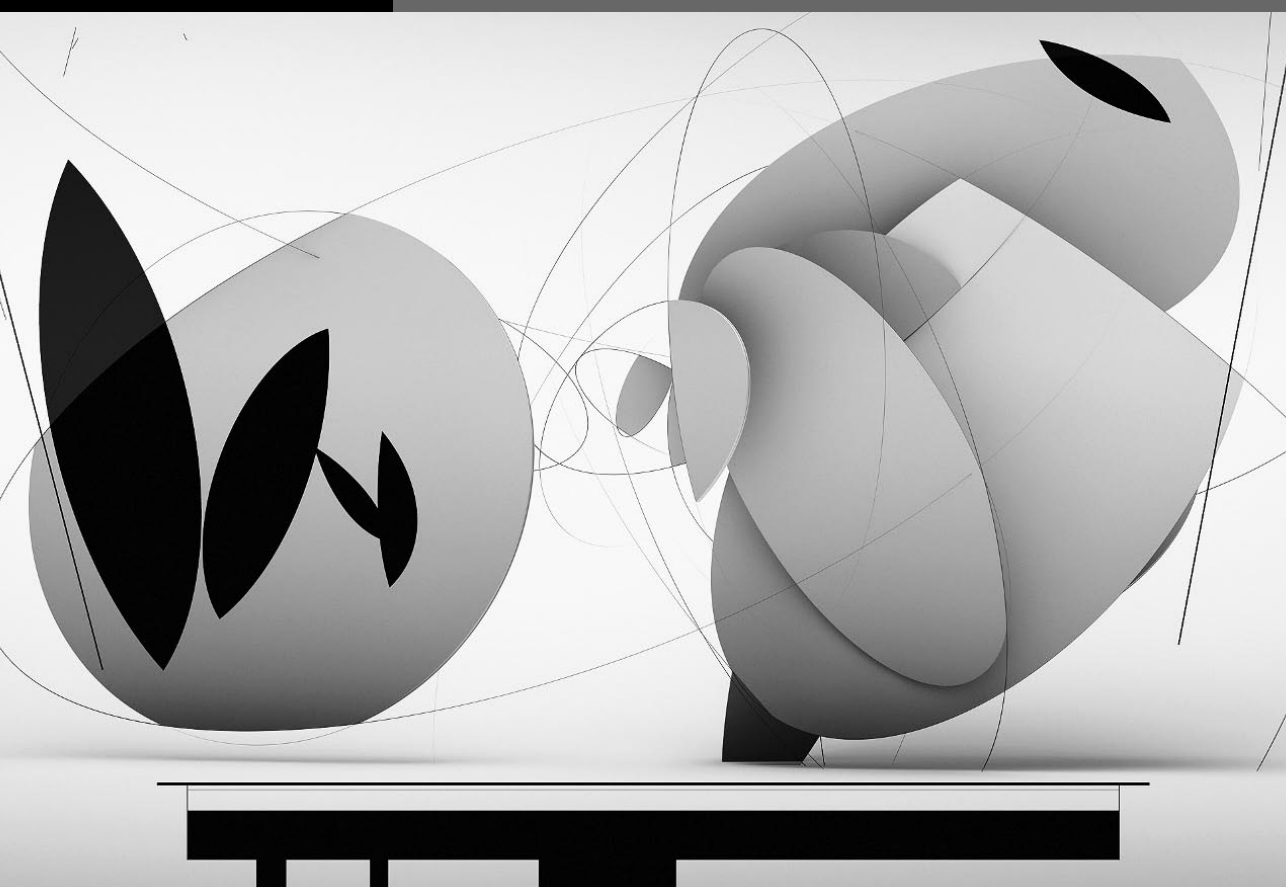
Ziele

Kennenlernen der Befehlszeilenumgebung

Verwenden der Befehlszeile für die grundlegende Dateimanipulation und Systemverwaltung

Verwenden von Automator und AppleScript für die Automatisierung von Aufgaben auf der grafischen Benutzeroberfläche

Verwenden von Skripten zum Automatisieren von Aufgaben in der Befehlszeile



Lektion 3

Befehlszeileneingabe und Automatisierung

Es lässt sich folgender Vergleich aufstellen: Sie können Ihr Leben lang Auto fahren, ohne zu wiBefehlssen, wie man es repariert. Wenn Sie aber als Automechaniker arbeiten möchten, müssen Sie verstehen, was unter der Motorhaube passiert. Das Gleiche gilt für Mac OS X: Sie arbeiten vielleicht schon Jahre mit dem Mac, ohne zu wissen, wie man Fehler beseitigt. Wenn Sie sich allerdings mit der Verwaltung und Fehlerbeseitigung beschäftigen möchten, müssen Sie auch hier wissen, was sich „unter der Haube“ abspielt. Im Fall von Mac OS X bedeutet „unter der Haube“ Arbeiten mit der Befehlszeile, da hier die grundlegenden Technologien von UNIX bereitgestellt werden.

Zunächst einmal gibt es viele unentbehrliche Werkzeuge für die Verwaltung und Fehlerbeseitigung, die nur über die Befehlszeile verfügbar sind. Sobald Sie mit der Befehlszeile etwas vertrauter sind, werden Sie feststellen, dass dort viele Aufgaben weitaus schneller erledigt werden können als auf der grafischen Benutzeroberfläche, insbesondere, wenn Sie wissen, wie man Skripte erstellt. Mit Skripten können Sie Routineaufgaben automatisieren und so schnell erledigen, wie es manuell nicht möglich ist. Bei Mac OS X sind Scripting-Technologien sowohl in die Befehlszeilenumgebung als auch in die grafische Benutzeroberfläche integriert.

In dieser Lektion erhalten Sie einen Einblick in die Befehlszeilenumgebung von Mac OS X. Sie lernen die Grundlagen der Befehlseingabe und Navigation in der Befehlszeile kennen sowie weitere allgemeine Befehle. Weiter hinten in dieser Lektion lernen Sie außerdem die Technologien des Befehlszeilen-Scripting und der Automatisierung in der grafischen Umgebung von Mac OS X kennen. Sie werden sehen, wie Sie und Ihre Benutzer mit diesen Werkzeugen bei der Erledigung von Routineaufgaben viel Zeit sparen können.

Grundlagen der Befehlszeileneingabe

Abgesehen davon, dass Sie mit Ihren Kenntnissen in der Befehlszeileneingabe andere beeindrucken können, gibt es mehrere gute Gründe zur Verwendung der Befehlszeile:

- Zusätzliche Optionen – In der Befehlszeile stehen Ihnen viele zusätzliche Optionen zur Verwaltung und Fehlerbeseitigung zur Verfügung, die in der grafischen Benutzeroberfläche nicht verfügbar sind. Die folgenden Programme z. B. haben ein Äquivalent in der Befehlszeile, das zusätzliche Optionen bietet (Befehle in Klammern): System-Profiler (system_profiler), Installationsprogramm (installer), Softwareaktualisierung (softwareupdate), Festplatten-Dienstprogramm (diskutil) und Spotlight (mdfind). Dies sind nur einige wenige Beispiele, da fast jede administrative Funktion ein Werkzeug sowohl in der grafischen als auch der Befehlszeilenumgebung hat.
- Finder-Beschränkungen – Im Gegensatz zum Finder, der absichtlich den Zugriff des Benutzers auf das gesamte Dateisystem einschränkt, haben Sie über die Befehlszeile uneingeschränkten Zugriff auf das Dateisystem. Im Finder sind z. B. viele Dateien und Ordner ausgeblendet, die in der Befehlszeile sichtbar sind. Auch viele Einstellungen für die Zugriffsrechte auf das Dateisystem kann der Finder nicht korrekt darstellen. Details zu den Zugriffsrechten im Dateisystem finden Sie in Lektion 4, „Dateisysteme“.
- „Unsichtbarer“ entfernter Zugriff – Sie können sich mithilfe des Secure-Shell-Protokolls (SSH) entfernt bei der Befehlszeilenumgebung am Mac anmelden, ohne dass der zurzeit in der grafischen Benutzeroberfläche angemeldete Benutzer etwas davon merkt. Dies ermöglicht Administratoren, Änderungen in der Befehlszeile vorzunehmen, ohne den Benutzer darauf aufmerksam zu machen. Die entfernte Anmeldung über SSH wird in Lektion 8, „Netzwerkdienste“, näher beschrieben.
- Zugriff für Administratoren als Systembenutzer (root) – Mit dem Befehl sudo kann sich jeder Administrator als Systembenutzer, auch bekannt als „root“, ausgeben. Dies ermöglicht enorme Flexibilität bei der Verwaltung über die Befehlszeile, wie weiter hinten in dieser Lektion beschrieben.

- Skriptfähig – Wenn Sie mit der interaktiven Arbeit in der Befehlszeile vertraut sind, können Sie dieselbe Syntax in einem Befehlszeilenskript anwenden. Dadurch können Sie Routineaufgaben ganz einfach automatisieren (wie weiter hinten in dieser Lektion beschrieben).
- Verwalten mehrerer Macs gleichzeitig – Wenn Sie die Anweisungen in der Befehlszeile mit Apple Remote Desktop (ARD) kombinieren, können Sie mehrere, sogar Tausende, Macs gleichzeitig entfernt verwalten. Mit ARD können Sie im Grunde mit nur einem Mausklick denselben Befehl entfernt an eine beliebige Anzahl von Macs senden. Dies bedeutet für alle, die mehrere Macs verwalten, eine enorme Zeitersparnis.

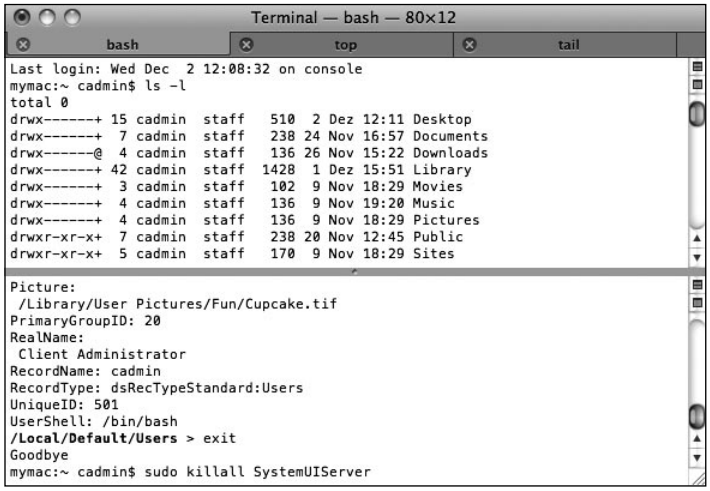
  WEITERE INFORMATIONEN ▶ ARD ist in Mac OS X nicht enthalten, der Mac stellt jedoch die Client-Version von ARD über den Dienst „Entfernte Verwaltung“ bereit. Mehr über ARD erfahren Sie in Lektion 8, „Netzwerkdienste“, oder unter http://www.apple.com/de/remotedesktop.

Zugriff auf die Befehlszeile

Wenn von der Befehlszeile gesprochen wird, werden Sie vielleicht in diesem Zusammenhang auch einmal den Begriff „Shell“ hören. Eine Shell ist der erste Befehl, der automatisch ausgeführt wird, wenn Sie auf die Befehlszeile zugreifen, und stellt Ihnen die eigentliche Befehlszeile bereit. Es sind verschiedene Arten von Shells verfügbar, doch Mac OS X startet standardmäßig mit der `bash`-Shell.

Die meisten Benutzer greifen auf die Mac OS X-Befehlszeile über das Programm „Terminal“ zu, aber es gibt mehrere Möglichkeiten:

- Terminal – Das Hauptprogramm in Mac OS X für den Zugriff auf die Befehlszeile befindet sich unter „/Programme/Dienstprogramme/Terminal“. Es ist technisch ausgereift und wurde im Lauf der Jahre durch viele praktische Funktionen ergänzt. Dazu gehören anpassbare Einstellungen für die Benutzeroberfläche, eine Oberfläche mit Titeln (Tabs) für den schnellen Zugriff auf mehrere Befehlszeilensitzungen und mehrere geteilte Fenster für eine übersichtliche Ansicht des Verlaufs.



- Eingeben von „>console“ im Anmeldefenster – Bei dieser Methode wird die grafische Benutzeroberfläche komplett umgangen und Sie sehen auf dem Bildschirm einen einfachen schwarzen Hintergrund und weißen Text. Sie müssen sich zwar immer noch mit Ihrem Benutzeraccount anmelden, aber dies ist eine praktische Methode, um nach Anmeldeproblemen zu suchen, da das Standard-Anmeldefenster wenig Aufschluss über Fehler gibt. Wenn Sie diesen Modus beenden möchten, verwenden Sie einfach den Befehl `exit`, um zum Standard-Anmeldefenster zurückzukehren.

HINWEIS ▶ Sie können den Konsolenmodus nur dann aktivieren, wenn Sie Ihren Benutzernamen im Anmeldefenster manuell eingeben können. Sie können in den Anmeldeoptionen der Systemeinstellung *Benutzer* die Option *Name und Kennwort* wählen, damit im Anmeldefenster standardmäßig Eingabefelder für den Namen und das Kennwort angezeigt werden. Alternativ haben Sie die Möglichkeit, mit dem Anmeldefenster in diesen Modus zu wechseln, indem Sie ⌘+⌥+→+↵ drücken.

- Starten im Einzelbenutzermodus – Hierbei handelt es sich um einen Modus zur Fehlerbeseitigung, den Sie durch Drücken von ⌘+S beim Systemstart aktivieren können. In diesem Modus wird das System mit den Mindestanforderungen gestartet, die zum Bereitstellen der Eingabeaufforderung erforderlich sind. Damit können Sie Befehle für die Instandsetzung eines Systems eingeben, das nicht vollständig gestartet werden kann. Der Einzelbenutzermodus wird in Lektion 10, „Systemstart", ausführlich beschrieben.
- Entfernte Anmeldung über SSH – Dies ermöglicht eine sichere Anmeldung über einen entfernten Computer für den Zugriff auf die Befehlszeile Ihres Macs. SSH ist ein gängiger Standard, d. h. jedes Betriebssystem, das SSH unterstützt, kann sich entfernt an Ihrem Mac anmelden. Die entfernte Anmeldung über SSH wird in Lektion 8, „Netzwerkdienste", näher beschrieben.

Arbeiten in der Befehlszeile

Wenn Anwender das Terminal zum ersten Mal öffnen, begegnen viele der Befehlszeile mit unnötiger Vorsicht. Obwohl die Befehlszeile nahezu unbegrenzte Möglichkeiten bietet, sind die Grundlagen nicht sehr schwer zu erlernen.

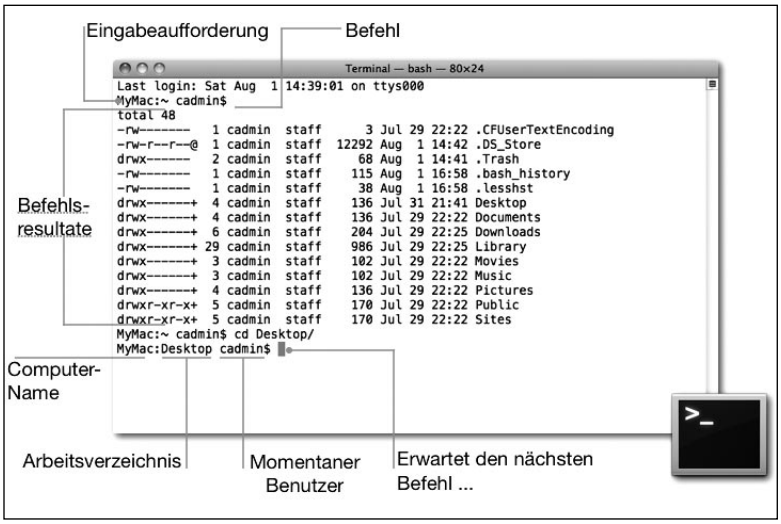
Eingabeaufforderung

Das Erste, was Sie in der Befehlszeile sehen, ist die Eingabeaufforderung. Sie wird vom Computer angezeigt, um Sie wissen zu lassen, dass er für den Empfang Ihres Befehls bereit ist. Standardmäßig wird in der Eingabeaufforderung immer der Name des verwendeten Computers angezeigt, wo Sie sich im Dateisystem befinden und als welcher Benutzer Sie angemeldet sind, d. h. Ihr Benutzeraccountname. Die Zeile endet mit einem Dollarzeichen ($). Das Dollarzeichen ($) am Ende der Eingabeaufforderung zeigt an, dass Sie die standardmäßige `bash`-Shell verwenden. Der Ort, an dem Sie sich im Dateisystem befinden, wird als Arbeitsverzeichnis bezeichnet und ändert sich entsprechend, wenn Sie im Dateisystem navigieren.

> **HINWEIS ▶** Während in Mac OS die Dateisystem-Container als „Ordner" bezeichnet werden, wird in der Befehlszeile der Begriff „Verzeichnis" verwendet, der ein Hinweis auf den UNIX-Ursprung ist. In Rahmen dieser Lektion werden diese beiden Begriffe synonym verwendet.

Nach der Eingabeaufforderung geben Sie Ihren Befehl ein (häufig mehr als ein Wort) und drücken dann den Zeilenschalter, um den eingegebenen Befehl zu initiieren oder auszuführen. Je nach Befehl wird das Terminal-Fenster in einen Texteingabebereich verwandelt, die Ergebnisse des Befehls werden angezeigt und anschließend sehen Sie erneut die Eingabeaufforderung oder es werden Aufgaben ausgeführt und die Eingabeaufforderung wird wieder angezeigt, wenn der Befehl abgeschlossen ist. Viele Befehle liefern nur dann Ergebnisse, wenn Probleme aufgetreten sind. Deshalb sollten Sie sich die Befehlsausgabe immer genau ansehen, um sicherzustellen, dass kein Fehler aufgetreten ist.

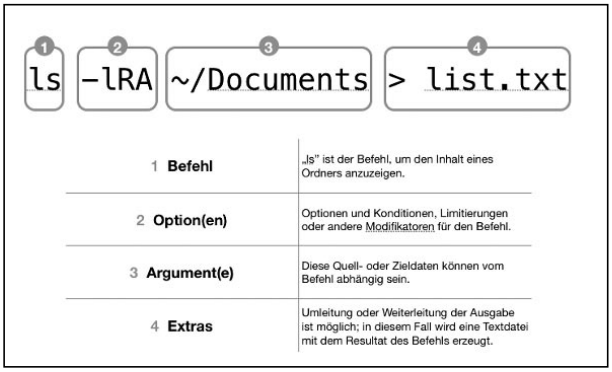
HINWEIS ▶ Bei einigen Befehlen dauert die Ausführung u. U. etwas länger und es wird auch keine Statusmeldung angezeigt. Allgemein gilt: Wenn Sie keine neue Eingabeaufforderung sehen, wird Ihr letzter Befehl wahrscheinlich noch ausgeführt.



Befehle

Der eigentliche Befehl besteht im Allgemeinen aus ein paar wenigen Elementen:

- Befehlsname – Befehle sind wie Programme, wobei sie allerdings auf spezifischere Funktionen ausgelegt sind. Bei einigen Befehlen müssen Sie nur den Namen des Befehls eingeben, damit er ausgeführt wird.
- Befehlsoptionen (auch „Flags“ genannt) – Nach dem Befehl können Sie einige Optionen eingeben, die das Standardverhalten des Befehls ändern. Diese optionalen Angaben sind nicht erforderlich und können für jeden Befehl unterschiedlich sein. Optionen beginnen mit einem oder zwei Bindestrichen, um sie von Argumenten zu unterscheiden. Bei vielen Befehlen können Sie nach einem einzelnen Bindestrich mehrere Einzelbuchstaben als Option eingeben. `ls -lA` ist z. B. dasselbe wie `ls -l -A`.
- Argumente (auch „Parameter“ genannt) – Nach dem Befehl und den Optionen geben Sie normalerweise das Argument bzw. das Objekt ein, das mit dem Befehl verändert werden soll. Auch in diesem Fall wird ein Argument nur benötigt, wenn für die Ausführung des Befehls ein Objekt erforderlich ist.
- Extras – Extras sind nicht erforderlich, sie können aber die Möglichkeiten, die Ihnen ein Befehl bietet, enorm erweitern. Sie könnten z. B. Objekte hinzufügen, die die Befehlsausgabe umleiten, andere Befehle hinzufügen oder ein Dokument erstellen.



Ein einfaches Beispiel für die Befehlszeile

Im folgenden Beispiel arbeitet die Benutzerin Michelle an einem Computer namens „MyMac". Ihr Arbeitsverzeichnis lautet „Documents" (Dokumente). Sie löscht ein Programm namens „Junk" innerhalb des Programmordners („Applications"). Es wird angenommen, dass Michelle den Zeilenschalter oder die ↵-Taste drückt, sobald sie den Befehl eingegeben hat.

```
MyMac:Documents michelle$ rm -R /Applications/Junk.app
MyMac:Documents michelle$
```

HINWEIS ▶ In diesem Handbuch wird Text, den der Benutzer eingibt, hervorgehoben. Dadurch lässt er sich leicht von dem Text unterscheiden, den der Computer erzeugt. Hierbei ist zu beachten, dass es sich lediglich um Beispielbefehle handelt. Höchstwahrscheinlich müssen Sie den Befehl an Ihre Arbeitsumgebung anpassen.

Im vorliegenden Beispiel wurde der Befehl richtig eingegeben und ausgeführt und der Computer zeigt anschließend eine neue Eingabeaufforderung an. Dies ist ein Beispiel für einen Befehl, bei dem nur dann Daten ausgegeben werden, wenn der Befehl nicht korrekt ausgeführt wurde. Normalerweise lässt Sie der Computer wissen, wenn Sie etwas Falsches eingegeben haben, indem er eine Fehlermeldung oder einen Hilfetext anzeigt. Der Computer bewahrt Sie jedoch nicht vor der Eingabe unüberlegter Befehle, wie beispielsweise dem unbeabsichtigten Löschen Ihres Benutzerordners. Wenn Sie sich eine Regel im Umgang mit der Befehlszeile merken sollten, dann ist es diese: Überprüfen Sie alle Eingaben ganz genau.

Über Befehle

Es gibt buchstäblich Tausende von Befehlen mit dutzenden Optionen oder Anforderungen für die korrekte Anwendung. Tatsächlich sind viele Benutzer mit der Befehlszeile überfordert, da sie annehmen, sie müssten sich die Befehle merken, um sie zu verwenden. Eigentlich müssen Sie nur einen Befehl kennen: `man`.

Die meisten Befehle haben Anleitungen, die alles Wissenswerte enthalten. Geben Sie einfach `man` und den Namen des Befehls ein, der Sie interessiert, und es wird die zugehörige man-Seite („man page"; kurz für „manual page"), also die Handbuchseite, angezeigt. Diese Seiten enthalten Verwendungszwecke des Befehls. Außerdem sind am Ende der Seite Verweise zu ähnlichen Befehlen enthalten. Wenn Sie sich innerhalb der Anzeige der man-Seiten befinden (die Sie automatisch zum Befehl `less` umleitet), können Sie mit einigen Tastaturkurzbefehlen in der entsprechenden Anleitung navigieren:

- Mit den Aufwärtspfeil- und Abwärtspfeiltasten blättern Sie zeilenweise durch die man-Seite.
- Mit der Leertaste gehen Sie eine Seite nach unten.
- Suchen Sie in der Anleitung, indem Sie `/` und ein Schlagwort eingeben.
- Mit der Taste Q verlassen Sie die man-Seite.

Was ist, wenn Sie nicht einmal den Namen des Befehls kennen, den Sie suchen? Geben Sie einfach `man -k` und ein Schlagwort ein, um die Datenbank mit den man-Seiten zu durchsuchen. Beispielsweise zeigt die Eingabe von `man -k owner` eine Liste der Befehle an, die für die Änderung von Datei- und Ordnereigentümern verwendet werden kann. Darunter finden Sie den Befehl `chown`. Wie Sie mit dem Befehl `chown` Eigentümer von Dateien und Ordnern ändern, erfahren Sie in Lektion 4, „Dateisysteme".

> **HINWEIS ▶** Neben den Befehlen enthält die Datenbank mit den man-Seiten auch Scripting- und Programmfunktionen sowie Dokumentation zu Dateiformaten. Somit liefert eine Suche mit `man -k` u. U. noch andere Ergebnisse als nur die relevanten Befehle.

Navigieren mit der Befehlszeile

Zur Beherrschung der Befehlszeile gehört neben der richtigen Verwendung der Befehle auch die effektive Navigation durch das Dateisystem. Auch hier ist es so, dass Sie, wenn Sie die Navigation mithilfe der Befehlszeile beherrschen, deutlich schneller vorankommen als mit dem Finder.

> **HINWEIS ▶** In der Befehlszeile wird zwischen Groß- und Kleinschreibung unterschieden und es müssen vollständige Dateinamen mit Suffix angegeben werden. Beispielsweise kann die Befehlszeile das Programm „itunes" nicht finden, sondern nur „iTunes.app".

Konzepte der Navigation

Zunächst müssen einige Navigationsbegriffe klar definiert werden. Während in Mac OS die Dateisystem-Container als „Ordner“ bezeichnet werden, wird im Zusammenhang mit der Befehlszeile der Begriff „Verzeichnis“ verwendet, der ein Hinweis auf den UNIX-Ursprung ist. Obwohl beide Begriffe verwendet werden können, wird hier weiterhin das Wort „Ordner“ verwendet, um Dateisystem-Container zu bezeichnen. Der Begriff „Verzeichnis“ wird häufig für Objekte verwendet, bei denen es sich nicht um ordnerähnliche Objekte handelt. Beispielsweise werden Netzwerkdatenbanken, in denen Benutzerinformationen gesichert werden, oftmals als „Verzeichnis“ bezeichnet. Außerdem wird der Prozess für den Zugriff auf diese Benutzerdatenbanken in Mac OS X als `DirectoryService` bezeichnet.

Ein neuer Begriff in dieser Lektion ist Pfad. Ein Pfad repräsentiert die Position einer Datei oder eines Ordners im Dateisystem, d. h. er führt Sie zu diesem Ort. Sie haben Pfade in diesem Buch bereits gesehen, die die genaue Position eines Programms oder Dienstprogramms beschrieben haben. Beispielsweise ist der Pfad für das Festplatten-Dienstprogramm „/Programme/Dienstprogramme/Festplatten-Dienstprogramm“. Die Befehlszeile verwendet Pfadnamen ausschließlich für die Navigation und Suche von Objekten im Dateisystem.

Es gibt zwei Arten von Pfadnamen innerhalb des Dateisystems: absolute und relative. Beide können zur Navigation oder Suche von Objekten in der Befehlszeile verwendet werden, doch besteht der Unterschied in ihrem Ausgangspunkt:

- Absolute Pfade – Diese Pfade sind eine exakte Beschreibung der Position eines Objekts, wobei im root-Verzeichnis (der obersten Ebene) des System- bzw. Startvolumes begonnen wird. Daher fängt ein absoluter Pfad immer mit einem Schrägstrich (/) an, der die oberste Ebene des Dateisystems bezeichnet. In diesem Buch werden absolute Pfade verwendet, um die Position von Objekten zu beschreiben. Beispielsweise lautet der Pfad zum Briefkasten der Benutzerin Michelle „/Users/michelle/Public/Drop Box“ (/Benutzer/michelle/Öffentlich/Briefkasten). Dieser Pfad hat folgende Bedeutung: Beim Startvolume beginnen, weiter zum Ordner „Benutzer“ und dann in den Unterordner „michelle“, von dort in den Unterordner „Öffentlich“ und das Objekt „Briefkasten“ auswählen.
- Relative Pfade – Diese Pfade sind eine teilweise Beschreibung der Position eines Objekts basierend auf der derzeitigen Position innerhalb des Dateisystems. Wenn Sie das Programm „Terminal“ zum ersten Mal öffnen, beginnt diese Sitzung auf der obersten Ebene Ihres Benutzerordners. Daher lautet der relative Pfad von Ihrem Benutzerordner zu Ihrem Briefkasten-Ordner „Public/Drop Box“ (Öffentlich/Briefkasten). Dieser Pfad hat folgende Bedeutung: Vom Ausgangspunkt zum Unterordner „Öffentlich“ und dann das Objekt „Briefkasten“ auswählen.

Verwenden von Navigationsbefehlen

Sie verwenden in der Befehlszeile drei grundlegende Befehle für die Navigation im Dateisystem: `pwd`, `ls` und `cd`.

pwd

Kurzform von „print working directory". Dieser Befehl zeigt den absoluten Pfad Ihrer derzeitigen Position im Dateisystem an:

```
MyMac:~ michelle$ pwd
/Users/michelle
```

ls

Kurzform von „list". `ls` zeigt den Inhalt des Ordners Ihrer aktuellen Position an. Die Eingabe eines Pfadnamens nach dem Befehl `ls` listet den Inhalt des entsprechenden Objekts auf. Dieser Befehl hat viele zusätzliche Optionen für die Auflistung von Datei- und Ordnerinformationen, die in diesem Buch behandelt werden.

```
MyMac:~ michelle$ ls
Desktop Library Pictures
Documents Movies Public
Downloads Music Sites
MyMac:~ michelle$ ls Public
Drop Box
```

cd

Kurzform von „change directory". Der Befehl `cd` dient zur Navigation in der Befehlszeile. Die Eingabe eines Pfadnamens nach dem Befehl `cd` ändert Ihre aktuelle Position auf den angegebenen Ordner. Die Eingabe von `cd` ohne Pfadangabe bringt Sie immer zurück zu Ihrem Benutzerordner. Im folgenden Beispiel verwendet Michelle den Befehl `cd`, um zu ihrem Briefkasten zu gelangen. Anschließend navigiert sie zu ihrem Benutzerordner zurück.

```
MyMac:~ michelle$ cd Public/Drop\ Box/
MyMac:Drop Box michelle$ pwd
/Users/michelle/Public/Drop Box
MyMac:Drop Box michelle$ cd
MyMac:~ michelle$ pwd
/Users/michelle
```

Verwenden von Sonderzeichen

An diesem Punkt haben Sie vielleicht bemerkt, dass in der Befehlszeile Sonderzeichen in der Eingabeaufforderung und den Pfadnamen verwendet werden. Viele dieser Sonderzeichen werden als Abkürzungen eingesetzt, um Zeit zu sparen. Andererseits stellt eines dieser Sonderzeichen keine Zeitersparnis dar, sondern eher eine notwendiges Übel. Der Backslash „\" wird vor einem Leerzeichen in einem Pfad- oder Dateinamen verwendet. Dies ist notwendig, da die Befehlszeile das Leerzeichen zwischen Objekten benötigt, um den Befehl in einzelne logische Teile zu unterteilen. Ein Leerzeichen in einem Dateinamen ohne Angabe eines Backslash würde zu einem Fehler führen, sodass der Befehl nicht richtig ausgeführt würde.

Es gibt andere Methoden für die Eingabe von Dateinamen und Pfaden mit Leerzeichen. Eine Alternative ist das Einschließen von Dateinamen und Pfaden in Anführungszeichen:

```
MyMac:~ michelle$ cd "Public/Drop Box"
MyMac:Drop Box michelle$ pwd
/Users/michelle/Public/Drop Box
```

Eine andere Lösung besteht darin, Objekte vom Finder per Drag&Drop in das Terminal-Fenster zu bewegen. Terminal fügt so den absoluten Pfad des Objekts mit den richtigen Backslash-Zeichen vor einem Leerzeichen im Namen ein. Die effizienteste Lösung ist jedoch die Verwendung der in Mac OS X integrierten Funktion, die Datei- und Pfadnamen automatisch vervollständigt. Wie Sie mit dieser Funktion Zeit sparen können, erfahren Sie im nächsten Abschnitt dieser Lektion.

> **HINWEIS ▶** Das Leerzeichen ist nicht das einzige Zeichen in der Befehlszeile, das gesondert behandelt werden muss. Andere sind !, $, &, *, ;, | und \ sowie runde Klammern und alle Arten von Anführungszeichen und eckigen Klammern. Sowohl Drag&Drop im Finder als auch die automatische Vervollständigung mithilfe der Tabulatortaste (Tab Completion) verarbeiten diese Zeichen richtig.

Wenn Sie sich durch das Dateisystem bewegen, können Sie auch mit der Verwendung eines doppelten Punkts „`..`" Zeit sparen, der den übergeordneten Ordner bezeichnet. Wenn Sie beispielsweise an der Stelle „/Users/<Benutzername>" in Ihrem Benutzerordner arbeiten, bewirkt die Eingabe von `cd ..` den Wechsel zum darüberliegenden Ordner „/Users". Im folgenden Beispiel wechselt Michelle zu ihrem Briefkasten, kehrt zu ihrem öffentlichen Ordner und anschließend zum Ordner „/Users" zurück:

```
MyMac:~ michelle$ cd Public/Drop\ Box/
MyMac:Drop Box michelle$ pwd
/Users/michelle/Public/Drop Box
```

```
MyMac:Drop Box michelle$ cd ..
MyMac:Public michelle$ pwd
/Users/michelle/Public
MyMac:Public michelle$ cd ../..
MyMac:Users michelle$ pwd
/Users
```

Schließlich gibt es noch die Tilde (~). Dieses Zeichen wird als Abkürzung verwendet, um den aktuellen Benutzerordner in einem Pfadnamen zu beschreiben. Im vorherigen Beispiel befindet sich der Briefkasten des aktuellen Benutzers unter „~/Public/Drop Box". Dies erklärt das Vorhandensein der Tilde in der Standard-Befehlseingabeaufforderung. Wenn Michelle beispielsweise Terminal geöffnet hat und sich auf einem Computer namens „MyMac" anmeldet, wird Folgendes in der Eingabeaufforderung angezeigt:

```
MyMac:~ michelle$
```

TIPP Mit der Tilde können Sie auch den Benutzerordner eines anderen Benutzers angeben. `~logan/Public` würde z. B. Logans öffentlichen Ordner beschreiben.

Wenn Michelle zu ihrem Briefkasten navigiert, ändert sich die Eingabeaufforderung. Beachten Sie, dass die Eingabeaufforderung nur die aktuelle Position und keinen absoluten oder relativen Pfad anzeigt:

```
MyMac:Drop Box michelle$
```

Automatisches Vervollständigen

Dies ist die Hauptfunktion der Befehlszeile, um Zeit zu sparen. Nicht nur, dass die automatische Vervollständigung Zeit spart, indem sie automatisch Dateinamen, Pfadnamen und Befehle für Sie vervollständigt, sie bewahrt Sie zusätzlich auch vor Schreibfehlern und überprüft, ob die eingegebenen Objekte auch existieren.

Die Verwendung der automatischen Vervollständigung ist denkbar einfach. Beginnen Sie in Ihrem Benutzerordner und geben Sie `cd` und danach `P` ein und drücken Sie dann die Tab-Taste. Das Terminal-Fenster blinkt kurz (Sie hören u. U. auch ein Signal), was darauf hinweist, dass es mehr als eine Möglichkeit für Objekte in Ihrem Benutzerordner gibt, die mit „P" beginnen. Drücken Sie erneut die Tab-Taste und der Computer zeigt zwei Möglichkeiten an, „Pictures" und „Public". Wenn Sie jetzt ein `u` nach dem P eingeben und anschließend erneut die Tab-Taste drücken, gibt der Computer automatisch „Public/" für Sie ein. Geben Sie abschließend ein `D` ein und drücken Sie ein letztes Mal die Tab-Taste. Der Computer schließt den Pfad mit „Public/Drop\ Box/" ab.

HINWEIS ▶ Wenn ein Ordnername automatisch vervollständigt wird, wird automatisch ein Schrägstrich (/) am Ende hinzugefügt (das Programm geht davon aus, dass Sie den Pfad von dort fortsetzen möchten). Die meisten Befehle ignorieren diesen angehängten Schrägstrich, einige reagieren jedoch darauf und liefern andere Ergebnisse. Wenn Sie sich nicht sicher sind, ist es in der Regel sinnvoller, den Schrägstrich am Ende des Pfads zu löschen.

Selbst in diesem kleinen Beispiel hat die automatische Vervollständigung einen Pfadnamen von 17 (`Public/Drop\ Box/`) auf nur 5 (`Pu<tab>D<tab>`) Tastenanschläge reduziert. Außerdem war sie bei der Vermeidung von Fehlern behilflich, indem sie die Eingaben überprüft und festgestellt hat, ob sich das Objekt dort befindet, wo Sie es vermutet haben. Wenn Sie sich angewöhnen, diese Vervollständigungsmethode in der Befehlszeile zu nutzen, können Sie leicht etliche Stunden einsparen. Daher sollten Sie sich bei der Verwendung der Befehlszeile an zwei Regeln halten: Überprüfen Sie Ihre Eingaben sehr genau und nutzen Sie immer die automatische Vervollständigung, um Tippfehler zu vermeiden und Zeit zu sparen.

Anzeigen von unsichtbaren Objekten

Um die Navigation durch das Dateisystem zu erleichtern, blenden die Befehlszeile und der Finder viele Dateien aus. Häufig sind dies systemrelevante Objekte die aus gutem Grund ausgeblendet werden. Im Finder gibt es keinen einfachen Weg, die ausgeblendeten Objekte anzeigen zu lassen, wohingegen dies in der Befehlszeile sehr einfach ist. Im Finder werden viele Objekte ausgeblendet, da sie als unsichtbare Dateien markiert sind. In der Befehlszeile wird diese Markierung (Flag) hingegen ignoriert, sodass diese Objekte trotzdem angezeigt werden. Mit dem Befehl `ls` werden jedoch Objekte mit Dateinamen ausgeblendet, die mit einem Punkt beginnen. Aber auch diese Objekte können ganz einfach wieder eingeblendet werden.

Um unsichtbare Objekte in der Befehlszeile anzuzeigen, fügen Sie einfach die Option `-a` zu `-l` hinzu, wenn Sie den Befehl `ls` verwenden:

```
MyMac:~ michelle$ ls -la
total 32
drwxr-xr-x  15 michelle  staff   510 Aug 20 17:33 .
drwxr-xr-x   8 root      admin   272 Aug 20 17:05 ..
-rw-------   1 michelle  staff     3 Aug 20 01:08 CFUserTextEncoding
-rw-------   1 michelle  staff  2666 Aug 20 16:42 .bash_history
-rw-------   1 michelle  staff    48 Aug 20 17:19 .lesshst
-rw-------   1 root      staff   632 Aug 20 14:25 .viminfo
drwx------+  5 michelle  staff   170 Aug 20 15:49 Desktop
```

```
drwx------+  3 michelle  staff   102 Aug 20 01:08 Documents
drwx------+  3 michelle  staff   102 Aug 20 01:08 Downloads
drwx------  19 michelle  staff   646 Aug 20 01:08 Library
drwx------+  3 michelle  staff   102 Aug 20 01:08 Movies
drwx------+  3 michelle  staff   102 Aug 20 01:08 Music
drwx------+  4 michelle  staff   136 Aug 20 01:08 Pictures
drwxr-xr-x+  7 michelle  staff   238 Aug 20 15:29 Public
drwxr-xr-x   5 michelle  staff   170 Aug 20 01:08 Sites
```

HINWEIS ▶ Mit `ls -a` werden alle Objekte im Ordner eingeblendet, einschließlich „`..`" (eine Kurzform für den übergeordneten Ordner) und „`.`" (eine Kurzform für den aktuellen Ordner). Der Befehl `ls -A` zeigt alle unsichtbaren Objekte außer diesen beiden Kurzformen für die Ordner an.

Wie Sie in Michelles Benutzerordner sehen können, wird jedes Objekt, das mit einem Punkt (.) beginnt, im Finder und in der Befehlszeile standardmäßig ausgeblendet. Diese Objekte werden vom Betriebssystem erstellt und von ihm verwendet. Daher sollten sie auch nicht modifiziert werden.

WEITERE INFORMATIONEN ▶ Wie Sie unsichtbare Objekte verwalten, wird in Lektion 5, „Datenverwaltung und Backup", näher beschrieben.

Navigieren zu anderen Volumes

In der Befehlszeile wird das Systemvolume, das auch root-Verzeichnis genannt wird, durch eine einzelnen Schrägstrich (/) dargestellt. Es mag zunächst überraschen, dass in der Befehlszeile andere Volumes, die nicht zum Systemvolume gehören, als Teil des Hauptdateisystems in einem Ordner namens „Volumes" erscheinen. Im folgenden Beispiel beginnt Michelle in ihrem Benutzerordner, wechselt zu „/Volumes" und listet die enthaltenen Objekte auf. Anschließend wechselt sie in ein Volume namens „Backup Drive", das über FireWire angeschlossen ist.

```
MyMac:~ michelle$ pwd
/Users/michelle
MyMac:~ michelle$ cd /Volumes/
MyMac:Volumes michelle$ pwd
/Volumes
MyMac:Volumes michelle$ ls
Backup Drive            Macintosh HD
Mac OS X Install DVD
MyMac:Volumes michelle$ cd Backup\ Drive/
MyMac:Backup Drive michelle$ pwd
/Volumes/Backup Drive
```

Dateimanipulation mit der Befehlszeile

Die grundlegende Dateiverwaltung ist von der Befehlszeile aus umfassender als im Finder. Leider führt die Dateiverwaltung in der Befehlszeile durch falsche Benutzereingaben u. U. zu erhöhtem Fehleraufkommen. Stellen Sie deshalb sicher, dass Sie Ihre Eingaben sorgfältig überprüfen, bevor Sie einen Befehl ausführen.

Befehle zur Dateiüberprüfung

Es gibt eine Vielzahl grundlegender Befehle, mit denen sich Dateien und Ordner von der Befehlszeile aus finden und überprüfen lassen, z. B. `cat`, `less`, `which`, `file` und `find`. Wie immer erhalten Sie ausführliche Informationen über die einzelnen Befehle, indem Sie deren man-Seite lesen.

cat

Dies ist die Abkürzung für „concatenate", also „verketten". Dieser Befehl liest eine Datei und gibt sie sequenziell über die Standardausgabe aus, meistens also im Terminal-Fenster. Die Syntax lautet `cat`, gefolgt von dem Pfad zu dem Objekt, das Sie anzeigen möchten. Mit dem Befehl `cat` können Sie auch Textdateien verketten, indem Sie den Umleitungsoperanden `>>` verwenden. Im folgenden Beispiel lässt sich Michelle den Inhalt der beiden Textdateien „TextDocOne.txt" und „TextDocTwo.txt" aus ihrem Schreibtisch-Ordner mithilfe von `cat` anzeigen. Anschließend verwendet sie `cat` mit dem Umleitungsoperanden `>>`, um die zweite Textdatei an das Ende der ersten anzuhängen.

```
MyMac:~ michelle$ cat Desktop/TextDocOne.txt
Dies ist der Inhalt des ersten Textdokuments.
MyMac:~ michelle$ cat Desktop/TextDocTwo.txt
Dies ist der Inhalt des zweiten Textdokuments.
MyMac:~ michelle$ cat Desktop/TextDocTwo.txt >> Desktop/TextDocOne.txt
MyMac:~ michelle$ cat Desktop/TextDocOne.txt
Dies ist der Inhalt des ersten Textdokuments.
Dies ist der Inhalt des zweiten Textdokuments.
```

WEITERE INFORMATIONEN ▶ Zusätzliche Optionen in der Befehlszeile, wie der Umleitungsoperand >>, werden in dieser Lektion im Abschnitt „Grundlagen des Befehlszeilen-Scripting" näher erläutert.

less

Eine Anspielung auf den früher beliebten Befehl `more` zur Textanzeige. Der Befehl `less` eignet sich weitaus besser zum Anzeigen langer Textdateien, da Sie sich hierbei interaktiv durch den Text bewegen und darin suchen können. Die Syntax lautet `less`, gefolgt von dem Pfad zu dem Objekt, das Sie anzeigen möchten. Der Befehl `less` verwendet die gleiche Schnittstelle, die Sie auch zur Anzeige von man-Seiten verwenden. Deshalb sind alle Navigationskurzbefehle mit denen des Befehls `man` identisch:

- Mit den Aufwärtspfeil- und Abwärtspfeiltasten blättern Sie durch den Text.
- Mit der Leertaste gehen Sie eine Seite nach unten.
- Wenn Sie `/` und dann einen Suchbegriff eingeben, können Sie den Text durchsuchen.
- Geben Sie „v“ ein, um die Textdatei automatisch an den Texteditor `vi` umzuleiten. Auf die Verwendung von `vi` wird weiter hinten in dieser Lektion näher eingegangen.
- Um `less` zu beenden, drücken Sie einfach Q.

HINWEIS ▶ Wenn Sie in Mac OS X den Befehl `more` ausführen, wird eigentlich der Befehl `less` ausgeführt, nur mit etwas anderen Optionen. Der Befehl `more` wird z. B. automatisch beendet, wenn das Ende eines Dokuments erreicht wird, während Sie `less` explizit beenden müssen.

which

Mit diesem Befehl finden Sie den Dateipfad eines angegebenen Befehls. Das heißt, er zeigt an, welche Datei Sie letztlich verwenden, wenn Sie den betreffenden Befehl eingeben. Die Syntax lautet `which`, gefolgt von den Befehlen, deren Pfad Sie finden möchten. Im folgenden Beispiel verwendet Michelle den Befehl `which`, um den Dateipfad der Befehle `man`, `ls`, `pwd` und `cd` herauszufinden:

```
MyMac:~ michelle$ which man ls pwd cd
/usr/bin/man
/bin/ls
/bin/pwd
/usr/bin/cd
```

Sie werden feststellen, dass sich die meisten Befehle in einem von vier Ordnern befinden: „/usr/bin“ für die meisten Befehle, „/usr/sbin“ für systemorientierte Befehle, „/bin“ für wichtige Befehle, die das System während des Startvorgangs benötigt, und „/sbin“ für wichtige systemorientierte Befehle.

file

Dieser Befehl versucht, den Typ einer Datei anhand ihres Inhalts herauszufinden. Dies ist ein sehr nützlicher Befehl, um Dateien zu identifizieren, die kein Dateisuffix haben. Die Syntax lautet `file`, gefolgt von dem Pfad zu der Datei, die Sie identifizieren möchten. Im folgenden Beispiel verwendet Michelle `file`, um den Dateityp der beiden Dokumente „PictureDocument“ und „TextDocument“ auf ihrem Schreibtisch herauszufinden:

```
MyMac:~ michelle$ ls Desktop/
PictureDocument TextDocument
MyMac:~ michelle$ file Desktop/PictureDocument
Desktop/PictureDocument: TIFF image data, big-endian
MyMac:~ michelle$ file Desktop/TextDocument
Desktop/TextDocument: ASCII English text
```

find

Mit diesem Befehl können Sie Objekte im Dateisystem anhand von Suchkriterien finden. Der Befehl `find` verwendet nicht den Suchdienst „Spotlight“, aber Sie können sehr spezifische Suchkriterien angeben und Platzhalter für Dateinamen verwenden (die Platzhalter werden im nächsten Abschnitt erläutert). Die Syntax lautet `find`, gefolgt vom Ausgangspfad der Suche, einer Option, die die Suchkriterien definiert, und schließlich den Suchkriterien in Anführungszeichen. Im folgenden Beispiel verwendet Michelle `find`, um alle Bilddateien in ihrem Benutzerordner zu finden. Dazu sucht sie nach Dateien mit dem Suffix .tiff:

```
MyMac:~ michelle$ find /Users/michelle/ -name "*.tiff"
/Users/michelle//Desktop/PictureDocument.tiff
/Users/michelle//Pictures/FamilyPict.tiff
/Users/michelle//Pictures/MyPhoto.tiff
```

TIPP Wenn Sie den Befehl `find` verwenden, um eine Suche auf der obersten Ebene des Systemlaufwerks zu beginnen, sollten Sie auch die Option `-x` verwenden, damit Sie nicht im Ordner „/Volumes“ suchen.

TIPP Um den Spotlight-Suchdienst von der Befehlszeile aus zu verwenden, verwenden Sie den Befehl `mdfind`. Die Syntax lautet einfach `mdfind`, gefolgt von den Suchkriterien.

Verwenden von Platzhaltern

Eine der leistungsfähigsten Funktionen der Befehlszeile ist die Möglichkeit, Platzhalter zu verwenden, um Pfadnamen und Suchkriterien zu verwenden. Die drei am häufigsten verwendeten Platzhalter sind:

- Stern (*) – Der Stern wird als Platzhalter für jede beliebige Zeichenfolge verwendet. Die Eingabe von `*` ermittelt alle Dateien als Übereinstimmung, `*.tiff` ermittelt alle Dateien, die die Endung .tiff haben.
- Fragezeichen (?) – Das Fragezeichen ist ein Platzhalter für ein einzelnes Zeichen. Die Eingabe `b?ch` stimmt z. B. mit buch überein, aber nicht mit bruch.
- Eckige Klammern ([]) – Eckige Klammern werden verwendet, um einen Bereich von Zeichen anzugeben, aus dem das Zeichen an der fraglichen Stelle stammen muss. So finden Sie mit `[Dd]ocument` alle Objekte, die „Document“ oder „document“ heißen. Die Eingabe `doc[1-9]` ermittelt alle Dateinamen, die mit „doc“ beginnen und dahinter eine Zahl zwischen 1 und 9 aufweisen.

Platzhalter lassen sich sehr wirkungsvoll kombinieren. Stellen Sie sich eine Sammlung von fünf Dateien namens „ReadMe.rtf“, „ReadMe.txt“, „read.rtf“, „read.txt“ und „It’s All About Me.rtf“ vor. Hier können Sie z. B. wie folgt Platzhalter einsetzen:

- `*.rtf` findet „ReadMe.rtf“, „read.rtf“ und „It’s All About Me.rtf“
- `????.*` findet „read.rtf“ und „read.txt“
- `[Rr]*.rtf` findet „ReadMe.rtf“ und „read.rtf“
- `[A-Z].*` findet „ReadMe.rtf“, „ReadMe.txt“ und „It’s All About Me.rtf“

Verwenden von rekursiven Befehlen

Wenn Sie in der Befehlszeile einen Befehl anwenden, um eine Aufgabe an einem Objekt auszuführen, wird nur das festgelegte Objekt beeinflusst. Handelt es sich dabei um einen Ordner, wird die Befehlszeile nicht automatisch in den Ordner wechseln, um den Befehl auch auf die darin enthaltenen Objekte anzuwenden. Soll ein Befehl auf einen Ordner und seinen Inhalt angewendet werden, müssen Sie den Befehl anweisen, sich rekursiv zu verhalten. „Rekursiv“ bedeutet, dass die Aufgabe von der von Ihnen festgelegten Position für jedes Objekt in jedem Ordner ausgeführt werden soll. Die meisten Befehle akzeptieren `-r` oder `-R` als Option, damit sich der Befehl rekursiv verhält.

Im folgenden Beispiel möchte Michelle den Inhalt ihres öffentlichen Ordners („Public“) einmal normal auflisten und anschließend den Befehl mit der Option `-R` rekursiv verwenden. Beachten Sie, dass das System beim rekursiven Auflisten des Ordnerinhalts von „Public“ (Öffentlich) auch der Ordnerinhalt von „Drop Box“ (Briefkasten) und „Drop Folder“ angezeigt wird:

```
MyMac:~ michelle$ ls Public
Drop Box PublicFile1 PublicFile2 PublicFile3
MyMac:~ michelle$ ls -R Public
Drop Box PublicFile1 PublicFile2 PublicFile3

Public/Drop Box:
Drop Folder DroppedFile1 DroppedFile2

Public/Drop Box/Drop Folder:
DropFolderFile1 DropFolderFile2
```

Bearbeiten von Dateien und Ordnern

Es gibt folgende grundlegende Befehle zum Bearbeiten von Dateien und Ordnern: `mkdir`, `cp`, `mv`, `rm`, `rmdir` und `vi`.

mkdir

Kurzform von „make directory“. Dieser Befehl wird zum Erstellen von neuen Ordnern verwendet. Die Syntax ist `mkdir`, gefolgt von den Pfaden der neuen Ordner, die Sie erstellen möchten. Eine häufig verwendete Option ist `-p`, die `mkdir` anweist, hierarchische Ordnerstrukturen zu erstellen, die in den von Ihnen festgelegten Pfaden noch nicht bestehen. Im folgenden Beispiel verwendet Michelle den Befehl `mkdir` zusammen mit der Option `–p`, um einen Ordner namens „Private“ zusammen mit den beiden Unterordnern „Stocks“ und „Bonds“ zu erstellen:

```
MyMac:~ michelle$ ls
Desktop Downloads Movies Pictures Sites
Documents Library Music Public
MyMac:~ michelle$ mkdir -p Private/Stocks Private/Bonds
MyMac:~ michelle$ ls
Desktop Downloads Movies Pictures Public
Documents Library Music Private Sites
MyMac:~ michelle$ cd Private/
MyMac:Private michelle$ ls
Bonds Stocks
```

TIPP Mit dem Befehl `mkdir` können Sie schnell temporäre Ordner für das Testen in der Befehlszeile erstellen. Mit `touch`, gefolgt von einem Dateinamen, können Sie schnell temporäre Dateien für das Testen in der Befehlszeile erstellen. Die ursprüngliche Funktion des Befehls `touch` liegt zwar in der Aktualisierung des Änderungsdatums eines bestimmten Objekts, aber damit lässt sich auch eine leere Datei erstellen, falls noch keine existiert.

cp

Kurzform von „copy". Dieser Befehl kopiert Objekte von einer Position zu einer anderen. Die Syntax lautet `cp`, gefolgt vom Pfad des ursprünglichen Objekts und dem Zielpfad der Kopie. Im folgenden Beispiel verwendet Michelle den Befehl `cp`, um eine Kopie der Datei „testfile" zu erstellen. Diese befindet sich in ihrem Benutzerordner. Michelle platziert die Kopie (testfile2) in ihrem Ordner „Desktop" (Schreibtisch).

HINWEIS ▶ Denken Sie beim Kopieren eines Ordners und seines gesamten Inhalts daran, dass Sie den Befehl `cp` durch Hinzufügen der Option `-R` anweisen müssen, sich rekursiv zu verhalten.

```
MyMac:~ michelle$ ls
Desktop Library Pictures testfile
Documents Movies Public
Downloads Music Sites
MyMac:~ michelle$ cp testfile Desktop/testfile2
MyMac:~ michelle$ ls Desktop/
testfile2
```

Wenn Sie mit dem Befehl `cp` arbeiten und einen Zielordner, aber keinen Dateinamen angeben, wird die Kopie unter dem gleichen Namen wie das Original gesichert. Wenn Sie hingegen einen Zieldateinamen, aber keinen Zielordner angeben, wird eine Kopie in Ihrem aktuellen Arbeitsverzeichnis erstellt. Außerdem werden Sie im Gegensatz zum Kopieren im Finder bei Verwendung des Befehls `cp` nicht gewarnt, wenn Ihre Kopie eine vorhandene Datei ersetzt. Die vorhandene Datei wird einfach gelöscht und durch die Kopie ersetzt, die erstellt werden soll. Dieses Verhalten trifft auf die meisten Befehle zu.

TIPP Mit `scp`, dem Befehl zum sicheren Kopieren, können Sie Dateien zwischen Macs, die mit dem Netzwerk verbunden sind, per Fernanmeldung über SSH kopieren. Die Aktivierung der Fernanmeldung über SSH wird in Lektion 8, „Netzwerkdienste", beschrieben.

mv

Kurzform von „move“. Dieser Befehl verschiebt Objekte von einer Position zu einer anderen. Die Syntax ist mv, gefolgt vom Pfad des ursprünglichen Objekts und abschließender Angabe des Zielpfads. Im folgenden Beispiel verwendet Michelle den Befehl mv, um die Datei „testfile2“ von ihrem Schreibtisch-Ordner („Desktop“) in ihren Benutzerordner zu verschieben:

```
MyMac:~ michelle$ ls Desktop/
testfile2
MyMac:~ michelle$ ls
Desktop Library Pictures testfile
Documents Movies Public
Downloads Music Sites
MyMac:~ michelle$ mv Desktop/testfile2 testfile2
MyMac:~ michelle$ ls
Desktop Library Pictures testfile
Documents Movies Public testfile2
Downloads Music Sites
```

Der Befehl mv verwendet dieselben Regeln für den Zielort wie der Befehl cp. Da der Zieldateiname ohne Ordner angegeben wurde, wird die Datei in das aktuelle Arbeitsverzeichnis verschoben. Der Befehl mv ist auch ein Umbenennungsbefehl. Im Grunde ist das Verschieben eines Objekts in denselben Ordner unter Verwendung eines anderen Namens das Gleiche wie eine Umbenennung. Im folgenden Beispiel arbeitet Michelle in ihrem Benutzerordner und verwendet den Befehl mv, um die Datei „testfile“ in „testfile1“ umzubenennen:

```
MyMac:~ michelle$ ls
Desktop Library Pictures testfile
Documents Movies Public testfile2
Downloads Music Sites
MyMac:~ michelle$ mv testfile testfile1
MyMac:~ michelle$ ls
Desktop Library Pictures testfile1
Documents Movies Public testfile2
Downloads Music Sites
```

rm

Kurzform von „remove“. Dieser Befehl löscht Objekte endgültig. Es gibt in der Befehlszeile keinen Papierkorb. Deshalb ist der Befehl `rm` endgültig. Die Syntax ist `rm`, gefolgt von den Pfaden zu den Objekten, die gelöscht werden sollen. Im folgenden Beispiel verwendet Michelle den Befehl `rm`, um die Dateien „testfile1“ und „testfile2“ zu löschen.

> **HINWEIS ▶** Denken Sie beim Löschen eines Ordners und seines gesamten Inhalts daran, dass Sie den Befehl `rm` durch Hinzufügen der Option `-R` anweisen müssen, sich rekursiv zu verhalten.

```
MyMac:~ michelle$ ls
Desktop Library Pictures testfile1
Documents Movies Public testfile2
Downloads Music Sites
MyMac:~ michelle$ rm testfile1 testfile2
MyMac:~ michelle$ ls
Desktop Downloads Movies Pictures Sites
Documents Library Music Public
```

> **TIPP** Objekte, die mit dem Befehl `rm` gelöscht wurden, können bis zu einem gewissen Grad mit Programmen zur Datenwiederherstellung wiederhergestellt werden. Zum sicheren Löschen eines Objekts eignet sich daher der Befehl `srm`. Weitere Informationen zum sicheren Löschen finden Sie in Lektion 4, „Dateisysteme“.

rmdir und rm -R

Kurzform von „remove directory“. Dieser Befehl löscht Ordner endgültig. Wie bereits erwähnt, gibt es in der Befehlszeile keinen Papierkorb. Deshalb ist der Befehl `rmdir` endgültig. Die Syntax ist `rmdir`, gefolgt von den Pfaden der Ordner, die Sie löschen möchten. Der Befehl `rmdir` kann keine Ordner löschen, die Objekte enthalten, weshalb er oftmals überflüssig ist, da Sie Ordner und ihren Inhalt einfach mit dem Befehl `rm` und der rekursiven Option löschen können.

Im folgenden Beispiel versucht Michelle den Ordner „Private“ mit dem Befehl `rmdir` zu löschen. Dies funktioniert aber nicht, da der Ordner Objekte enthält. Daher versucht sie es mit dem Befehl `rm`, womit sie aber aus demselben Grund scheitert. Abschließend verwendet sie den Befehl `rm` zusammen mit der rekursiven Option `-R`, um den Ordner und seinen Inhalt zu löschen.

```
MyMac:~ michelle$ rmdir Private/
rmdir: Private/: Directory not empty
MyMac:~ michelle$ rm Private/
```

```
rm: Private/: is a directory
MyMac:~ michelle$ rm -R Private/
MyMac:~ michelle$ ls
Desktop Downloads Movies Pictures Sites
Documents Library Music Public
```

vi

Kurzform von „visual". Für viele hat dieser Befehl den paradoxesten Namen, da es sich hierbei um einen Texteditor handelt, der so gut wie keine visuellen Elemente bietet. `vi` ist jedoch der gängige Texteditor in der Befehlszeile. Um ein Textdokument zur Bearbeitung mit diesem Befehl zu öffnen, geben Sie einfach `vi` ein, gefolgt vom Pfad oder Namen einer Textdatei.

> **TIPP** Mac OS X leitet automatisch von `vi` zur neueren verbesserten Variante, `vim`, um. Wenn Sie den Befehl nur für grundlegende Funktionen verwenden, werden Sie wahrscheinlich keinen Unterschied feststellen.

> **TIPP** Mac OS X stellt `nano` bereit, einen moderneren und in vielerlei Hinsicht bedienungsfreundlicheren Texteditor. Er enthält unten im Fenster z. B. einen „Spickzettel" mit häufig verwendeten Befehlen. `vi` ist jedoch in einigen Fällen der Standardeditor, z. B. beim Bearbeiten bestimmter Systemdateien. Deshalb ist es erforderlich, dass Sie zumindest die grundlegenden Techniken von `vi` kennenlernen.

Ähnlich wie der Befehl `less` nimmt `vi` für den Inhalt der Textdatei das gesamte Terminal-Fenster in Anspruch. Wenn `vi` zum ersten Mal geöffnet wird, befindet sich der Editor im Befehlsmodus. Im Befehlsmodus wartet `vi` auf die Eingabe von vordefinierten Zeichen, die `vi` mitteilen, welchen Vorgang Sie als nächstes abschließen möchten. Sie können die Datei auch im Befehlsmodus mithilfe der Pfeiltasten durchsuchen. Geben Sie nach `vi` einfach den Buchstaben „a" ein, um den Text zu bearbeiten.

In diesem Modus fügt `vi` an der Stelle, an sich der Cursor befindet, neuen Text in das Dokument ein. Den Cursor können Sie mithilfe der Pfeiltasten bewegen. Wenn Sie mit der Textbearbeitung fertig sind, müssen Sie die Änderungen sichern. Hierfür müssen Sie zunächst in den Befehlsmodus von `vi` zurückkehren. Durch Drücken der Taste Esc gelangen Sie jederzeit in den Befehlsmodus von `vi`. Im Befehlsmodus können Sie dann gleichzeitig Änderungen sichern und `vi` beenden, indem Sie „ZZ" eingeben.

Zusammenfassend benötigen Sie eigentlich nur drei Tastaturkurzbefehle, um mit dem Texteditor `vi` arbeiten zu können: „a", um Text eingeben zu können, Esc, um in den Befehlsmodus zurückzukehren, und „ZZ", um Ihre Änderungen zu sichern und den Editor zu beenden (denken Sie an „A bis Z"). Es gibt noch einen weiteren `vi`-Befehl, den Sie für den Fall kennen sollten, dass Sie bei der Textbearbeitung schwerwiegende Fehler machen. Im Befehlsmodus können Sie `vi` ohne Sichern von Änderungen beenden, indem Sie „:quit!" eingeben.

Befehlszeilenverwaltung

Die vielleicht leistungsfähigste Funktion in der Befehlszeile ist die Möglichkeit des schnellen Wechsels zu einem anderen Benutzeraccount oder sogar zum root-Account. In diesem Abschnitt beschäftigen Sie sich mit einigen Befehlen, die für Administratoren sehr nützlich sind, da Sie damit auf Objekte zugreifen können, die normalerweise durch Dateisystemzugriffsrechte geschützt sind.

> **WEITERE INFORMATIONEN ▶** Ausführliche Informationen zu den Zugriffsrechten im Dateisystem finden Sie in Lektion 4, „Dateisysteme".

Verwenden von „su"

Der Befehl `su` ist die Kurzform von „substitute user identity". Er ermöglicht es Ihnen, in der Befehlszeile einfach zu einem anderen Benutzeraccount zu wechseln. Geben Sie dazu `su` und den Kurznamen des Benutzers ein, zu dessen Account Sie wechseln möchten. Geben Sie anschließend das Kennwort des Accounts ein (das Kennwort wird in Terminal nicht angezeigt). Die Eingabeaufforderung ändert sich, was darauf hinweist, dass Sie nun die Zugriffsrechte eines anderen Benutzers haben. Sie können Ihren derzeitigen Account bestimmen, indem Sie `who -m` in der Befehlszeile eingeben. Sie bleiben so lange als Ersatzbenutzer angemeldet, bis Sie Terminal beenden oder den Befehl `exit` eingeben. Im folgenden Beispiel verwendet Michelle den Befehl `su`, um von ihrer Shell zu Kevins Account zu wechseln. Anschließend kehrt sie zu ihrem eigenen Account zurück:

```
MyMac:~ michelle$ who -m
michelle ttys001 Aug 20 14:06
MyMac:~ michelle$ su kevin
Password:
bash-3.2$ who -m
kevin ttys001 Aug 20 14:06
bash-3.2$ exit
exit
MyMac:~ michelle$ who -m
michelle ttys001 Aug 20 14:06
```

Verwenden von „sudo"

Ein noch leistungsfähigerer Befehl ist `sudo`. Dies ist die Kurzform von „substitute user do" bzw. „super user do". Die Einleitung eines Befehls mit `sudo` weist den Computer an, den Befehl unter root-Berechtigung auszuführen. Die einzige Voraussetzung für die Verwendung von `sudo` in Mac OS X ist, dass der Benutzer ein Administrator ist und sich nach Eingabe des Befehls authentifiziert (auch hier wird das Kennwort nicht in Terminal angezeigt).

Das heißt, in Mac OS X kann jeder Administrator den Befehl sudo verwenden, um root-Zugriff in der Befehlszeile zu erhalten. sudo funktioniert auch, wenn kein root-Benutzer in der grafischen Benutzeroberfläche aktiviert wurde. Dieser Zugriff ist einer der Hauptgründe, warum es in vielen Umgebungen zu gefährlich ist, jedem Benutzer administrativen Zugriff zu gewähren. Sie können die Konfigurationsdatei von sudo jedoch so anpassen, dass die Verwendung des Befehls weiter eingeschränkt wird, wie weiter hinten in diesem Abschnitt beschrieben.

> **TIPP** Der Befehl sudo kann auch genutzt werden, um einen Befehl als bestimmter Nicht-root-Benutzer auszuführen. Geben Sie vor einem Befehl `sudo -u Benutzername` ein, wobei Benutzername der Kurzname des Benutzers ist, als der Sie den Befehl ausführen möchten.

Im folgenden Beispiel ist es Michelle normalerweise nicht gestattet, die Textdatei „Secrets" mit dem standardmäßigen Textausgabebefehl cat einzusehen. Daher verwendet sie den Befehl sudo, um root-Zugriff für den Befehl cat zu erhalten, sodass sie den Inhalt von „Secrets" lesen kann:

```
MyMac:~ michelle$ cat Secrets.txt
cat: Secrets.txt: Permission denied
MyMac:~ michelle$ sudo cat Secrets.txt
Password:
Dies ist der Inhalt der Textdatei „Secrets.txt", auf die der Benutzeraccount
von Michelle normalerweise keinen Lesezugriff hat. Da sie jedoch eine Ad-
ministratorin ist, kann sie den Befehl „sudo" verwenden, um root-Zugriff zu
erhalten und somit den Inhalt der Textdatei zu lesen.
```

> **TIPP** Wenn die Befehlszeile eine Fehlermeldung ausgibt, da Sie vergessen haben, sudo zu verwenden, geben Sie nach der nächsten Eingabeaufforderung einfach `sudo !!` ein, um den vorherigen Befehl mit vorangestelltem sudo auszuführen.

Denken Sie daran, dass mit einem erweiterten Zugriff auch mehr Verantwortung verbunden ist. Die Verwendung von sudo mit einem falsch geschriebenen Befehl kann ein Chaos in Ihrem Betriebssystem verursachen. Die Befehlszeile weist Sie nur beim ersten Versuch, sudo zu verwenden, darauf hin, dass Sie ernsthaften Schaden anrichten können. Danach setzt die Befehlszeile voraus, dass Sie wissen, was Sie tun. Wenn Sie sich drei Regeln im Umgang mit der Befehlszeile merken sollten, dann sind es die folgenden: Überprüfen Sie Ihre Eingaben sehr genau, verwenden Sie immer die automatische Vervollständigung, um Tippfehler zu vermeiden und Zeit zu sparen, und überprüfen Sie Ihre Eingaben bei der Verwendung von sudo am besten drei Mal.

Wechseln der Shell mit „sudo"

Wenn Sie als Administrator mehr als einen Befehl mit root-Zugriff ausführen müssen, können Sie die gesamte Befehlszeilen-Shell vorübergehend auf root-Zugriffsebene umschalten.

Geben Sie hierfür einfach `sudo -s` und Ihr Kennwort ein. Dadurch schaltet die Shell auf root-Zugriff um. Sie können Ihren derzeitigen Account bestimmen, indem Sie `who -m` in der Befehlszeile eingeben. Sie bleiben so lange als root-Benutzer angemeldet, bis Sie Terminal beenden oder den Befehl `exit` eingeben. Im folgenden Beispiel verwendet Michelle den Befehl `sudo`, um ihre Shell auf den root-Benutzer umzuschalten. Anschließend kehrt sie zu ihrem eigenen Account zurück:

```
MyMac:~ michelle$ who -m
michelle ttys001 Aug 20 14:31
MyMac:~ michelle$ sudo -s
Password:
bash-3.2# who -m
root ttys001 Aug 20 14:31
bash-3.2# exit
exit
MyMac:~ michelle$ who -m
michelle ttys001 Aug 20 14:31
```

Verwalten des sudo-Zugriffs

Wie bereits erwähnt, kann in Mac OS X jeder Administrator standardmäßig den Befehl `sudo` verwenden, um als Systemadministrator (root) Zugriff auf Ressourcen zu erhalten. Nachdem Sie sich für den Befehl `sudo` authentifiziert haben, bleibt dieser fünf Minuten lang aktiv, sodass Sie sich während dieser Zeitspanne bei erneuter Ausführung eines `sudo`-Befehls nicht erneut authentifiziert müssen. Durch Eingabe von `sudo -s` erfolgt der Wechsel zum Administrator mit root-Zugriff dauerhaft.

Aus diesen Gründen sollten Sie in Erwägung ziehen, den `sudo`-Zugriff einzuschränken. Sie können natürlich nur Standard-Benutzeraccounts in Ihrem Mac-System erlauben. Dies würde allerdings mehr als nur den `sudo`-Zugriff einschränken. Aber, wie in Lektion 2, „Benutzeraccounts“, beschrieben, ist dies der sicherste Benutzeraccount für den allgemeinen Gebrauch.

TIPP Jede Verwendung des sudo-Befehls wird im Systemprotokoll „system.log“ dokumentiert. Dort können Sie überprüfen, ob ein Benutzer seine Administratorrechte unrechtmäßig ausübt. Das Systemprotokoll „system.log“ finden Sie unter „/Programme/Dienstprogramme/Konsole“.

Alternativ können Sie die sudo-Konfigurationsdatei „/etc/sudoers“ verwalten. Diese Datei enthält die Regeln, nach denen der Befehl sudo die zulässigen Aktionen bestimmt. Als Administrator können Sie diese Konfigurationsdatei lesen, indem Sie den Befehl cat oder less verwenden. Im folgenden Beispiel verwendet Michelle den Befehl cat, um auf die sudo-Konfigurationsdatei zuzugreifen. Beachten Sie, dass sie den cat-Befehl mit sudo beginnen muss, da die Datei „sudoers“ root-Zugriff erfordert. Im folgenden Beispiel wurde die Ausgabe des Befehls less gekürzt, um nur die interessantesten Teile der sudoers-Datei anzuzeigen.

```
MyMac:~ michelle$ sudo cat /etc/sudoers
Password:
# sudoers file.
#
# This file MUST be edited with the 'visudo' command as root.
# Failure to use 'visudo' may result in syntax or file permission errors
# that prevent sudo from running.
#
# See the sudoers man page for the details on how to write a sudoers file.
#
...
# User privilege specification
root        ALL=(ALL) ALL
%admin      ALL=(ALL) ALL
...
```

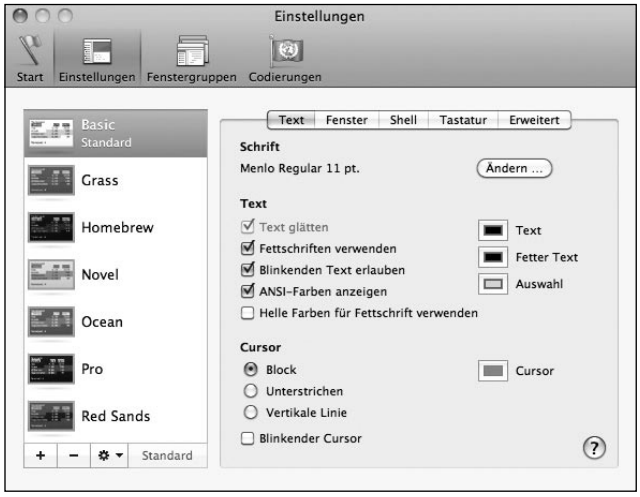
Wie im Abschnitt „User privilege specification“ dieser Datei ersichtlich ist, hat der root-Benutzer oder jeder Benutzer der Gruppe „admin“ uneingeschränkten sudo-Zugriff auf alle Befehle. Sie können diese Datei bearbeiten, Sie müssen jedoch die in der Datei erwähnte spezielle Version des vi-Befehls verwenden, nämlich visudo. Die Verwendung von vi zum Bearbeiten von Textdateien ist weiter vorne in dieser Lektion beschrieben.

Wenn Sie mit der Verwendung von vi vertraut sind, ist die Bearbeitung der sudoers-Datei mit dem Befehl visudo relativ einfach. Um den sudo-Zugriff für Administratoren zu deaktivieren, fügen Sie einfach das Nummernzeichen (#) am Anfang der Zeile „%admin“ ein. Daraufhin ignoriert der Befehl sudo diese Zeile. Sie können zusätzliche Benutzer oder Gruppen für den sudo-Zugriff hinzufügen, indem Sie die bestehenden Zeilen für die Benutzerzugriffsrechte mit alternativen Accountnamen duplizieren. Sie müssen nur daran denken, den „Kurznamen“ des Accounts zu verwenden sowie das Prozentzeichen (%), um Gruppennamen festzulegen.

Tipps und Tricks für die Befehlszeileneingabe

Folgende Tipps zur Befehlszeile sollen Ihnen helfen, die Oberfläche an Ihre Bedürfnisse anzupassen und sehr viel Zeit bei der Befehlszeileneingabe zu sparen:

- Der beste Tipp für die Befehlszeile ist, bei der Pfadeingabe immer die automatische Vervollständigung zu verwenden. Die automatische Vervollständigung wurde bereits weiter vorne in dieser Lektion erklärt.
- Bewegen Sie Dateien und Ordner per Drag&Drop vom Finder in das Terminal-Fenster, um ihre Pfadangaben automatisch in die Befehlszeile einzufügen.
- Geben Sie `open .` nach der Eingabeaufforderung ein, um die derzeitige Position im Dateisystem als Ordner im Finder zu öffnen.
- Sehen Sie sich die Einstellungen des Programms „Terminal“ genau an (wählen Sie *Terminal* > *Einstellungen* in der Menüleiste), um das Erscheinungsbild der Befehlszeile anzupassen.



- Um einen fehlerhaften Befehl abzubrechen oder den Befehlseintrag zu löschen, verwenden Sie den Tastaturkurzbefehl Ctrl + C.
- Sie können Befehle bearbeiten, bevor Sie sie ausführen. Die ← - und → -Tasten sowie die Taste Entf funktionieren wie erwartet, die Maus allerdings nicht.
- Verwenden Sie die Aufwärtspfeil- und Abwärtspfeiltasten, um durch den Befehlsverlauf zu navigieren und Befehle wiederzuverwenden. Dies schließt auch die Möglichkeit ein, alte Befehle vor erneuter Verwendung zu bearbeiten. Durch Eingabe des Befehls `history` können Sie Ihren Befehlsverlauf anzeigen.

- Um die Einträge im Terminal-Fenster zu löschen, geben Sie den Befehl zum Löschen ein oder verwenden Sie Ctrl+L.
- Um den Cursor an den Anfang der aktuellen Zeile zu bewegen, verwenden Sie Ctrl+A.
- Um den Cursor an das Ende der aktuellen Zeile zu bewegen, verwenden Sie Ctrl+E.
- Um den Cursor um ein Wort nach vorne zu bewegen, verwenden Sie Esc+F.
- Um den Cursor um ein Wort zurück zu bewegen, verwenden Sie Esc+B.

Verwenden von Automator und AppleScript

Mac OS X hält zwei primäre Technologien für die Automatisierung von Aufgaben in der grafischen Benutzeroberfläche bereit: Automator und AppleScript. Mit Automator im Ordner „/Programme“ können Sie aus Aktionen Arbeitsabläufe erstellen, um Routineaufgaben zu automatisieren. Die Technologie von Automator basiert eigentlich auf der älteren Technologie von AppleScript. AppleScript ist eine leistungsstarke Skriptsprache, die der englischen Sprache ähnelt, erfordert aber wie jede andere Programmiersprache auch bestimmte Vorkenntnisse.

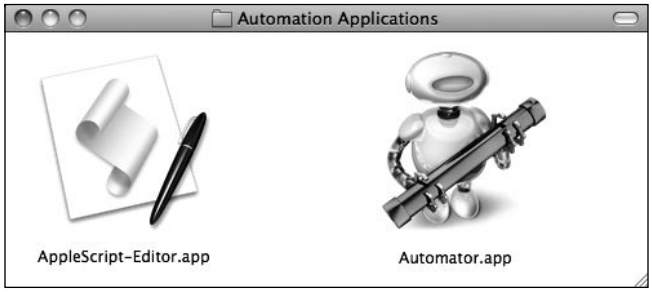
Die Beherrschung von Automator- und AppleScript-Techniken geht weit über den Rahmen dieses Handbuchs hinaus. Ziel dieses Handbuchs ist es, zu verdeutlichen, was diese Automatisierungswerkzeuge leisten können, und Ihnen ausreichend Informationen an die Hand zu geben, dass Sie mit der Entwicklung eigener Automatisierungslösungen beginnen können. Wie bei vielen gängigen Entwicklungstechnologien gibt es eine umfangreiche Bibliothek mit Automator- und AppleScript-Beispielen, die Sie als Ausgangsbasis verwenden können. Häufig lernen Sie aus diesen Beispielen, wie Sie Ihre Automatisierungsziele am schnellsten erreichen können.

Automatisierungsergebnisse in Mac OS X

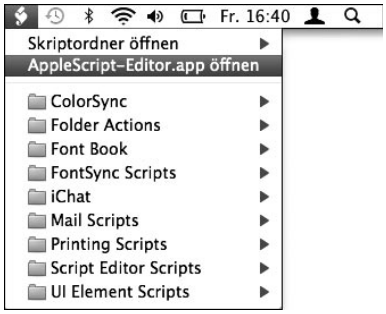
Am besten lernen Sie Automator und AppleScript kennen, wenn Sie sich ansehen, wie ein Benutzer von den von Ihnen erzeugten Automatisierungswerkzeugen profitiert. Mac OS X bietet eine Reihe von Methoden, um auf automatisierte Lösungen zuzugreifen oder sie zu initiieren.

Mit Automator oder AppleScript lässt sich Folgendes erzeugen:

- Programme – Auf diese Objekte können Sie wie auf normale Programme zugreifen, d. h. der Benutzer kann sie im Finder oder Dock öffnen.

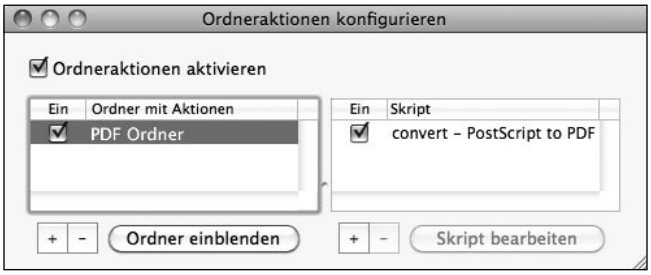


- Objekte für das Skriptmenü – Automatisierungsobjekte lassen sich schnell über das Skriptmenü öffnen, das Sie in den Einstellungen des AppleScript-Editors unter „/Programme/Dienstprogramme/AppleScript-Editor“ aktivieren können, sodass es in der Menüleiste angezeigt wird. Dieses Menü setzt sich aus dem Inhalt der Ordner „/Library/Scripts“ und „~/Library/Scripts“ zusammen. Über das Skriptmenü können Sie schnell auf geöffnete Arbeitsablaufdateien in Automator, AppleScript-Dateien und automatisierte Programme zugreifen.



- Applets – Diese Objekte ähneln Programmen. Der einzige Unterschied ist, dass der Benutzer nicht auf das Symbol doppelklickt, sondern per Drag&Drop Dateien und Ordner auf das Applet-Symbol im Finder oder Dock bewegt. Die Automatisierung erfolgt für diese Objekte und das Applet wird nach Beendigung einfach geschlossen.
- Ordneraktionen im Finder – Diese Aktionen werden an Ordner angeheftet, die der Finder auf neuen Inhalt prüft. Wenn in einen Ordner mit einer Ordneraktion Objekte abgelegt werden, sendet das System diese Objekte automatisch an das Applet. Sie können Ordneraktionen im Finder konfigurieren, indem Sie mit der rechten Maustaste auf einen Ordner klicken und dann aus dem Kontextmenü die Option „Ordneraktionen konfigurieren“ auswählen. Dadurch wird das Programm in „/System/Library/CoreSer-

vices/Ordneraktionen konfigurieren“ geöffnet, in dem Sie einem vom Finder zu überwachenden Ordner ein Skript zuweisen können.



HINWEIS ▶ Plug-Ins – Diese Objekte sind so konzipiert, dass sie Eingaben von bestimmten Programmen empfangen. Automator stellt Vorlagen zum Erstellen von Plug-Ins für Druckdialoge und Scanner für „Digitale Bilder“ bereit. Es können auch Plug-Ins für andere Programme verwendet werden, soweit diese von dem Programm unterstützt werden.

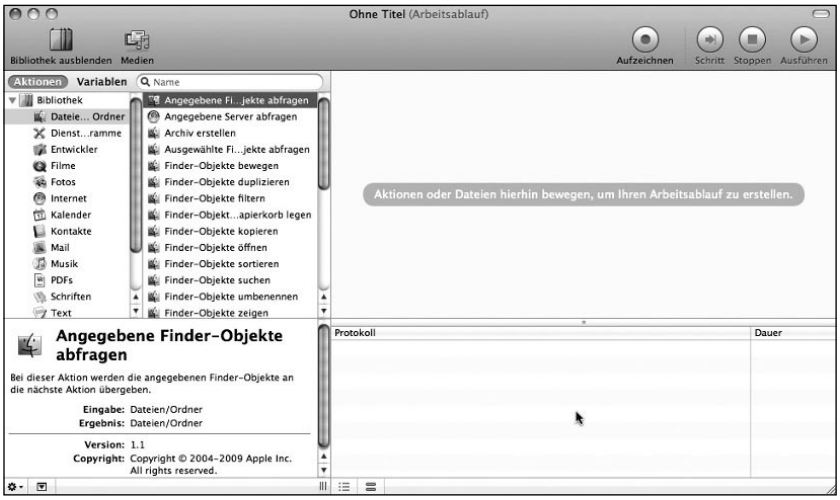
- Dienste – Dienste ermöglichen es Ihnen, aus einem Programm heraus auf Funktionen eines anderen Programms zuzugreifen. Im Rahmen der Automation können Sie Automatisierungsdienste erstellen, auf die Sie aus Menüs von unterstützten Programmen zugreifen können. Die Verwendung von Diensten wird in dieser Lektion im Abschnitt „Kombinieren von Automatisierungstechniken“ genauer beschrieben.

Einführung in Automator

Mit Automator können sogar unerfahrene Benutzer ohne Hintergrundwissen über das Schreiben von Code von den leistungsstarken Automatisierungsfunktionen von AppleScript profitieren. Automator arbeitet mit Automator-Aktionen, die jeweils eine kleine grafische Benutzeroberfläche darstellen, mit der Sie eine spezifische automatisierte Aufgabe in einem bestimmten Programm ausführen können. Sie können diese Aktionen als Bausteine verwenden, indem Sie mehrere Automator-Aktionen in einer geordneten Liste kombinieren und so einen Automator-Arbeitsablauf erstellen, mit dem Sie eine Routineaufgabe ausführen können.


Automator.app

Um die Arbeit in Automator zu beginnen, öffnen Sie einfach das Programm. Es wird ein neuer Arbeitsablauf geöffnet. Sie können eine Vorlage auswählen oder auf *Arbeitsablauf* klicken, um eine leere Vorlage zu öffnen. Links sehen Sie die Bibliothek, in der alle verfügbaren Aktionen aufgeführt sind. Wenn Sie eine Aktion auswählen, wird die zugehörige Funktion unterhalb der Liste angezeigt. Automator ist eine erweiterbare Technologie, mit der jeder Benutzer zusätzliche Aktionen entwickeln kann. Zusätzliche Aktionen finden Sie unter „/Library/Actions/“, „~/Library/Actions/“ oder sogar innerhalb eines Programms.



Rechts im Automator-Fenster befindet sich der Bereich für den Arbeitsablauf, in den Sie die Aktionen zum Erstellen eines Arbeitsablaufs per Drag&Drop bewegen. Wenn sich die Aktion im Arbeitsablauf befindet, können Sie sie an Ihre Bedürfnisse anpassen. In vielen Fällen ist die Reihenfolge der Automator-Aktionen in einem Arbeitsablauf von Bedeutung, nicht nur, da dadurch die Reihenfolge der ausgeführten Aktionen festgelegt wird, sondern auch, weil aufeinander folgende Aktionen miteinander über Eingabe und Ausgabe kommunizieren können. Eine Automator-Aktion für den Finder kann z. B. ein Volume auf einem Image aktivieren, benötigt aber zuerst eine Eingabe von einer anderen Aktion, die das zu öffnende Image identifiziert. Die erste Aktion wählt die Image-Datei aus und gibt diese Informationen aus, sodass die Aktion, die das Image öffnet, weiß, welches Image zu aktivieren ist. Die folgende Abbildung zeigt ein einfaches Beispiel eines Automator-Arbeitsablaufs. Beachten Sie hierbei die Eingabe-Ausgabe-Verbindung zwischen den beiden Aktionen.

TIPP Mit der Automator-Aufzeichnungsfunktion, die Sie über die Taste *Aufzeichnen* in der Symbolleiste initiieren, können Sie Arbeitsabläufe basierend auf der Interaktion mit Ihrem Computer automatisch erstellen.

Um eine verlässliche Automatisierungslösung zu erstellen, ist häufig ein Testdurchlauf erforderlich. Mit den Tasten *Schritt, Stoppen* und *Ausführen* rechts in der Symbolleiste können Sie einen Automator-Arbeitsablauf ausführen. Der Status des Arbeitsablaufs (positiv und negativ) wird im Bereich *Protokoll* des Automator-Fensters angezeigt. Sobald Sie eine funktionsfähige Lösung erarbeitet haben, können Sie sie als Arbeitsablauf zur Archivierung oder späteren Bearbeitung sichern. Wenn Sie möchten, dass der Arbeitsablauf ohne Zutun von Automator ausgeführt wird, können Sie ihn auch als eigenständiges Programm oder als eines der Vorlagenformate sichern, die beim Öffnen von Automator angezeigt werden.

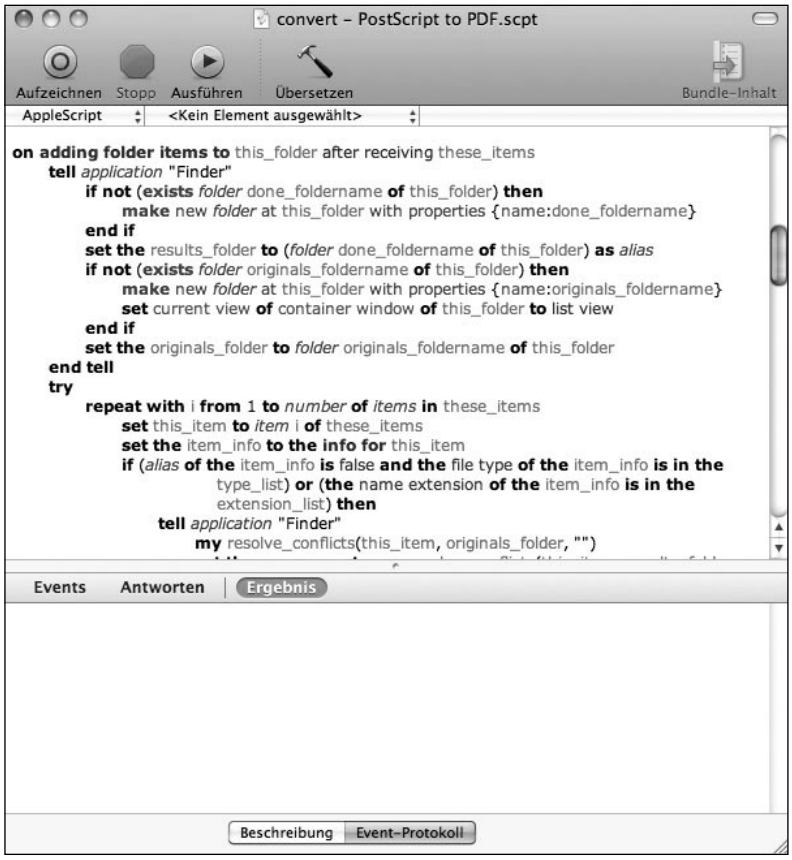
WEITERE INFORMATIONEN ▶ Automator stellt eine hervorragende integrierte Hilfe bereit, auf die Sie über das Menü *Hilfe* zugreifen können. Sie umfasst sowohl Dokumentation als auch Beispiele. Weitere Automator-Aktionen und Beispiele für Arbeitsabläufe sind auf folgenden Websites verfügbar: http://www.macosxautomation.com/ und http://macscripter.net.

Einführung in AppleScript

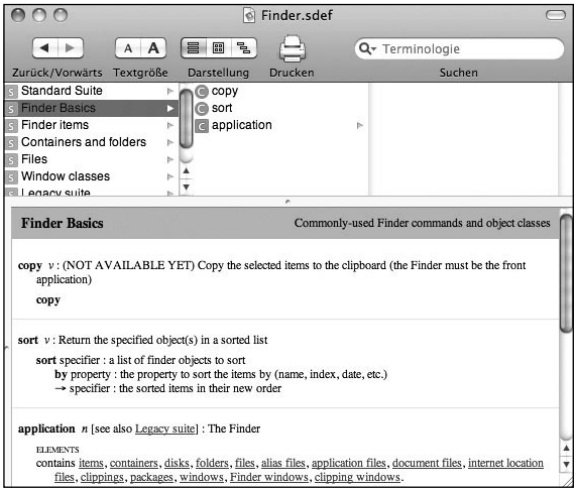
AppleScript bildet nicht nur die Basis für Automator, sondern kann auch alleine verwendet werden, um Aufgaben für verschiedene Programme zu automatisieren. AppleScript ist eine der englischen Sprache ähnelnde Skriptsprache, die ursprünglich für die Classic-Umgebung des Mac OS-Betriebssystems entwickelt wurde. Die primäre Schnittstelle zum Erstellen und Bearbeiten von AppleScript-Skripten ist der AppleScript-Editor unter „/Programme/Dienstprogramme/AppleScript-Editor“.

Sie können den AppleScript-Editor natürlich im Finder öffnen, allerdings wird dann ein leeres Dokument angezeigt. Am besten sehen Sie sich die Ordner unter „/Library/Scripts“

an und suchen ein Skript aus, das Sie interessiert. Passen Sie es dann an Ihre Bedürfnisse an. Im folgenden Beispiel wird ein in Mac OS X enthaltenes Skript angezeigt, das eine PostScript-Datei in eine PDF-Datei umwandelt.



Da es sich bei AppleScript um eine komplexe Skriptsprache handelt, sind bestimmtes Vorkenntnisse erforderlich. Wie bereits erwähnt, ist die Hilfe im AppleScript-Editor ein guter Ausgangspunkt, da Sie dort auf die ausführliche allgemeine AppleScript-Dokumentation zugreifen können. Eine weitere lokale Ressource ist das AppleScript-Funktionsverzeichnis, das in der Menüleiste über *Ablage* > *Funktionsverzeichnis öffnen* verfügbar ist. Es wird eine Liste mit AppleScript-Funktionsverzeichnissen geöffnet. Die meisten Programme, die AppleScript unterstützen, haben ein Funktionsverzeichnis, in dem die unterschiedlichen Funktionen, mit denen Sie das Programm steuern können, beschrieben sind. Wenn Sie das Funktionsverzeichnis für ein bestimmtes Programm auswählen, können Sie dort in den AppleScript-Funktionen, die für das entsprechende Programm verfügbar sind, suchen.



TIPP Mit der Aufzeichnungsfunktion im AppleScript-Editor, die Sie über die Taste *Aufzeichnen* in der Symbolleiste initiieren, können Sie Skripte basierend auf der Interaktion mit Ihrem Computer automatisch erstellen.

Auch hier ist häufig ein Testdurchlauf erforderlich, um eine verlässliche Automatisierungslösung zu erstellen. Sie können ein Skript mit den Tasten *Stopp* und *Ausführen* in der Symbolleiste ausführen. Der Status des Skripts (positiv und negativ) wird im Bereich *Event-Protokoll* unten im AppleScript-Editor-Fenster angezeigt. Sobald Sie eine funktionsfähige Lösung erarbeitet haben, können Sie sie als Skript zur Archivierung oder späteren Bearbeitung sichern. Wenn Sie möchten, dass das Skript ohne Zutun des AppleScript-Editors ausgeführt wird, können Sie es als eigenständiges Programm sichern.

WEITERE INFORMATIONEN ▶ Der AppleScript-Editor stellt eine hervorragende integrierte Hilfe bereit, auf die Sie über das Menü *Hilfe* zugreifen können. Sie umfasst sowohl Dokumentation als auch Beispiele. Weitere AppleScript-Beispiele sind auf folgenden Websites verfügbar: http://www.macosxautomation.com/ und http://macscripter.net.

Grundlagen des Befehlszeilen-Scripting

Wenn Sie einen Befehl in das Programm „Terminal“ eingeben können, ist er auch skriptfähig. Die gleiche „Sprache“, die Sie für die Interaktion mit der Befehlszeile verwenden, wird auch für das Befehlszeilen-Scripting, oder genauer gesagt das Shell-Scripting, verwendet. Schließlich wird die interaktive Befehlszeile über einen Shell-Prozess verwaltet. Genau dieser Shell-Prozess kann auch Befehle in einer Textdatei interpretieren. Somit ist ein Befehlszeilenskript, abgesehen von ein paar wenigen Formatierungsdetails, im Grunde eine Textdatei mit einer Liste von Befehlen.

So einfach das Befehlszeilen-Scripting auch klingt: Wenn Sie den vorherigen Abschnitt zu Automator und AppleScript lesen, fragen Sie sich vielleicht, warum Sie das Befehlszeilen-Scripting anstatt der Automatisierungstechnologien der grafischen Benutzeroberfläche verwenden sollten. Dafür gibt es mehrere Gründe:

- Schwerpunkt liegt auf der Systemverwaltung – Je mehr Sie sich mit Mac OS X und den verschiedenen Werkzeugen für die Befehlszeile beschäftigen, werden Sie feststellen, dass viele Verwaltungswerkzeuge der grafischen Benutzeroberfläche auf ihrem Äquivalent in der Befehlszeile beruhen. Genauso werden Sie feststellen, dass die meisten Verwaltungsaufgaben nicht ausreichend von Automator und AppleScript unterstützt werden (wenn überhaupt). Während Automator und AppleScript primär für die Automatisierung von Programmen der grafischen Benutzeroberfläche verwendet werden, wird das Befehlszeilen-Scripting hauptsächlich für die Automatisierung von Systemverwaltungsaufgaben eingesetzt.
- Zugriff als Systemadministrator (root) – Den Großteil der Automatisierung mithilfe der grafischen Benutzeroberfläche führen Sie als normaler Benutzer aus. Deshalb können dadurch die Systemressourcen nicht verändert werden. Auf der anderen Seite ist es ganz einfach, ein Befehlszeilenskript mit root-Zugriff auszuführen. Die gängigste Methode ist, das Skript mit dem vorangestellten Befehl `sudo` zu verwenden.
- Keine Benutzeroberfläche – Wenn Programme der grafischen Benutzeroberfläche über einen Automatisierungsvorgang gesteuert werden, werden „die Ergebnisse“ auch auf dem Bildschirm angezeigt. Dies ist bei Befehlszeilenskripten nicht der Fall, da sie im Wesentlichen im Hintergrund ablaufen.
- Höhere Leistung – Wenn Sie lediglich eine bestimmte Aufgabe erledigt haben möchten, erfordert die Aktualisierung der grafischen Oberfläche einen unnötigen Leistungsaufwand dar. Aufgaben, die ein hohes Maß an Wiederholung erfordern, können meistens schneller in der Befehlszeile erledigt werden, da dieser Leistungsaufwand normalerweise wegfällt.
- Mehr Entwicklungsoptionen – Das Befehlszeilen-Scripting ist nicht auf die Funktionen in der Terminal-Shell beschränkt. Neben den Shell-Skripten können Sie Skripte mithilfe einer Vielzahl von Entwicklungssprachen erstellen. Mac OS X bietet u. a. integrierte Unterstützung für Perl, Python, PHP, Tcl und Ruby.

Wie bereits erwähnt, geht die Beherrschung der Techniken für das Befehlszeilen-Scripting weit über den Rahmen dieses Handbuchs hinaus. Unser Ziel ist es, zu verdeutlichen, wie Sie von Befehlszeilenskripten profitieren können, und Ihnen ausreichend Informationen an die Hand zu geben, dass Sie mit der Arbeit an eigenen Skripten beginnen können. Wie bei vielen gängigen Entwicklungstechnologien gibt es eine umfangreiche Bibliothek mit Skriptbeispielen, die Sie als Ausgangsbasis verwenden können. Häufig lernen Sie aus diesen Beispielen, wie Sie die erforderlichen Aufgaben mithilfe des Befehlszeilen-Scripting am schnellsten realisieren können.

„Kleine Helfer“ für die Befehlszeile

Bevor Sie mit dem Befehlszeilen-Scripting beginnen, sollten Sie sich mit einigen Sonderzeichen und Befehlen vertraut machen, die Sie bei der Automatisierung mithilfe der Befehlszeile unterstützen. Dazu gehören `grep`, | (pipe) und > (redirect).

grep

Dieser Befehl, kurz für „Global Regular Expression Print“, sucht nach Mustern in Text (anhand regulärer Ausdrücke) und gibt nur Zeilen aus, die übereinstimmen. Dies ist nicht nur nützlich zum Filtern bestimmter Informationen in einer großen Datei, sondern auch zum Filtern der Ausgabe anderer Befehle, wie Sie weiter hinten im Abschnitt zu „pipe“ sehen werden. Um in einer Datei zu filtern, geben Sie `grep` ein, gefolgt von dem Suchbegriff und dem Dateipfad.

> **WEITERE INFORMATIONEN ▶** Der Befehl `grep` verwendet reguläre Ausdrücke als Filterkriterien, die den weiter vorne in dieser Lektion beschriebenen Platzhaltern ähneln. Weitere Informationen zu regulären Ausdrücken erhalten Sie, wenn Sie `man re_format` in der Befehlszeile eingeben.

Im folgenden Beispiel verwendet Michelle den Befehl `grep`, um in der Datei „/etc/services“ nach der Zeichenfolge „afp“ zu suchen. Die Datei enthält alle gängigen Netzwerkports und -dienste. Sie ist mehr als 14000 Zeilen lang. Der Befehl `grep` findet die beiden gesuchten Zeilen jedoch umgehend. Eine manuelle Suche wäre niemals so schnell.

```
MyMac:~ michelle$ grep "afp" /etc/services
afpovertcp      548/udp     # AFP over TCP
afpovertcp      548/tcp     # AFP over TCP
```

| (pipe)

Das Sonderzeichen „|“, das Sie durch Drücken von ⌥+7 (⇧+\ auf US-Tastaturen) eingeben, nennt sich „Pipe“. Wie der Name schon sagt, leitet der Befehl die Ausgabe eines Befehls als Eingabe an einen anderen Befehl weiter (ähnlich wie eine Pipeline). Dies kann bei der Kombination von Befehlsfunktionen von großem Nutzen sein. Der Befehl `system_profiler` z. B. ist das Befehlszeilenäquivalent zum Programm „System-Profiler“ und gibt die Informationen standardmäßig als reinen Text im Terminal-Fenster aus. Die Ausgabe dieses Befehls ist deshalb unübersichtlich und es ist schwierig, die gewünschten Informationen zu finden.

Eine Lösung hierfür ist, die Ausgabe von `system_profiler` per pipe-Befehl an den Textausgabebefehl `less` zu leiten, sodass Sie durch die Informationen blättern können. Hierfür müssten Sie `system_profiler | less` eingeben. Eine andere Möglichkeit wäre, mit dem Befehl `grep` die Ausgabe von `system_profiler` so zu filtern, dass nur die gesuchten Informationen ausgegeben werden. Im folgenden Beispiel macht Michelle genau das, indem Sie mit `grep` die Ausgabe von `system_profiler` nach der Systemversion filtert.

```
MyMac:~ michelle$ system_profiler | grep "System Version"
     System Version: Mac OS X 10.6 (10A432)
```

TIPP Sie können mehrere Pipes in einer Befehlszeile verwenden, um die Ausgabe von einem Befehl zum nächsten weiterzuleiten, und so eine Art Arbeitsablauf in der Befehlszeile erstellen.

> und >>

Das Sonderzeichen „>“ ist bekannt als Zeichen für „Größer als“, es wird jedoch manchmal auch als „Umleitung“ bezeichnet. In der Befehlszeile kann dieses Zeichen die Ausgabe eines Befehls an eine Textdatei umleiten. Wenn die Datei nicht existiert, wird eine neue Datei erstellt. Wenn sie existiert, wird sie ersetzt. Bei zwei Größer-Zeichen („>>“) werden die Daten an die vorhandene Datei angefügt.

Die Syntax für die Umleitung ist einfach der Befehl, gefolgt von dem Umleitungssymbol und dem Dateipfad. Im Fall von „system_profiler“ können Sie das Zeichen für die Umleitung verwenden, um die Ausgabe von „system_profiler“ als Textdatei zu sichern. Im folgenden Beispiel erstellt Michelle einen System-Profiler-Bericht auf ihrem Schreibtisch.

```
MyMac:~ michelle$ system_profiler > Desktop/SystemReport.txt
```

Erstellen eines einfachen Skripts

Wie bereits erwähnt, handelt es sich bei einem Befehlszeilenskript lediglich um eine Textdatei mit entsprechender Syntax. Sie können Ihre Skripte mit jedem beliebigen Texteditor erstellen, einschließlich „TextEdit“, „vi“ und „nano“. Es sind jedoch nicht alle Texteditoren gleich aufgebaut. Einige sind „skriptfreundlich“, da sie über spezielle Funktionen verfügen, die Sie beim Entwickeln von Skripten unterstützen. Eine häufig verwendete Skriptfunktion ist die Verwendung von unterschiedlichen Farben für bestimmte Wörter, basierend auf deren Bedeutung im Skript. Das Programm „Xcode“, das als optionales Installationspaket der Xcode Developer Tools auf der Mac OS X-Installations-DVD enthalten ist, bietet viele Funktionen, die beim Erstellen von Skripten hilfreich sind.

HINWEIS ▶ Befehlszeilenskripte müssen als reiner Text vorliegen, nicht als RTF- oder andere formatierte Textdatei.

Was den Skriptnamen angeht, können Sie im Grunde jeden beliebigen Namen ohne Suffix wählen. Schließlich bestehen die meisten Befehle nur aus dem Namen und es wird kein Dateityp benötigt. Es ist dennoch empfehlenswert, das Suffix „.sh“, kurz für „shell script“, an den Namen des Befehls anzuhängen. Falls Sie andere Skriptsprachen verwenden, können Sie die anderen Dateitypen anhand des Dateisuffixes erkennen.

In Mac OS X haben Sie auch die Möglichkeit, das Dateisuffix „.command“ zu verwenden. Wenn ein Benutzer in diesem Fall Ihr Skript im Finder durch Doppelklicken öffnet, wird es automatisch in Terminal ausgeführt. So kann ein Benutzer ganz einfach Ihr Skript ausführen, ohne wissen zu müssen, wie eine Aktion in der Befehlszeile gestartet wird.

WEITERE INFORMATIONEN ▶ Ein empfehlenswertes Drittanbieter-Programm (Freeware) für das Scripting ist TextWrangler von BareBones Software (www.barebones.com).

Skriptinhalt

Wenn Sie sich für einen Texteditor entschieden und eine neue entsprechend benannte Textdatei angelegt haben, können Sie ein Skript erstellen. Ein einfaches Befehlszeilenskript enthält nur ein paar wenige Objekte:

- #!/bin/bash – Dies ist die erste Zeile Ihres Skripts. Sie teilt dem Befehl mit, dass es sich bei dieser Textdatei um ein Skript handelt. Genau genommen wird der Befehlszeile gesagt, dass sie den Shell-Befehl „bash“ verwenden soll, um die Textdatei zu interpretieren und die Anweisungen auszuführen. Wenn das Skript eine andere Skriptsprache verwendet, muss in der ersten Zeile die Sprache angegeben werden.
- Kommentare – Jede Zeile, die mit einem Nummernzeichen (#) beginnt, wird in den meisten Skriptsprachen ignoriert. Nach dem Nummernzeichen können Sie Kommentare zu Ihrem Skript hinzufügen. Kommentare sollen Ihnen und anderen Benutzern helfen, den Inhalt des Skripts zu verstehen, ohne das gesamte Skript lesen zu müssen. Sie sollten sich angewöhnen, Kommentare in Ihre Skripte zu schreiben, damit Sie auch nach Monaten noch wissen, warum Sie an einer bestimmten Stelle einen bestimmten Code verwendet haben.
- Befehle – Dies ist der eigentliche Inhalt Ihres Skripts. In einem `bash`-Shell-Skript geben Sie einfach die Befehle wie in der Befehlszeile ein. Das Drücken des Zeilenschalters in der Textdatei ist das Gleiche wie ein Drücken des Zeilenschalters in der Befehlszeile. Geben Sie jeden Befehl in eine separate Zeile ein. Die Reihenfolge der Befehle in Ihrem Skript gibt vor, in welcher Reihenfolge sie ausgeführt werden.

HINWEIS ▶ Sie sollten in Skripten immer absolute Pfade verwenden, damit Ihre Skriptdateien portierbar bleiben.

HINWEIS ▶ Während `bash`-Shell-Skripte Skriptbefehle ähnlich wie die interaktive Befehlszeile interpretieren können, verwenden andere Skriptsprachen andere Befehle und Syntax.

Im Folgenden ist ein Beispielskript aufgeführt, das einige neue Befehle verwendet. Mit den Befehlen `pbpaste` und `pbcopy` greifen Sie auf den Inhalt der Zwischenablage zu. Die Zwischenablage ist ein temporärer Speicher, in dem alles gesichert ist, was Sie in einem Programm kopieren. Mit dem Befehl `sort` sortieren Sie Listen mit Objekten. Dieser Befehl nimmt den Inhalt der Zwischenablage, sortiert ihn in alphabetischer Reihenfolge und fügt das Ergebnis wieder in die Zwischenablage ein. Wie Sie sehen, wurden Pipes verwendet, um die Daten von einem Befehl zum nächsten weiterzuleiten.

```
#!/bin/bash

# Sorts the Clipboard contents, providing the content is a text list.

pbpaste | sort | pbcopy
```

Skriptfeedback

Feedback in Ihren Skripten hilft Ihnen dabei, festzustellen, welche Teile Ihres Skripts wann ausgeführt werden. Zwei Befehle, über die Sie Feedback in Ihren Skripten erhalten, sind `echo` und `date`.

Der Befehl `echo` gibt einfach nur das wieder, was Sie gerade in der Befehlszeile eingegeben haben. Auch wenn dies beim Arbeiten in Terminal trivial erscheint, ist es eine äußerst nützliche Funktion, um Feedback von einem Befehlsskript zu erhalten. Wenn Sie z. B. die Zeile `echo „Operation complete."` am Ende Ihres Skripts einfügen, wird der Text in Anführungszeichen bei Beendigung des Skripts in Terminal angezeigt.

Sie können diesen Befehl auch mit dem Umleitungszeichen >> kombinieren, damit Ihr Skript eine Protokolldatei erstellt. Wenn Sie z. B. die Zeile `echo „Operation complete" >> scriptlog.txt` einfügen, wird der Text in Anführungszeichen immer dann an die Datei „scriptlog.txt“ angehängt, wenn Ihr Skript diesen Punkt erreicht. Durch Hinzufügen des Befehls `date` und des Umleitungszeichens >> werden in der Protokolldatei Ihres Skripts auch Datum und Uhrzeit angegeben. Hierfür müssen Sie einfach `date >> scriptlog.txt` in Ihr Skript eingeben.

Verwenden von Variablen

Es gibt viele Sonderzeichen und Syntaxoptionen, die zur Logik Ihres Skripts beitragen. Schließlich ist Ihr Skript noch effizienter, wenn es in der Lage ist, Entscheidungen zu treffen und dynamisch auf Änderungen zu reagieren. Die gängigste Form der Skriptlogik ist die Verwendung von Variablen. Eine Variable ist einfach ein Ersatzwert für ein potenziell dynamisches Objekt. Angenommen, Ihr Skript verwendet einen bestimmten Pfad mehrere Male. Anstatt diesen Pfad nun mehrfach einzugeben, können Sie dafür eine Variable festlegen und stattdessen den Namen der Variablen verwenden. Das heißt, wenn der Pfad geändert werden muss, müssen Sie ihn nur einmal an der Stelle ändern, an der die Variable definiert ist.

Die Syntax für die Definition von Variablen lautet `variablename="variable value"`. Immer, wenn Sie den Wert der Variable in Ihrem Skript verwenden möchten, geben Sie einfach „`$variablename`" ein. Das folgende rudimentäre Beispielskript zeigt eine Variable, die mit dem Befehl `echo` verwendet wird. Der Name der Variable ist in diesem Fall äußerst lang. Die meisten Namen sind viel kürzer, um nicht so viel Text eingeben zu müssen. Variablen, die nur aus einem Buchstaben bestehen, sind nichts Außergewöhnliches. Das folgende Skript gibt den Text „Hello world!" in Terminal aus.

```
#!/bin/bash

# Test for echo.

myfirstvariable="Hello world! "

echo „$myfirstvariable"
```

Mit Variablen können Sie auch Argumente in Ihrem Skript implementieren. Wie bereits erwähnt, geben Sie Argumente nach einem Befehl ein, um die Objekte anzugeben, auf die der Befehl angewendet werden soll. Ihr Skript akzeptiert auch Argumente in Form von Zahlenvariablen. Bei der Variable „`$0`" handelt es sich immer um den Namen des Befehls (oder den Pfad zum Skript), mit dem das Skript aufgerufen wird. Jede Variable dahinter ist ein Argument, das nach dem Namen des Skripts in der Befehlszeile eingegeben wird, d. h. „`$1`", „`$2`", „`$3`" usw. Das folgende Skript ist so aufgebaut, dass zwei Argumente akzeptiert und diese dann in Terminal ausgegeben werden. Das heißt, um das Skript zu verwenden, müssten Sie `scriptname argument1 argument2` eingeben.

```
#!/bin/bash

# Test for echo, part two.

echo „$1"

echo „$2"
```

Sie haben auch die Möglichkeit, die Ausgabe anderer Befehle als Variablen zu verwenden. Die Syntax hierfür lautet `$(command string)`. Dem Skript wird mitgeteilt, den Befehl innerhalb der runden Klammern zuerst auszuführen und dann die Ergebnisse an der Stelle einzufügen, an der sich die Variable befindet. Im folgenden letzten Beispiel wird der Befehl `echo` erneut verwendet, um Text in Terminal auszugeben. In diesem Fall wird jedoch der Text dynamisch mithilfe des Befehls `hostname` erstellt, der den DNS-Hostnamen des Computers, und den Befehl `date`, der dann wiederum Datum und Uhrzeit ausgibt.

```
#!/bin/bash

# Test for echo, the third.

echo „This computer is named: $(hostname)"

echo „Today's date is: $(date)"
```

Ausführen von Befehlszeilenskripten

Wenn Sie Ihr Textdateiskript erstellt und gesichert haben, müssen Sie das Skript zunächst als ausführbare Datei definieren, bevor Sie es in der Befehlszeile ausführen können. Sie verwenden hierfür den Befehl `chmod`, um die Dateisystemzugriffsrechte des Skripts zu ändern und somit das Skript ausführbar zu machen. Im folgenden Beispiel verwendet Michelle den Befehl `chmod +x`, um das Skript „myscript.command" für alle Benutzer ausführbar zu machen.

```
MyMac:Desktop michelle$ chmod +x myscript.command
```

WEITERE INFORMATIONEN ▶ Ausführliche Informationen zu den Zugriffsrechten im Dateisystem finden Sie in Lektion 4, „Dateisysteme".

Wenn Ihr Skript ausführbar ist, können Sie es ausführen, indem Sie einen absoluten Pfad zu Ihrem Skript eingeben oder, falls sich das Skript im selben Ordner befindet, in dem Sie arbeiten, in der Befehlszeile `./` sowie den Dateinamen des Skripts eingeben. Dies erscheint vielleicht etwas umständlich, aber hierbei handelt es sich um eine UNIX-Sicherheitsmaßnahme, damit kein schädlicher Code ausgeführt wird. Alternativ können Sie Ihr Skript im Ordner „/usr/local/bin" ablegen, wodurch es sich dann in einem der Ordner des Standardpfads befindet. Objekte in diesen Ordnern müssen lediglich mit ihrem Dateinamen aufgerufen werden, damit sie in der Befehlszeile ausgeführt werden.

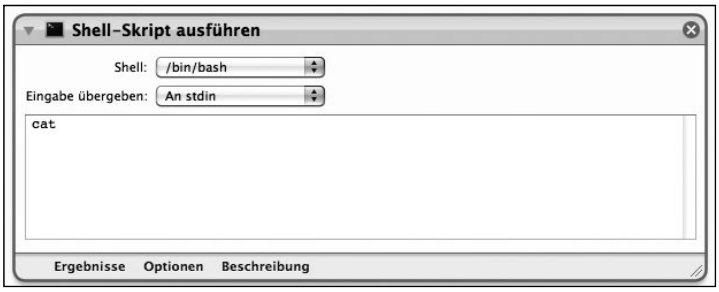
Kombinieren von Automatisierungstechniken

Die Automatisierungswerkzeuge der grafischen Benutzeroberfläche und der Befehlszeile unterscheiden sich zwar grundlegend, sie können jedoch so kombiniert werden, dass sie eine hybride Automatisierungslösung darstellen. Im folgenden Abschnitt lernen Sie die grundlegenden Techniken zur Integration dieser scheinbar völlig unterschiedlichen Automatisierungstechnologien kennen. Außerdem lernen Sie „Dienste" kennen, eine weitere Automatisierungstechnologie, die es Ihnen ermöglicht, eigene automatisierte Aufgaben zu Menüoptionen innerhalb von bestehenden Programmen hinzuzufügen.

Integrieren von Automator

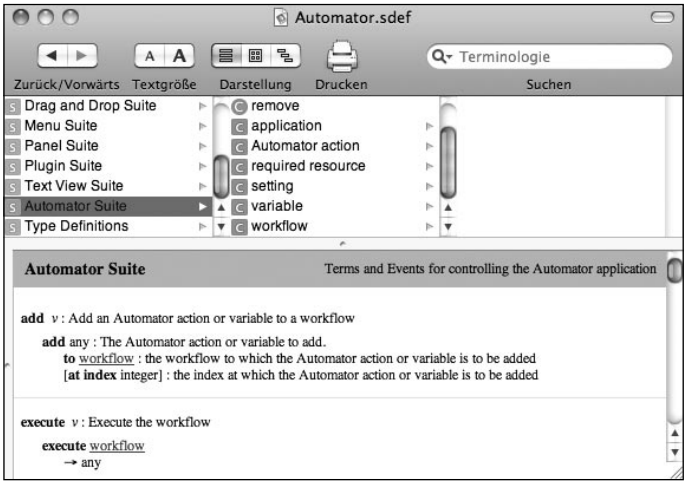
Wie bereits weiter vorne in dieser Lektion beschrieben, setzen sich Automator-Arbeitsabläufe aus Aktionen zusammen. Das heißt, Sie müssen für die Integration mit anderen Automatisierungstechnologien Aktionen verwenden. Zunächst können Sie mit der Aktion „AppleScript ausführen" AppleScript-Code ausführen. Fügen Sie hierfür einfach den Skriptinhalt im entsprechenden Fenster des Automator-Arbeitsablaufs ein.

Als nächstes können Sie mit der Aktion „Shell-Skript ausführen" Befehlszeilenskripte aus verschiedenen Sprachen ausführen (einschließlich Bash, Perl, Python, Ruby sowie mehreren Varianten der Shell-Umgebung). Fügen Sie im Aktionsfenster den Skriptinhalt ein oder geben Sie einen Pfad zu einem ausführbaren Skript an oder geben Sie einfach einen einzelnen Befehl ein.



Integrieren von AppleScript

Wie bereits weiter vorne in dieser Lektion beschrieben, bauen AppleScript-Textskripte auf der AppleScript-Sprache auf. Das heißt, Sie müssen die AppleScript-Syntax innerhalb des Skripts verwenden, um andere Automatisierungstechnologien zu integrieren. Da Automator auf AppleScript basiert, ist ein vollständiges Funktionsverzeichnis mit AppleScript-Begriffen verfügbar, die auf Automator-Aktionen und -Arbeitsabläufe verweisen. Auf die AppleScript-Funktionsverzeichnisse können Sie über *AppleScript-Editor* > *Ablage* > *Funktionsverzeichnis öffnen* zugreifen. Wählen Sie dann *Automator* aus der Liste aus.



Um ein Befehlszeilenskript oder einen Befehl über ein AppleScript-Skript auszuführen, geben Sie einfach `do shell script` ein, gefolgt von einem Pfad zu Ihrem Befehlszeilenskript in Anführungszeichen. Sie können hier nicht einfach den Skriptinhalt in das AppleScript-Skript einfügen oder einen Befehl eingeben, da AppleScript Anführungszeichen in der Mitte des Befehls u. U. als Ende des Befehls interpretiert. Dieses Problem können Sie umgehen, indem Sie den Anführungszeichen innerhalb des Befehls das Zeichen „\“ voranstellen, also: `do shell script „echo \"Hello world!\""`. Dies bedeutet allerdings, dass Sie vom standardmäßigen `bash`-Shell-Scripting abweichen müssen, wo diese Bedingung nicht gilt.

Integrieren von Befehlszeilenskripten

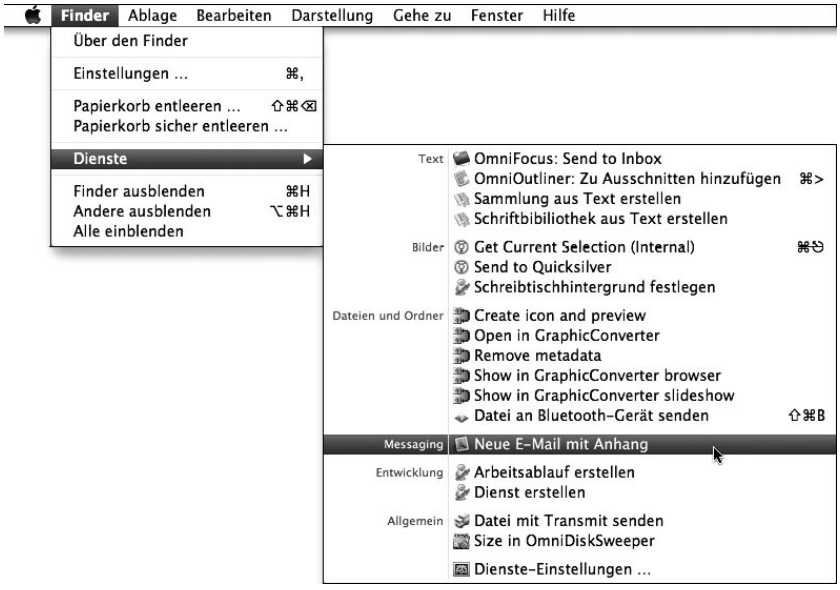
Wie bereits weiter vorne in dieser Lektion beschrieben, werden Befehlszeilenskripte mithilfe einer Skriptsprache und Befehlen erstellt. Das heißt, Sie müssen die Befehlszeilensyntax innerhalb des Skripts verwenden, um andere Automatisierungstechnologien zu integrieren. Zunächst einmal kann alles, auf das Sie im Finder doppelklicken können, in der Befehlszeile mithilfe des Befehls `open` ausgeführt werden. Geben Sie einfach `open` ein, gefolgt von dem Pfad zu dem Objekt, das Sie in der grafischen Benutzeroberfläche öffnen möchten. Dies schließt Automator-Arbeitsabläufe oder AppleScript-Skripte ein, die als eigenständiges Programm gesichert wurden.

Sie können auch den Befehl `osascript` verwenden (gefolgt von einem Pfad zu dem Skript), um AppleScript-Skripte in deren Textformat auszuführen. Der Unterschied ist, dass der Befehl `open` sich zwar ähnlich wie ein Doppelklicken auf die Skriptdatei im Finder verhält, nur würde dadurch das Skript im AppleScript-Editor geöffnet und nicht ausgeführt werden. Außerdem können Sie AppleScript-Befehle direkt in einem Befehlszeilenskript mithilfe des Befehls `–e` angeben, z. B. `osascript -e „tell application \"Finder\" to activate"`.

HINWEIS ▶ Ähnlich wie beim AppleScript-Befehl `do shell script` ist es u. U. schwierig zu unterscheiden, welche Anführungszeichen von der Shell und welche von AppleScript interpretiert werden. Im vorliegenden Beispiel werden die Anführungszeichen vor \ von der Shell ignoriert und von AppleScript interpretiert.

Verwenden von eigenen Diensten

Dienste ermöglichen im Wesentlichen, aus den Menüs eines Programms heraus auf Funktionen eines anderen Programms zuzugreifen. Auf die Menüobjekte in *Dienste* können Sie über das Programmmenü oder per Rechtsklick auf ein ausgewähltes Objekt in einem Programm zugreifen. Das Menü *Dienste* ist automatisch und dynamisch, da es neue Dienste beim Hinzufügen zu Ihrem Computer automatisch erkennt und nur Dienste anzeigt, die auf das aktuell ausgewählte Objekt anwendbar sind. Die folgende Abbildung zeigt das Menü *Dienste* im Finder bei Auswahl einer Image-Datei.



HINWEIS ▶ Diese Abbildung des Menüs *Dienste* zeigt integrierte Dienste sowie Dienste von Drittanbietern.

Dienste stellt Mac OS X schon seit geraumer Zeit bereit, aber sie wurden in Mac OS X Version 10.6 wesentlich verbessert. Das Menü *Dienste* wurde so gestaltet, dass nun klarer hervorgeht, welche Programme Dienste anbieten. Außerdem können einzelne Dienste nun

manuell aktiviert oder deaktiviert werden und, was viel bedeutender ist, eigene Tastaturkurzbefehle können zur Aktivierung der Dienste erstellt werden. Diese neuen Diensteinstellungen befinden sich in der Systemeinstellung *Tastatur* im Bereich *Tastaturkurzbefehle*.

Die bedeutendste Neuerung hinsichtlich der Dienste in Mac OS X 10.6 ist, dass Sie ganz einfach neue eigene Dienste auf der Basis eines Automator-Arbeitsablaufs erstellen können. Wählen Sie hierfür die Vorlage *Dienst* aus, wenn Sie ein neues Automator-Projekt öffnen. Oben im Arbeitsablauf wählen Sie den Typ des ausgewählten Objekts, den der Dienst empfangen soll. Anschließend können Sie Ihren eigenen Arbeitsablauf erstellen. Wenn Sie den Dienst-Arbeitsablauf sichern, wird er automatisch zur Liste der verfügbaren Dienste hinzugefügt und kann sofort verwendet werden.

Die Abbildung zeigt einen nützlichen eigenen Dienst, der innerhalb von nur 30 Sekunden erstellt wurde. Dieser Dienst empfängt den ausgewählten Text eines beliebigen Programms, sortiert dann die Zeilen des Texts alphabetisch und ersetzt den Text anschließend. Wie sich herausstellt, gibt es keine Automator-Aktion zum Sortieren von ausgewähltem Text, aber es gibt einen `sort`-Befehl. Deshalb wurde die Aktion „Shell-Skript ausführen“ hinzugefügt, wobei nur der Name des Befehls in das Shell-Skript-Feld eingegeben werden musste. Das Ergebnis ist, dass die Funktionalität des Befehls `sort` in ein beliebiges grafisches Programm integriert wurde, das Text auswählen und ersetzen kann.

WEITERE INFORMATIONEN ▶ Zusätzliche Beispiele für Dienste sowie weitere Dokumentation finden Sie unter www.macosxautomation.com.

Das haben Sie gelernt

- Bei der Befehlszeile handelt es sich um eine leistungsstarke Methode für die Arbeit am Mac, da es die einzige Methode ist, mit der Sie bestimmte Verwaltungsaufgaben effizient erledigen können.
- Mit den Befehlen `ls`, `pwd` und `cd` können Sie im Dateisystem mithilfe der Befehlszeile navigieren.
- Sie haben grundlegende Techniken und Tastaturkurzbefehle für die Befehlszeile kennengelernt, um Zeit zu sparen und Fehler zu vermeiden, wie das automatische Vervollständigen zur automatischen Eingabe von Pfadnamen.
- Administratoren können in der Befehlszeile mithilfe des Befehls `sudo` als Systemadministratoren (root) agieren.
- Automator und AppleScript können zum Automatisieren von Aufgaben in Programmen der grafischen Benutzeroberfläche verwendet werden, während das Befehlszeilen-Scripting zum Automatisieren von Systemverwaltungsaufgaben dient.
- Alle drei Automatisierungstechniken, Automator, AppleScript und Befehlszeilenskripte, können so miteinander kombiniert werden, dass nahezu jede Aufgabe automatisiert werden kann.

Literatur

Sie können unter www.apple.com/de/support nach neuen und aktualisierten Knowledge Base-Artikeln suchen.

URL-Adressen

Die Clientverwaltungssoftware „Apple Remote Desktop“: www.apple.com/de/remotedesktop

Mac OS X Automatisierungsressourcen: www.macosxautomation.com

Mac OS X Automatisierungsressourcen: http//macscripter.net

AppleScript-Website für Entwickler: http://developer.apple.com/applescript

TextWrangler von BareBones Software: http://www.barebones.com

Der „Advanced Bash-Scripting Guide“: http://tldp.org/LDP/abs/html

Bash Pitfalls (eine Liste mit häufigen Fehlern beim Shell-Scripting): http://mywiki.wooledge.org/BashPitfalls

Das Gelernte überprüfen

1. Nennen Sie sechs Gründe für die Verwendung der Befehlszeilenumgebung.
2. Welche vier Methoden können für den Zugriff auf die Befehlszeilenumgebung verwendet werden?
3. Welche drei Objekte werden standardmäßig in der Eingabeaufforderung angezeigt?
4. Wie lauten die drei Komponenten eines typischen Befehls?
5. Was beschreiben die folgenden Begriffe: Ordner, Verzeichnis, Pfad, absoluter Pfad und relativer Pfad?
6. Was ist der Unterschied zwischen absoluten und relativen Pfaden?
7. Mit welchem Befehl werden Objekte in einem Ordner aufgelistet?
8. Mit welchen beiden Befehlen können Textdateien gelesen werden?
9. Wofür wird der Befehl `sudo` verwendet?
10. Wie lauten die beiden primären Automatisierungstechnologien für die grafische Benutzeroberfläche? Inwieweit unterscheiden sie sich?
11. Welches sind die drei Schritte, die für das Erstellen eines Befehlszeilenskripts mindestens erforderlich sind?

Antworten

1. Sechs Gründe für die Verwendung der Befehlszeile: Sie stellt Zugriffsoptionen bereit, die auf der grafischen Benutzeroberfläche nicht verfügbar sind. Benutzer können Beschränkungen im Finder umgehen. Administratoren können in der Befehlszeile als Systemadministrator (root) agieren. Fernzugriff über SSH ist für den Benutzer unsichtbar. Die Befehlszeile erleichtert die Automatisierung mithilfe von Skripten. Benutzer können die Befehlszeile mit ARD kombinieren, um administrative Befehle entfernt an mehrere Macs gleichzeitig zu senden.
2. Vier Methoden für den Zugriff auf die Befehlszeile: das Programm „Terminal", Eingeben von „>console" im Anmeldefenster, Starten im Einzelbenutzermodus und entfernte Anmeldung über SSH.
3. Die drei Objekte in der standardmäßigen Eingabeaufforderung lauten von links nach rechts: Computer-Hostname, Arbeitsverzeichnis und Benutzeraccount.
4. Die drei Hauptkomponenten eines typischen Befehls lauten: der Name des Befehls, Befehlsoptionen und Befehlsargumente.
5. Ordner und Verzeichnisse sind beides Begriffe für Container im Dateisystem. Ein Pfad beschreibt die Position eines bestimmten Objekts im Dateisystem. Absolute Pfade sind komplette Positionsbestimmungen für ein Objekt, während relative Pfade nur eine teilweise Beschreibung sind, die von der aktuellen Position des Benutzers ausgehen.
6. Absolute Pfade beginnen immer auf der obersten Ebene des Dateisystems (root-Verzeichnis), relative Pfade an der aktuellen Position. Die standardmäßige Ausgangsposition von Benutzern ist die oberste Ebene ihres Benutzerordners.
7. Der Befehl `ls` zeigt Objekte in einem Ordner an.
8. Die beiden Befehle, mit denen Sie Textdateien lesen können, lauten `cat` und `less`.
9. Mit dem Befehl `sudo` können Administratoren Befehle mit root-Zugriffsrechten ausführen.
10. Automator ist ein intuitives Programm zum Erstellen von Arbeitsabläufen, die auf vordefinierten Aktionen beruhen. AppleScript ist eine der englischen Sprache ähnelnde Skriptsprache, mit der Sie Programme der grafischen Benutzeroberfläche automatisieren können.
11. Die drei Schritte, die für das Erstellen eines Befehlszeilenskripts erforderlich sind: Erstellen einer reinen Textdatei mit einer Liste von Befehlen, Eingeben von „#!/bin/bash" in die erste Zeile und Ändern der Dateizugriffsrechte, um die Ausführung des Skripts zu ermöglichen.