



Apple Skripterstellung für Administratoren (Einsteiger

...und Neueinsteiger)

Wenn Sie Apple Administrator sind und Ihnen die Skripterstellung schwerfällt, nur Mut: Dieser Leitfaden ist für Sie gemacht!

Mit nur wenigen einfachen Skripten können Sie Ihre Geschwindigkeit, Genauigkeit und die Zufriedenheit der Endanwender verbessern.

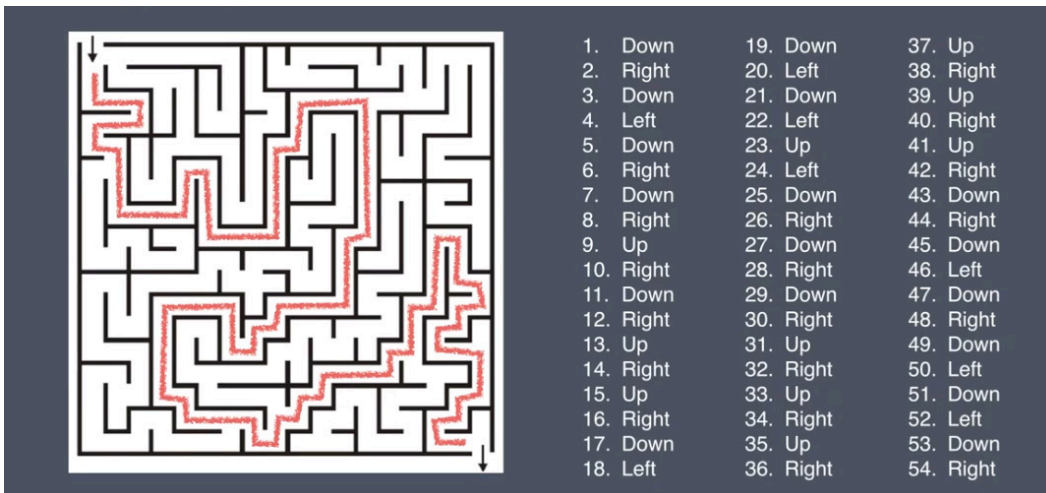
Wenn Sie schon einmal mit Skripten gearbeitet haben und das schwierig fanden, gibt es wahrscheinlich einen Grund dafür: Bei den meisten Tutorials werden Ihnen direkt Skripte vorgestellt, die es auswendig zu lernen gilt, ob diese nun für Sie nützlich sind oder nicht. Sinnvoller ist es jedoch, zunächst zu erklären, was Skripterstellung bedeutet und warum es sinnvoll ist, die Grundlagen zu erlernen.

Der Einstieg fällt nicht schwer, wenn Sie erst einmal verstanden haben, wie und weshalb die Skripterstellung funktioniert, bevor Sie versuchen, Befehlsfolgen auswendig zu lernen.

Wozu Skripte erstellen?

Mit Skripten lassen sich Aufgaben vereinfachen.

Wir können komplexe Aufgaben in eine Reihe grundlegender Befehle zerlegen. Dies lässt sich mit der Wegführung durch ein Labyrinth vergleichen. Dazu braucht es nur vier Befehle: nach oben, nach unten, nach rechts, nach links. Man zerlegt das Problem also in kleine Schritte, wie im folgenden Beispiel:



Mit nur vier Befehlen lässt sich eine scheinbar schwierige Aufgabe bewältigen.

Mit Skripten lassen sich Aufgaben automatisieren.

Zum Öffnen einer Datei auf dem Computer braucht es nur einen Doppelklick. Zum Öffnen einer Datei auf zwei Computern braucht es zwei Doppelklicks. Was aber, wenn Sie 24 Dateien auf 24 Computern gleichzeitig öffnen müssen?

Mit einer MDM-Lösung wie Jamf Pro können wir diesen Befehl auf allen Computern mit einem einfachen einzeiligen Befehl automatisieren: Öffnen, zusammen mit der Angabe des Speicherorts der Datei.

Mit Skripten lassen sich Fehler vermeiden.

Wenn Aufgaben menschliches Eingreifen erfordern, sind Fehler durch den Bediener nicht ausgeschlossen. Fehler können dazu führen, dass Arbeiten wiederholt werden müssen. Außerdem erhöht sich dadurch die Anzahl der Anfragen beim Helpdesk. Routinemäßige Skriptaufgaben stellen sicher, dass diese immer korrekt (und schnell!) erledigt werden.

Wie funktioniert „Terminal“?

Befehlsinterpreter

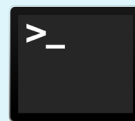
Sie wissen nicht, was das ist? Doch, ganz sicher!

Sie nutzen Befehlsinterpreter schon seit Jahren. Die bekanntesten Interpreters sind Windows Explorer und Mac Finder. Ihre Aufgabe ist es, Mausclicks und Tastaturbefehle in Computerschritte umzuwandeln. Wenn Sie auf eine Datei klicken, interpretieren sowohl der Windows Explorer als auch der Mac Finder dies so, dass Sie diese Datei auswählen möchten. Wenn Sie auf eine Datei doppelklicken, interpretieren die Programme, dass Sie diese Datei öffnen möchten.

Doch nicht alle Befehlsinterpreter interpretieren Ihre Handlungen auf dieselbe Weise. Ein Beispiel: Wenn Sie in Explorer eine Datei auswählen und dann die Eingabetaste drücken, wird die Datei geöffnet. Wenn Sie auf dem Mac so vorgehen, können Sie den Dateinamen ändern.

Terminal selbst ist kein Befehlsinterpreter. Es dient lediglich als Fenster zu dem Interpreter Bash. Der Mac verfügt über mehrere Befehlsinterpreter, doch in diesem E-Book liegt das Hauptaugenmerk auf Bash.

Wenn Sie Terminal im Ordner „Dienstprogramme“ der Anwendung öffnen, werden Sie automatisch beim Interpreter Bash angemeldet. Bash ist eine Abkürzung für „Bourne-Again-SHell“. Es handelt sich um eine Überarbeitung des Interpreters Shell (sh), der vor 40 Jahren von Steven Bourne entwickelt wurde.



Im Terminal führen Apple Benutzer Skripte aus. Wenn Sie wissen, wie Terminal funktioniert, wird die Skripterstellung klarer.



Wie funktionieren Befehle?

Einfach gesagt teilt man mit Befehlen dem Computer mit, was er tun soll. Mit Befehlen können Sie alles erledigen, was Sie auch mit dem Finder tun können. Sie können beispielsweise eine Datei öffnen, indem Sie auf sie doppelklicken. Sie können aber auch Terminal öffnen und den folgenden Befehl eingeben, wobei Sie den Namen der Datei durch den einer Datei ersetzen, die sich auf Ihrem Schreibtisch befindet:

```
open /Users/jamfwebinar/Desktop/scripting101.docx
```

Dasselbe funktioniert auch mit einer Webseite statt einer Datei. Auch dazu kann man den Befehl „open“ nutzen. Man muss lediglich die URL und den Browser angeben, der zum Öffnen der Webseite verwendet werden soll:

```
open https://www.jamf.com -a Safari
```

Jetzt sind Sie dran: Sie schöpfen das volle Potenzial dieses E-Books aus, wenn Sie Terminal öffnen und die beschriebenen Schritte dort nachvollziehen.

Programme > Dienstprogramme > Terminal

Beginnen wir mit den Grundlagen.

Das erste Programm, das die meisten Menschen in Terminal erlernen, ist „**Hello, World**“.

Dies ist zwar eine einfache Übung. Doch Sie werden den Befehl „echo“ künftig bei der Skripterstellung auf viele verschiedene Arten verwenden. Probieren Sie es aus!

Geben Sie in Terminal Folgendes ein:

```
echo 'Hello World'
```

Drücken Sie dann die Eingabetaste.

Wie Sie sehen, bedeutet der Befehl echo: **Wiederholen, was ich sage**.

Um einem Befehl echte Wirksamkeit zu verleihen, fügen Sie ein Argument hinzu: den Text innerhalb der einfachen Anführungszeichen. Mit dem Argument wird dem Befehl mitgeteilt, was zu tun ist.

Um weitere einfache Skripte zu erlernen, die in Ihren täglichen Arbeitsabläufen relevant sind, können Sie mit grundlegenden Dingen beginnen, die Sie sonst auf dem Schreibtisch erledigen würden. Eine gute Übung ist es, dies den ganzen Tag über ganz selbstverständlich zu tun, wenn Sie mit dem Finder Routineaufgaben erledigen, damit Sie bei der Skripterstellung sicherer werden.

Sie wissen bereits, wie Sie im Mac Finder folgende Aufgaben erledigen:

- Einen Ordner mit der Bezeichnung «MyStuff» anlegen
- Eine Datei mit Inhalten erstellen
- Die Datei in dem Ordner ablegen

Diese Aufgaben können Sie mit den folgenden Befehlen auch in Terminal erledigen:

`mkdir` = make directory (neues Verzeichnis erstellen)
`echo` = Wiederholen, was ich sage
`mv` = move (verschieben oder umbenennen; in dieser Übung verwenden wir die Funktion zum Verschieben)

Sie können die Befehle auch auf neuartige Weise nutzen.

Um auf dem Schreibtisch den Ordner «MyStuff» zu erstellen, geben Sie in der Befehlszeile in Terminal Folgendes ein:

```
mkdir Desktop/MyStuff
```

Drücken Sie dann die Eingabetaste.

Um eine Datei mit Inhalt zu erstellen, gehen Sie umgekehrt vor. Erstellen Sie zunächst den Inhalt und verweisen Sie diesen (>) an den Speicherort und das Dateiformat, in dem der Inhalt gespeichert werden soll. Verwenden Sie dazu das übliche Format für einen Dateispeicherort:

```
echo "The quick brown fox jumped over the lazy dogs." > Desktop/Quick-Brown-Fox.txt
```

Drücken Sie dann die Eingabetaste.

Legen Sie die Datei in dem Ordner ab. Geben Sie in der Befehlszeile Folgendes ein:

```
mv Desktop/Quick-Brown-Fox.txt Desktop/MyStuff
```

Drücken Sie dann die Eingabetaste.

Die Befehlszeile in Terminal macht genau das, was Sie ihr anweisen. Sie gibt keine Bestätigungsmeldung aus (es sei denn, Sie haben sie dazu angewiesen).

Prüfen Sie daher zur Bestätigung im Finder, dass der neue Ordner „MyStuff“ erstellt wurde und dieser das Dokument Quick-Brown-Fox.txt enthält.

Sie können das Dokument in Terminal auch öffnen!

Geben Sie Folgendes ein:

```
open Desktop/MyStuff/Quick-Brown-Fox.txt
```

Drücken Sie dann die Eingabetaste.

Sie können im Terminal mit dem Befehl `rm` auch Dokumente und Ordner löschen.

`rm` = remove (entfernen)

`rm -r` = remove recursively (rekursiv entfernen)

Zum Entfernen eines Dokuments benötigen Sie nur den Befehl `rm`. Testen Sie den folgenden Befehl:

```
rm Desktop/MyStuff/Quick-Brown-Fox.txt
```

Schauen Sie mit dem Finder im Ordner nach. Das Dokument Quick-Brown-Fox.txt ist nicht mehr vorhanden.

Sehen Sie, was geschieht, wenn Sie mit demselben Befehl einen Ordner entfernen:

```
rm Desktop/MyStuff
```

Wenn Terminal den von Ihnen erteilten Befehl nicht ausführen kann, wird, wie Sie gesehen haben, eine Fehlermeldung angezeigt:

```
rm: Desktop/MyStuff: is a directory
```

Um Terminal anzuweisen, das Sie den Ordner bzw. das Verzeichnis und alle darin enthaltenen Inhalte **WIRKLICH** entfernen möchten, müssen Sie den Befehl zum rekursiven Entfernen verwenden, also `rm -r`.

Probieren Sie diesen Befehl in Terminal aus. Überprüfen Sie dann, dass der Ordner MyStuff nicht mehr vorhanden ist.

```
rm -r Desktop/MyStuff
```

Hinweis: Wenn der Ordner- bzw. Dokumentname Leerzeichen enthalten soll, müssen Sie den Namen in Anführungszeichen einschließen, z. B. „My Stuff“ bzw. „Quick Brown Fox.txt“.

Skripte aus Terminal-Befehlen erstellen

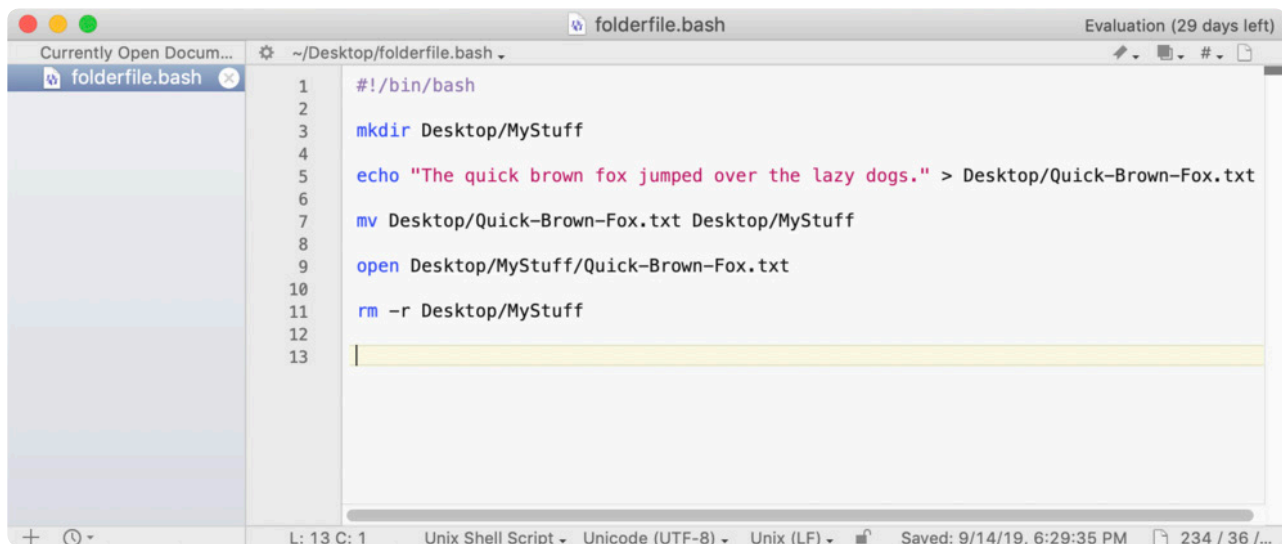
Jetzt erstellen wir ein Skript aus den Schritten, die Sie soeben in Terminal ausgeführt haben!

Laden Sie zunächst einen Skript-Editor herunter und öffnen Sie ihn. Wir empfehlen BBEdit, doch am Ende dieses Buches werden auch noch weitere Editoren als Ressourcen genannt. Sie können auch einen Texteditor verwenden. Ein kostenloser Skript-Editor bietet jedoch den Vorteil, dass die Syntax, also spezielle Teile von Skripten, mit unterschiedlichen Farben gekennzeichnet ist. So erkennen Sie auf einen Blick, um welche Teile es sich handelt, und Sie können Fehler schnell auffinden und korrigieren. Dies ist für Neulinge bei der Skripterstellung und bei der Fehlersuche und -behebung besonders hilfreich. In diesem Beispiel werden **Befehle in Blau** und **Argumente in Rosa** dargestellt.

Geben Sie zu Beginn jedes Skripts zunächst immer das so genannte „Shebang“ (bzw. Hash-Bang) ein: Dabei handelt es sich um die beiden Zeichen `#!/`. Folgt darauf `/bin/bash`, teilen Sie damit Terminal mit, dass bei der Verwendung der Skripte Bash als Befehlsinterpreter verwendet werden soll.

```
#!/bin/bash
```

Kopieren Sie die einzelnen Befehlszeilen, die Sie bereits genutzt haben, und fügen Sie sie in den Skript-Editor ein. Wenn Sie Leerzeilen eingeben, wird der Text besser lesbar.



```
1  #!/bin/bash
2
3  mkdir Desktop/MyStuff
4
5  echo "The quick brown fox jumped over the lazy dogs." > Desktop/Quick-Brown-Fox.txt
6
7  mv Desktop/Quick-Brown-Fox.txt Desktop/MyStuff
8
9  open Desktop/MyStuff/Quick-Brown-Fox.txt
10
11 rm -r Desktop/MyStuff
12
13
```

Speichern Sie das Skript unter dem Namen „**folderfile.bash**“ auf dem Schreibtisch.

Ziehen Sie diese Datei nun in Terminal und drücken Sie die Eingabetaste.

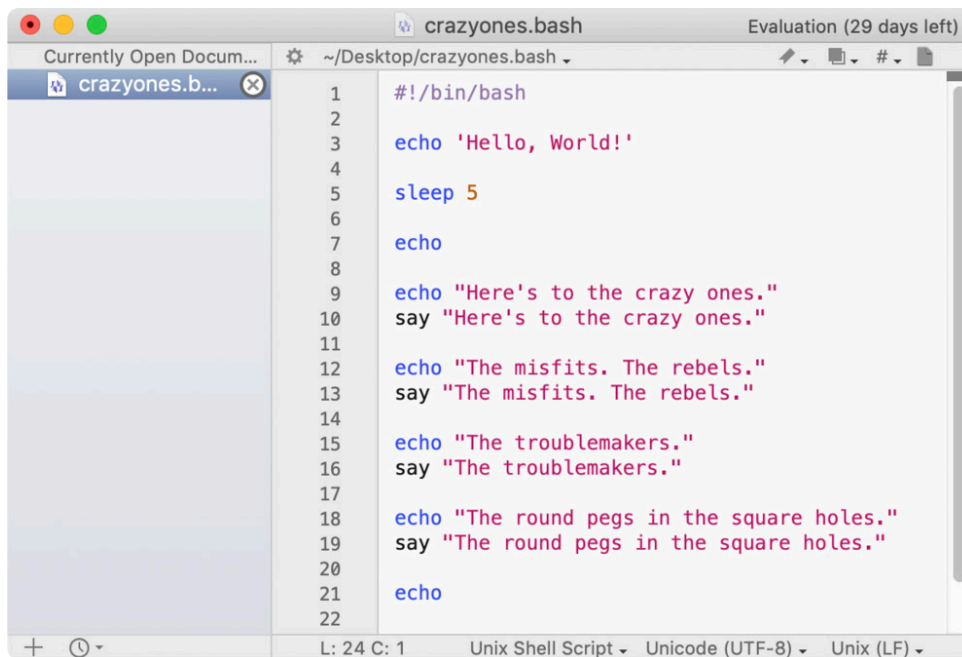
Haben Sie gerade geblinzelt? Dann haben Sie die Ausführung des Skripts vielleicht gar nicht mitbekommen! Das ist ein weiterer Vorteil von Skripten: Sie sind blitzschnell!

Das folgende Skript ist cool. Dieses Mal beginnen wir im Skript-Editor.

Öffnen Sie den Skript-Editor und geben Sie Folgendes ein:

```
#!/bin/bash
echo 'Hello, World!'
sleep 5
echo
echo "Here's to the crazy ones."
say "Here's to the crazy ones."
echo "The misfits. The rebels."
say "The misfits. The rebels."
echo "The troublemakers."
say "The troublemakers."
echo "The round pegs in the square holes."
say "The round pegs in the square holes."
echo
exit
```

Das sieht dann so aus:



The screenshot shows a macOS text editor window titled 'crazyones.bash'. The window has a title bar with three colored buttons (red, yellow, green) and a status bar on the right that says 'Evaluation (29 days left)'. The main editing area shows the following code:

```
1  #!/bin/bash
2
3  echo 'Hello, World!'
4
5  sleep 5
6
7  echo
8
9  echo "Here's to the crazy ones."
10 say "Here's to the crazy ones."
11
12 echo "The misfits. The rebels."
13 say "The misfits. The rebels."
14
15 echo "The troublemakers."
16 say "The troublemakers."
17
18 echo "The round pegs in the square holes."
19 say "The round pegs in the square holes."
20
21 echo
22
```

The status bar at the bottom indicates 'L: 24 C: 1', 'Unix Shell Script', 'Unicode (UTF-8)', and 'Unix (LF)'.

Wie Sie sehen, ist Zeile 3 ein echo-Befehl mit „Hello, World!“ echo ist Blau, und die Variablen sind Rosa.

Die fünfte Zeile lautet `sleep 5`.

`sleep` bedeutet Pause, und 5 heißt, dass die Ausführung 5 Sekunden lang pausiert werden soll.

`say` ist ein weiterer Befehl. Sie sehen gleich, was er bedeutet.

Der Befehl `echo` ohne weiteres Argument fügt einfach einen Leerraum ein: „`echo [Leerraum]`“.

Alle diese Befehle in einer Liste bilden zusammen ein Skript.

1. Speichern Sie dieses Skript wie Ihr voriges Skript. Verwenden Sie `.bash` als Dateisuffix.
2. Wählen Sie die Datei aus und ziehen Sie sie in das Terminal.
3. Achten Sie darauf, dass die Tonwiedergabe am Computer eingeschaltet ist.
4. Drücken Sie die Eingabetaste. Das Skript wird ausgeführt.

Wissen Sie jetzt, was wir gemacht haben? Jetzt wissen Sie, was der Befehl `say` bedeutet.

Diese Befehle haben Sie bisher erlernt:

`echo` = Wiederholen, was ich eingebe

`sleep` = Pause (wird in Verbindung mit einem Argument für die Dauer der Pause in Sekunden verwendet)

`say` = Vorlesen

`open` = Eine Datei öffnen

`mkdir` = Verzeichnis/Ordner erstellen

`mv` = Eine Datei verschieben bzw. umbenennen

`rm` = Eine Datei entfernen

`rm -r` = Rekursiv entfernen (wird zum Entfernen eines Ordners/Verzeichnisses verwendet)



Wie Skripte funktionieren und wie die Skripterstellung mit Jamf optimiert werden kann

Variablen: Platzhalter für nachfolgende Elemente

Variablen sind Platzhalter für nachfolgende Elemente. Die Werte, die wir ihnen zuweisen, können Zahlen, Texte, Dateinamen, Gerätenamen oder sonstige Daten sein. Eine Variable ist kein Element im eigentlichen Sinne. Sie VERWEIST auf echte Daten, die später eingebunden werden.

Variablen sind sehr nützlich, denn wenn sie korrekt definiert werden, kann man dasselbe Skript unverändert vielfach ausführen, wenn man mit veränderlichen Daten arbeitet.

Schauen wir ihre Funktionsweise anhand der einfachsten Variablen an: ein Name. Dabei gehen wir wieder in umgekehrter Reihenfolge vor, wie weiter oben beim Erstellen einer Datei mit Inhalten.

Zunächst teilen wir dem Computer mit, um welche Art von Daten es sich handelt. Geben Sie in diesem Fall Folgendes in Terminal ein:

```
myName=Martin
```

Drücken Sie dann die Eingabetaste. Verwenden Sie immer klar verständliche, für die spätere Verwendung leicht zu merkende Variablennamen.

Dann weisen wir den Computer an, wie er die Daten verarbeiten soll. Bei der Verwendung von Variablen müssen diese immer mit dem `$`-Symbol gekennzeichnet werden. Geben Sie Folgendes in Terminal ein:

```
echo Hello, my name is $myName.
```

Drücken Sie dann die Eingabetaste.

In Terminal wird Folgendes angezeigt:

```
Hello, my name is Martin.
```

Dafür, dass eine Variable erst definiert und dann in einem Skript verwendet wird, gibt es viele praktische Anwendungsmöglichkeiten in Ihren Arbeitsabläufen.

Einzeilige Befehle

Es folgt ein weiterer nützlicher Befehl, der durch eine Variable modifiziert wird. Nehmen wir an, Sie möchten feststellen, ob für einen Mac ein Betriebssystem-Upgrade erforderlich ist.

Auch hier gehen wir wieder in umgekehrter Reihenfolge vor.

Zunächst müssen wir überlegen, welche Variable wir verwenden werden. Dann werden wir das Ergebnis mithilfe eines Befehls ermitteln.

Dazu benötigen wir zwei neue Befehle (eigentlich ist es nur einer, der dann näher bestimmt wird):

```
sw_vers = Softwareversion  
sw_vers -productVersion = nur die Produktversion der Software
```

Der Befehl `sw_vers` liefert uns Informationen über macOS: die Bezeichnung, die Version und die Build-Nummer des verwendeten Betriebssystems. Wenn Sie nur die Version benötigen, können Sie in Ihrem Skript nach dem Befehl den Zusatz `-productVersion` anbringen. So erhalten Sie nur die gewünschten Informationen.

Probieren Sie es in Terminal mit dem Befehl einfach selbst aus.

Bei der ersten Version wird ein Ergebnis wie das Folgende angezeigt:

```
ProductName:  Mac OS X  
ProductVersion:  10.14.4  
BuildVersion:  18E226
```

Bei der zweiten Version lautet das Ergebnis einfach:

```
10.14.4
```

Wenn wir mit bereits vorhandenen Befehlen arbeiten, müssen wir die Variable nicht vorab festlegen. Wir müssen allerdings angeben, dass wir eine Variable verwenden. Die Variable können Sie folgendermaßen definieren:

```
$( sw_vers -productVersion )
```

Wir verwenden wieder das `$`-Zeichen, doch hier verwenden wir den Befehl für die Softwareversion als Variable. Den Befehl klammern wir dann ein. Wie in der Mathematik wird der Ausdruck in Klammern zuerst berechnet, dann erst die Komponenten, die außerhalb der Klammern stehen.

Versuchen Sie es selbst, indem Sie Folgendes in die Befehlszeile eingeben:

```
echo My operating system is $( sw_vers -productVersion ).
```

Damit die Befehle noch nützlicher sind, müssen wir so genannte Bedingungen einsetzen.

Das ist ganz logisch: Mithilfe von Bedingungen in Form von Wenn-dann-Befehlen können Skripte logische Entscheidungen treffen

Bedingungen sind Wenn-dann-Fragen. Mithilfe von If/then-Anweisungen werden Entscheidungen getroffen: Wir geben die Bedingungen vor, die erfüllt sein müssen. Dann weisen wir den Computer an zu ermitteln, ob diese Bedingungen erfüllt sind.

Beispiel:

WENN ich Zitronen habe, DANN mache ich Limonade.

Sie können Variablen und Bedingungen kombinieren, um sehr nützliche Ergebnisse zu erhalten.

Geben Sie beispielsweise Folgendes im Skript-Editor ein:

```
#!/bin/bash
osVersion=$( sw_vers -productVersion )
if [ $osVersion = "10.14.5" ]; then
    echo No upgrade needed.
fi
```

Mit einer Variable am Anfang kann der Administrator abrufen, welche Betriebssystemversion auf dem Mac läuft.

Durch die mit `if` eingeleitete Bedingung und den dann folgenden Befehl `echo` wird Terminal angewiesen zu prüfen, welche Betriebssystemversion auf dem Mac läuft. Wenn es sich dabei um die vom Administrator gewünschte Version handelt, meldet Terminal „No upgrade needed“ (Kein Upgrade erforderlich).

Bedingungen werden mit `fi` (also „if“ rückwärts gelesen) abgeschlossen. Das ist doch ganz schön clever, oder?

Probieren Sie es in Terminal selbst aus. Speichern Sie dazu dieses Skript und ziehen Sie es in das Terminal.

Doch was geschieht, wenn die `if`-Bedingung nicht erfüllt ist? Fügen Sie einfach eine `else`-Anweisung hinzu und geben Sie an, was in diesem Fall geschehen soll. Else ist eine Kurzform von „or else“ (also „andernfalls“).

Ergänzen Sie das Skript wie folgt:

```
#!/bin/bash
osVersion=$( sw_vers -productVersion )
if [ $osVersion = "10.14.5" ]; then
    echo No upgrade needed.
else
    echo Update required.
fi
```

Führen Sie das Skript nun in Terminal aus.

Kann man ein Skript schreiben, das nicht nur feststellt, welche Betriebssystemversion auf dem Mac läuft, sondern bei Bedarf auch automatisch die aktuellste Version installiert?

Selbstverständlich kann man das, und zwar mithilfe von Jamf Richtlinien.

Richtlinien einrichten

In diesem Beispiel weisen wir mit der Binärdatei „Jamf Helper“ den Mac an, ein Ereignis auszulösen: ein Betriebssystem-Upgrade.

Bei jedem registrierten Mac installiert Jamf Pro seine eigene jamf Befehlszeile, das sogenannte Binärtool. Mit dem Befehl jamf werden spezielle Befehle des Jamf Management angesteuert, wie etwa die Ausführung von Richtlinien. Wir können jeder Richtlinie einen eindeutigen Namen zuweisen, einen sogenannten „Custom Trigger“. Mit dem Befehlszeilentool jamf können wir die Richtlinie dann über ihren Custom-Trigger-Namen aufrufen und ausführen.

In diesem Fall ruft das von uns verwendete Skript die Richtlinie „runUpgrade“ auf, die genau das macht, was ihr Name besagt: Sie führt ein Software-Upgrade aus.

Um dies auf mehreren Computern gleichzeitig erledigen zu können, benötigen Sie eine MDM-Lösung wie etwa Jamf Pro.

Um dieses Skript in Jamf Pro einzubinden, wählen Sie die Menübefehle **Settings > Computer Management > Scripts**. Klicken Sie dann auf die Taste „New“.

Hier können Sie dem Skript eine Bezeichnung zuweisen, z. B. CheckOSVersionandRunUpgrade, Sie können Notizen zum Skript hinzufügen, das Skript im Register „Script“ eingeben, Optionen und Einschränkungen angeben und das Skript speichern.

Geben Sie im Register „Script“ Folgendes ein:

```
#!/bin/bash
osVersion=$( sw_vers -productVersion )
if [ $osVersion = "10.14.5" ]; then
    echo No upgrade needed.
else
    jamf policy -event runUpgrade
fi
```

Jamf Pro verfügt über eine Skriptansicht, in der die Art der Skripterstellung wie in einem Skript-Editor gekennzeichnet wird. Wählen Sie aus der Dropdown-Liste die Option „Shell“. Dann wird künftig standardmäßig dieser besser lesbare Editor verwendet.

Sie können echo-Befehle hinzufügen, die dann ihrerseits wieder in die Richtlinienprotokolle eingebunden werden. Diese unterstützen die Fehlersuche und die Suche nach Geräten, die nicht aktualisiert wurden.

Erstellen Sie eine neue Richtlinie und weisen Sie ihr als Bereich eine intelligente Gruppe zu. Konfigurieren Sie sie als wiederkehrendes Ereignis und verknüpfen Sie sie mit dem Skriptcode, den Sie zuvor erstellt und mit einem Namen versehen haben. (Weitere Informationen zum Erstellen von Richtlinien finden Sie unter https://docs.jamf.com/10.1.0/jamf-pro/quickstart-computers/Create_a_Policy_to_Run_Software_Update.html).

While: die Bedingung, die wartet, bis ihre Zeit gekommen ist.

Eine weitere Bedingung ist die Anweisung „while“. Sie wertet nicht aus, ob eine Bedingung erfüllt ist oder nicht. Vielmehr wartet sie, bis eine Bedingung ERFÜLLT IST.

```
#!/bin/bash
while [ $( pgrep TextEdit ) ]
do
    echo Waiting for TextEdit to quit.
    sleep 2
done
echo TextEdit has quit.
```

Der Befehl `pgrep` ermittelt, ob ein bestimmter Prozess läuft oder nicht: In diesem Fall ist dies die App TextEdit. Mit den nächsten beiden Anweisungen wird der Computer angewiesen, was zu tun ist, solange TextEdit läuft:

1. `echo`-Befehl: Es wird angezeigt, dass gewartet wird, bis der Prozess beendet ist.
2. `sleep`-Befehl: zwei Sekunden warten.
3. Die „while“-Bedingung wird erneut ausgewertet.
4. Sobald TextEdit beendet wurde und die „while“-Bedingung nicht mehr zutrifft, wird die Ausführung des übrigen Skripts fortgesetzt.

Sie sehen, wie dieses Skript funktioniert, wenn Sie TextEdit öffnen, dieses Skript in Terminal ausführen und dann TextEdit beenden.

Schleifen mit der Anweisung „for“: Maßnahmen mit Dateien oder Prozessen ausführen.

Wir werden nun eine `for`-Schleife auf eine Liste von Dateien anwenden. Durch die `for`-Schleife werden alle Dateien aus der Liste umbenannt, bis die Liste komplett abgearbeitet ist.

In diesem Beispiel verwenden wir einige neue Befehle:

`cd` = change directory (Verzeichnis wechseln)

Das Zeichen `~` wird als Tilde bezeichnet. Es handelt sich um ein Kürzel für den Benutzerordner.

`ls` = list (Liste der Elemente im ausgewählten Verzeichnis erstellen)

`aFile` = Variablenname für „beliebige Datei“; hierfür kann eine beliebige Zeichenfolge verwendet werden.

„for“ definiert in diesem Fall eine Variable und liest gleichzeitig eine Variable aus. Im Klartext bedeutet dies: Gehe bei jeder Datei aus der Liste folgendermaßen vor: Zeige per `echo`-Befehl an, dass die aktuelle Datei umbenannt wird. Benenne die Datei mit dem `move`-Befehl dann tatsächlich um.

Im folgenden Beispiel stellen wir jeder Datei im Ordner „MyStuff“ den Zusatz „webinar“ voran.

Und so wird's gemacht:

```
#!/bin/bash

cd ~/Desktop/MyStuff

filelist=$( ls )

for aFile in $filelist
do
    echo Renaming file $aFile
    mv $aFile webinar-$aFile
done
```

Die Namen aller Dateien im Ordner „MyStuff“ beginnen jetzt mit „webinar-“.

Vorausschau

Keine Sorge. Wir erwarten von Ihnen nicht, dass Sie die folgenden Befehle alle kennen! Doch Sie werden wahrscheinlich überrascht sein, wie gut Sie die Grundlagen schon verstehen.

Beispiel: Die Benutzer zum Upgrade auf Mojave bewegen

Nehmen wir an, ein Administrator möchte die Benutzer dazu bewegen, auf ihrem Computer ein Upgrade auf Mojave durchzuführen. Er möchte außerdem, dass die Mac Computer automatisch aktualisiert werden, die bis zum Ablauf einer bestimmten Frist noch nicht manuell aktualisiert wurden.

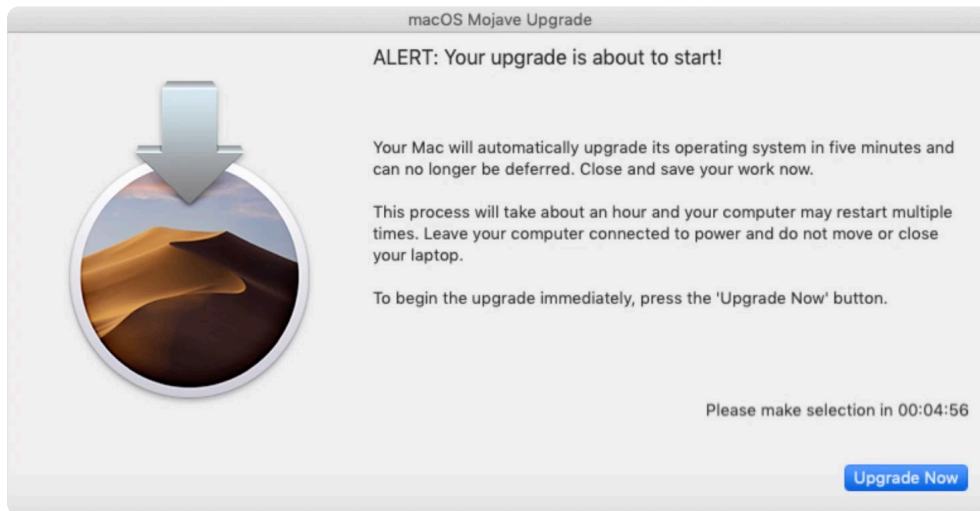
Das Skript, das nach Ablauf der Erinnerungszeit ausgeführt wird, könnte etwa folgendermaßen aussehen:

```
#!/bin/bash

"/Library/Application Support/JAMF/bin/jamfHelper.app/Contents/MacOS/jamfHelper" \
-windowType utility \
-lockHUD \
-title "macOS Mojave Upgrade" \
-heading "ALERT: Your OS upgrade is about to start!" \
-description "Your Mac will automatically upgrade its operating system in five
minutes and can no longer be deferred. Close and save your work now. The process
will take about an hour and your computer may restart multiple times. Leave your
computer connected to power and do not move or close your laptop. To begin the
upgrade immediately, select the 'Upgrade Now' button." \
-icon "/Applications/Install macOS Mojave.app/Contents/Resources/InstallAssistant.
icns" \
-iconSize 256
-button1 "Upgrade Now" \
-defaultButton 1 \
-countdown \
-timeout 300 \
-alignCountdown right

exit 0
```

Bei der Ausführung des Skripts wird Folgendes angezeigt:



Wie Sie sehen, nimmt das Dialogfenster nicht die gesamte Bildschirmfläche ein (-iconSize 256). Dadurch kann der Benutzer das Dialogfenster „wegklicken“ und seine Arbeit erst speichern. Auf der rechten Seite wird der Countdown angezeigt. Die Standardtaste trägt die Bezeichnung „Upgrade Now“.

Der folgende Prozess wird mit der Taste oder nach Ablauf des Countdowns ausgelöst:

```
"/Applications/Install macOS Mojave.app/Contents/Resources/startosinstall"  
--agreetolicense &
```

Dieses Skript ist in die meisten Installationsprogramme integriert. Bei diesem einzeiligen Befehl wird davon ausgegangen, dass Mojave bereits in dem betreffenden Ordner gespeichert ist. Durch das &-Zeichen wird der Computer neu gestartet.

Dieser Einblick in etwas komplexeren Skriptcode soll Sie nicht verschrecken. Er soll Ihnen lediglich zeigen, wie Sie mit etwas Hintergrundwissen mit Skripten viel erreichen können: Ihre Arbeitsabläufe und die Benutzerfreundlichkeit für die Anwender automatisieren und optimieren.

Er soll Ihnen aber auch Folgendes zeigen: Auch wenn Sie Skripte nicht völlig selbst geschrieben haben, können Sie diese dennoch nutzen. Sie möchten eine Aufgabe per Skript erledigen? Fragen Sie Google. Vielleicht hat schon jemand vor Ihnen die Lösung des Problems gefunden!

Damit sind wir bei den Ressourcen angelangt.

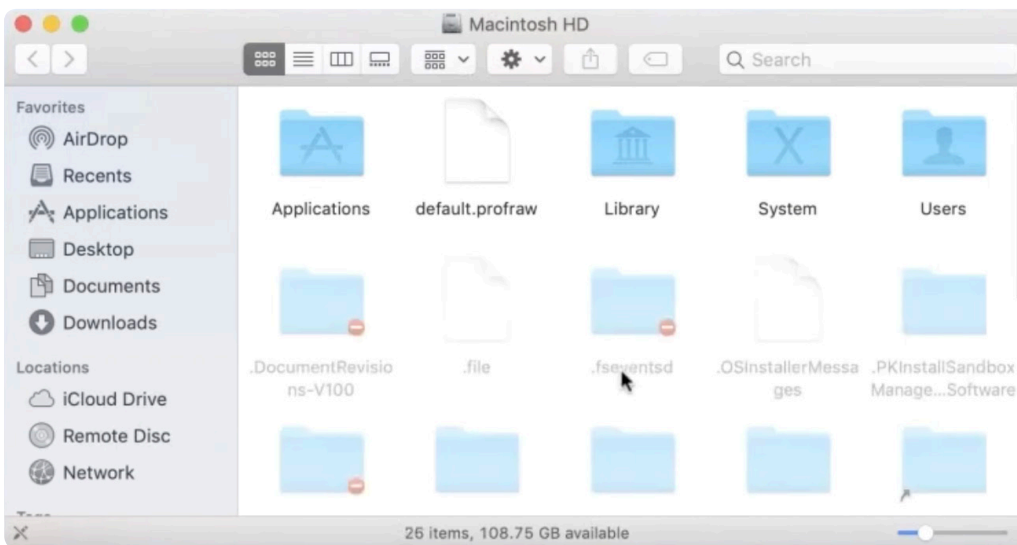
Nächster Punkt: Ressourcen

Der Mac

Wenn Sie neue Befehle erlernen möchten, steht Ihre wichtigste Ressource schon direkt vor Ihnen: Ihr Computer selbst ist eine hervorragende Informationsquelle.

Probieren Sie es aus:

Betätigen Sie in Finder folgende Tasten: Umschalttaste $\hat{}$, Befehlstaste \mathbb{C} und Punkt \cdot . Damit zeigen Sie verborgene Dateien und Ordner an. Schließlich sind die Terminal-Befehle auf dem Computer als Dateien hinterlegt. Im Computer sind alle Anweisungen zu einem bestimmten Befehl in einer separaten Datei gespeichert.



Der erste Ordner, den Sie sich anschauen sollten, ist der Ordner „bin“. Bin ist eine Abkürzung für binär. Öffnen Sie den Ordner, um den Inhalt anzuzeigen. In diesem Ordner befinden sich etwa 35 Elemente: echo, bash, mkdir. Einige kennen Sie bereits, einige sind neu.

Sie müssen nicht die Funktion aller Befehle erlernen. Überlegen Sie beim Schreiben von Skripten stattdessen, was Sie im Finder tun würden. Suchen Sie dann die Befehle, die das machen, was Sie brauchen. Google hilft Ihnen dabei! Wenn man eine Aufgabe im Finder erledigen kann, gibt es dafür auch eine entsprechende Befehlszeile.

Andere Ordner, die Sie sich anschauen können, sind „sbin“ und „usr“ (Abkürzung für Unix-Systemressourcen). Schauen Sie sich um!

Insgesamt stehen Ihnen mehr als 1000 Befehle zur Verfügung.

Schulungsvideos

Dieser Leitfaden ist eine vereinfachte Version des Webinars „Scripting for Beginners“ (in englischer Sprache). Wenn Sie durch Zuhören besser lernen, empfiehlt es sich, das Webinar anzuschauen. Weitere Lernbeispiele finden Sie auch hier (in englischer Sprache): [Scripting 101 for Apple Admins](#)

Nachdem Sie sich das Video angeschaut haben, beachten Sie auch den Blogpost in englischer Sprache, der darauf aufbaut. Er enthält Ideen für die nächsten Schritte und zeigt Ihnen, wie Sie mithilfe einer detaillierteren Ressourcenliste, die über die in diesem Leitfaden genannten Ressourcen hinausgeht, weiterlernen können:

[Just 10 Commands, Then Keep Learning](#)

[Trainingcatalog.jamf.com](https://trainingcatalog.jamf.com)

Wir bieten kurze Anleitungsvideos (Anfänger- und Fortgeschrittenenkurse) über die Skripterstellung mit einer Gesamtlänge von 15 Stunden Material. Schauen Sie sich unsere Serie über die Skripterstellung an. Diese ist für Abonnementkunden erhältlich.

Die Open-Source-Community

In der Open-Source-Community unter github.com/jamf gibt es viele Open-Source-Skripts und kostenlose Skripts, die Sie nutzen können. Sie brauchen das Rad nicht neu zu erfinden! Schauen Sie sich um, ob jemand anders schon Skripte für Ihre Aufgaben geschrieben hat. In der Jamf Nation finden Sie zudem eine hervorragende, engagierte Mac Admin Community. Die Jamf Nation ist ein unabhängiges Forum für Apple Administratoren, das von Jamf gehostet wird: <https://www.jamf.com/jamf-nation/>

Ein guter Skript-Editor

Hoffentlich konnten wir Sie davon überzeugen, einen Skript-Editor zu nutzen, damit Sie die Skriptsyntax klarer erkennen können. Hier nennen wir Ihnen eine Auswahl. Die meisten sind kostenlos.

bbedit: barebones.com (kostenlose Basisversion)

Atom: atom.io (kostenlose Basisversion)

Coderunner: coderunnerapp.com 14,99 US-Dollar zum Zeitpunkt der Drucklegung dieses Leitfadens

Wir hoffen, dass wir Ihnen mit diesem Leitfaden die Skripterstellung etwas näher bringen konnten und Sie jetzt in der Lage sind, damit zu experimentieren, zu lernen und Ihre Arbeitsabläufe zu optimieren.

Die Skripterstellung kann für Sie durchaus von Nutzen sein. Doch erst in Verbindung mit einer guten MDM-Lösung kann sie Ihnen stundenlange Arbeit mit nur einem Tastendruck abnehmen. Wir möchten Ihnen Jamf Pro empfehlen.



PRO

Testversion anfordern



www.jamf.com/de

© 2019 Jamf, LLC. Alle Rechte vorbehalten.

Gerne können Sie sich auch an einen autorisierten Händler für Apple Geräte wenden, um Jamf Pro zu testen.