

ImageMagick -

Bildbearbeitung auf der Konsole

Imagemagick ist ein hervorragendes Tool für die (gleichzeitige) Manipulation vieler digitaler Bilder.

Für wiederkehrende Tätigkeiten ist die Integration der jeweiligen Shell-Befehle in so genannte Bash-Skripts empfehlenswert.

Ein Bash-Skript ist im Grunde genommen eine Textdatei (ohne Dateiendung oder .bash). Imagemagick Anweisungen gehorchen dabei folgender Syntax:



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

```
command [options] image1 [options] image2 [options] output_image
```

input settings
operation settings
output settings

image creation operators
image modification operators
...

TYPISCHE ANWENDUNGSBEREICHE

Die ImageMagick-Programme eignen sich besonders für die folgenden Aufgabenbereiche:

- Konvertieren in andere Bildformate
- Ändern der Bildgröße
- Pixelgenau drehen, skalieren und ausschneiden
- Anwenden von Filtern und Effekten
- Erstellung von Animationen für das Web
- Erstellen von einfachen Grafiken mittels geometrischer Grundformen oder durch Kopieren von (kleineren) Grafikschnipseln
- Erzeugen von Thumbnails und Rahmen
- Mehrere Bilder zu einem Gesamtwerk zusammenstellen (etwa als Poster, Collage oder Bildergalerie)

PROGRAMMÜBERSICHT

Programm	Funktion
convert	konvertiert das Originalbild in ein anderes Format – dabei angewandte Filter und Effekte betreffen nur das Ergebnis
mogrify	ändert im Gegensatz zu <code>convert</code> das Bild direkt, Manipulation mehrerer Bilder
composite	überlagert mehrere Bilder
montage	kombiniert Bilder zu einem Panorama, einem Poster oder einer Übersicht
import	erstellt Screenshots
identify	liefert Informationen über eine Bilddatei
display	zeigt Bilder in einem Fenster an
animate	spielt eine Animation ab

Betrachten

Zum Betrachten aller `.jpg` Bilder in einem Ordner dient das Kommando

```
display *.jpg
```

Mit der Leertaste schaltet man zwischen den einzelnen Bildern durch. Das Menü kann über die linke Maustaste aufgerufen werden.



Eine **Übersicht** über den Fotobestand, die per `[Strg]+[S]` auch gespeichert werden kann, liefert der Befehl

```
display vid:*.jpg
```

Informationen

Details zu einem Bild werden über folgende Anweisung abgerufen:

```
identify [-verbose] bild.jpg
```



Mit dem Parameter `-format "%wx%h"` kann etwa nur die Angabe zur Auflösung extrahiert werden.

Paradebeispiel

Ein Paradebeispiel für den Einsatz von Imagemagick ist die Konvertierung von .jpg Fotos aus der Digitalkamera in .png Bilder bei gleichzeitiger Verkleinerung und Umbenennung.

```
for i in *.jpg; do
    convert $i ./neu/${basename $i .jpg}.png
done
```

konvertieren

basename findet den vollständigen Dateinamen der einzelnen Bilder heraus und streicht alle Pfadangaben weg. Gibt man **basename** ein weiteres Argument mit auf dem Weg, interpretiert das Kommando dies als Dateiendung, die es ebenfalls wegzustreichen gilt.

```
for i in *.png; do
    convert $i -resize 640x480 $i
done
```

verkleinern

Anstatt der pixelgenauen Auflösung kann auch eine Angabe in % erfolgen. Anstatt **-resize** kann auch **-sample**, **-geometry** oder **-thumbnail** verwendet werden. Genauso ist eine Angabe von **640x** bzw. **x480** möglich.

ImageMagick achtet bei allen Befehlen auf das richtige Seitenverhältnis beim Skalieren. Wenn dieses Verhalten geändert werden soll, muss ein Rufzeichen nach der Größenangabe folgen: z.B: **200x150!**

```
let a=0
for i in *.png; do
    let a=a+1;
    convert $i ./neu/name_.$a.png
done
```

umbenennen

und jetzt alles zusammen:

```
let a=0
for i in *.jpg; do
    let a=a+1;
    convert $i -resize 640x480 ./neu/name_.$a.png
done
```

konvertieren
verkleinern
umbenennen

Konvertieren und Verkleinern leicht gemacht

```
mogrify -format png -resize 25% *.jpg
```

Mittels `-quality 70` kann die Qualität von `.jpg` Bildern angegeben werden (Voraussetzung: `-format jpg`). 0 entspricht hierbei der schlechtesten Qualität bei bester Kompression.

Achtung: Bei `.png` Grafiken konnte dieses Verhalten nicht verifiziert werden – eine niedrigere Zahl bei `-quality` führte in den Test zu einer größeren Datei.

Drehen



Der Befehl `mogrify` manipuliert im Gegensatz zu `convert` das Originalbild! `-rotate` bewirkt eine Drehung der Bilder z.B. um 90°.

```
mogrify -rotate "90>" bild.png bzw.  
mogrify -rotate "90<" bild.png
```

Quer-/Hochformat

Screenshots erstellen



Der Befehl `import` stellt ein Fadenkreuz bereit, wodurch ein Bildschirmfoto durch Klicken in ein Fenster erzeugt werden kann – die Speicherung erfolgt im Home-Verzeichnis des aktuellen Benutzers.

```
import bild.png
```

Alternativ kann ein Fenster bzw. der gesamte Desktop gleich als Parameter übergeben werden:

```
import -window terminal bild.png  
import -window root bild.png
```

Rahmenlinie erzeugen

Folgende Anweisung bewirkt eine dünne, schwarze Rahmenlinie um das Bild.

```
mogrify -bordercolor "#ff0000" -border 1x1 bild.png
```

Rahmen

Rahmen aus Bild

Durch unterschiedlich farbige Ränder kann ein Bild scheinbar etwas aus der Ebene hervorstehen (`-raise`) bzw. darunterliegen (`+raise`).

```
convert alt.png -raise 15x15 neu.png
convert alt.png +raise 15x15 neu.png
```

Anspruchsvolle Rahmen

Mit der Kombination `-frame` und `-mattecolor` können anspruchsvollere Rahmen erzeugt werden.

```
convert -mattecolor black alt.png -frame 10x10+5 neu.png
convert -mattecolor black alt.png -frame 10x10+5+5 neu.png
convert -mattecolor gray alt.png -frame 15x15+0+15 neu.png
convert -mattecolor gray alt.png -frame 15x15+15+0 neu.png
```

Die Syntax von `-frame` gehorcht dabei folgender Regel:

Rahmenbreite x Rahmenhöhe + äußererSchatten + innererSchatten

Schatten

Der erste Parameter von `-shadow` ist für die Transparenz der Schattenfarbe verantwortlich, der zweite Wert bewirkt einen sanfteren Schatten und die letzten beiden Angaben sind für den XY-Versatz zuständig.

```
convert alt.png \(+clone -background black -shadow 80x0+5+5\)
+swap -background none -mosaic neu.png
```

Tiefe

Diese Methode erzeugt einen 3D Eindruck.

```
convert -alt.png \
\(+clone -fill DarkSlateGrey -colorize 100% -repage +0+1\) \
\(+clone -repage +1+2\) \
\(+clone -repage +1+3\) \
\(+clone -repage +2+4\) \
\(+clone -repage +2+5\) \
\(+clone -repage +3+6\) \
-background none -compose DstOver -mosaic neu.png
```

Weicher, ovaler Rahmen

Interessante Effekte können vor allem durch Verändern des zweiten Parameters von `-vignette` erzielt werden – die vollständige Syntax lautet:
Radius x Sigma ± Versatz X ± Versatz Y

```
convert alt.png -background none -vignette 0x20 neu.png
```

Weicher, nicht rein ovaler Rahmen

Folgende Methode legt transparente Pixeln über die Ränder und Ecken des Bildes. Unterschiedliche Effekte können auch hier am besten durch Verstellen des zweiten Parameters von `-blur` erreicht werden. Interessant ist auch ein Vertauschen der Parameter von `-level`.

```
convert alt.png -alpha set -virtual-pixel transparent -channel A  
-blur 0x20 -level 0,50% neu.png
```

Runde Ecken

Das Ergebnis weist in dieser Form leicht ausgefranste Ecken aus:

```
convert alt.png -alpha set -virtual-pixel transparent -channel A  
-blur 0x20 -threshold 50% neu.png
```

Komplexere Anwendungen

Hier sei auf www.imagemagick.org/Usage/thumbnails verwiesen (Rounded Corners, Torn Paper Edge).

Polaroid

Die zusätzlich Option `+polaroid` (+ ist Standardeinstellung mit zufälliger Drehung) in Kombination mit `-bordercolor` (Umrandung) und `-background` (Schatten) manipuliert das Bild so, dass es wie ein eingeklebtes, sich leicht wellendes Foto aussieht.

```
mogrify -bordercolor white -background black +polaroid bild.png
```

`-polaroid` ermöglicht die Angabe des Drehwinkels und `-thumbnail` wirkt wie `-resize`, nur werden hierbei Bildinformationen (Kommentare, ...) gelöscht.

```
convert -bordercolor AliceBlue -background SteelBlue4 alt.png  
-thumbnail 120x120 -polaroid 5 neu.png
```

Eine Beschriftung kann über `-caption` (Auslesen der Metadaten, Vergabe eines individuellen Titels) erreicht werden.

```
convert -caption „Beschriftung“ -font Courier -pointsize 18
```

```
-bordercolor Snow -background black -fill dodgerblue -stroke navy -gravity center alt.png -thumbnail 120x120 -polaroid 5 neu.png
```

Thumbnails

```
montage -label '%f\n%wx%h' [-tile x1] [-geometry '200x150+5+5'] [-frame 5] [-background transparent] *.png uebersicht.png
```

Dieser Befehl legt um jedes Vorschaubild einen **Rahmen**, schreibt den **Dateinamen** samt **Auflösung** darunter und packt schließlich die fertigen Grafiken in eine Datei. Mittels `-tile x{Zahl}` kann die Anzahl der Zeilen festgelegt werden.

Variante mit eigener Beschriftung

```
montage -title 'Hier steht ein Text' -pointsize 8 -label Titel1 Datei1.png -label Titel2 Datei2.png -label3 Titel3 Datei1.png -tile x2 -geometry '200x150+5+5' -frame 5 -background transparent -shadow uebersicht.png
```

Über das Attribut `-title` kann den Thumbnails eine Überschrift verpasst werden. Mittels `-pointsize` kann die Schriftgröße verändert werden – dies ist vor allem bei langen Dateinamen empfehlenswert. `-shadow` erzeugt einen Schatten rund um die einzelnen Thumbnails.

HTML Thumbnails

```
montage ... uebersicht.html
```

Hierbei werden 3 Dateien erzeugt, wobei die Datei `vid_index_map.shtml` gelöscht werden kann.

HTML Thumbnails mit Polaroid Effekt - 1 (komplex, ohne exakte Erklärung)

```
montage -caption '%t' -bordercolor AliceBlue -background grey20 -geometry +1+1 -tile 5x './*.png' -thumbnail 128x128 +polaroid -background white \ polaroid_index.html
```

`-bordercolor` regelt die Farbe der Umrahmung und das erste `-background` die Farbe des Schattens. Um zwischen den Bildern wieder eine weiße Farbe zu erhalten, muss das Attribut `-background` anschließend zurück gesetzt werden.

HTML Thumbnails mit Polaroid Effekt - 2 (komplex, ohne exakte Erklärung)

```
montage -bordercolor Lavender -background black -geometry -10+2  
-tile x1 null: './*.png' null: -thumbnail 200x200 +polaroid  
-resize 30% -background LightGray polaroid2_index.html
```

Alternative mittels convert und visual directories

```
convert 'vid:../fotos/*' vid_index.png bzw.  
convert 'vid:../fotos/*' vid_index.html
```

Dabei sollte beachtet werden, dass die Attribute `-tile`, `-frame` und `-shadow` andere Parameter erwarten als dies bei montage der Fall ist.

Filter



Scharfzeichnen:

```
convert alt.png -sharpen 5 neu.png
```

Aufhellen:

```
convert alt.png -sigmoidal-contrast 4,0% neu.png
```

Graustufen u. S/W:

```
convert alt.png -colorspace gray neu.png
```

Achtung: Hier muss das zu bearbeitende Bild zuerst angegeben werden!

```
convert -channel R alt.png -separate neu.png
```

```
convert alt.png -monochrome grau.png
```

Kohlezeichnung:

```
convert alt.png -charcoal 2 neu.png
```

Der Wert hinter `-charcoal` bestimmt dabei die Härte des Kohlestiftes.

Negativ:

```
convert alt.png -solarize 0% neu.png
```

Hiermit kann ein Negativ wie bei der analogen Fotografie erzeugt werden.

Strudel:

```
convert alt.png -swirl 90 neu.png
```

`-swirl` verdreht Pixel um eine Gradangabe um das Zentrum des Bildes.

Implodieren

```
convert alt.png -implode 0.5 neu.png
```

`-implode` implodiert Pixel um einen Faktor um das Zentrum des Bildes.

Wasserzeichen

```
composite [-gravity SouthEast] -watermark 30% [-geometry +50+50]  
logo.png alt.png neu.png
```

Der Befehl `composite` kopiert das Bild aus `logo.png` in jene Ecke, die über `-gravity` als Bezugspunkt angegeben wurde. Mittels `-geometry +50+50` kann zusätzlich eine pixelgenaue Positionierung bezogen auf den jeweiligen Bezugspunkt erreicht werden. Die Prozentangabe nach `-watermark` bewirkt ein Durchscheinen des Wasserzeichens.

Beschriftung

Mit dem Befehl `locate *.ttf | grep Name-der-Schrift` läßt sich der exakte Pfad zur gewünschten Schrift ermitteln, die im Argument `-font` angegeben werden muss.

```
convert  
-font @/usr/share/fonts/truetype/msttcorefonts/Verdana.ttf \  
-pointsize 20 \  
-fill "#ff0000" \  
alt.png \  
alt.png
```

```
-draw "text 50, 100 'Hier steht der Text'" \  
neu.png
```

Der Übersichtlichkeit halber ist obiges Listing zeilenweise gegliedert.

`-pointsize` gibt die Schriftgröße der unter `-font` gewählten Schriftart an. Mittels `-fill` kann eine Schriftfarbe definiert werden und `-draw` ist schließlich für die tatsächliche Beschriftung zuständig. Die beiden Werte geben den x- bzw. y-Abstand vom gerade aktiven Bezugspunkt (Änderung per `-gravity` möglich).

Text mit Schatten

```
convert  
-font @/usr/share/fonts/truetype/msttcorefonts/Verdana.ttf \  
-pointsize 50 \  
alt.png \  
-fill black \  
-annotate +50+100 'Hier steht der Text' \  
-fill gray \  
-annotate 0x130+50+100 'Hier steht der Text' \  
neu.png
```

Ein Schatteneffekt wird durch das versetzte Einfügen einer Beschriftung mit unterschiedlichen Farben erzeugt. Der Befehl `-annotate` ist eine vereinfachte Variante zum Einfügen von Text mit weniger Einstellungsmöglichkeiten als `-draw`. Die Werte `xxy` bewirken einen Dreheffekt.

Text mit einfacher Rahmenlinie

```
convert  
-font @/usr/share/fonts/truetype/msttcorefonts/Verdana.ttf \  
-pointsize 50 \  
-fill white \  
-stroke black \  
-strokewidth 1 \  
alt.png \  
-annotate +50+100 'Hier steht der Text' \  
neu.png
```

`-stroke` zieht eine Rahmenlinie um den Text, die Linienstärke kann mit `-strokewidth` angegeben werden.

Text mit fortgeschrittenem Schatten

```
convert
-font @/usr/share/fonts/truetype/msttcorefonts/Verdana.ttf \
-pointsize 50 \
-fill white \
alt.png \
-stroke black \
-strokewidth 5 \
-annotate +50+100 'Hier steht der Text' \
-stroke none \
-strokewidth 0 \
-annotate +50+100 'Hier steht der Text' \
neu.png
```

Dieses Beispiel liefert eine leicht versetzte schattige Umrandung.

Durchsichtiger Text

```
convert
-font @/usr/share/fonts/truetype/msttcorefonts/Verdana.ttf \
-pointsize 50 \
-fill none \
-stroke black \
-strokewidth 1 \
alt.png \
-annotate +50+100 'Hier steht der Text' \
neu.png
```

Dieses Beispiel generiert eine durchsichtige Schrift mit einer dünnen, schwarzen Rahmenlinie.

Text mit doppelter Rahmenlinie

```
convert
-font @/usr/share/fonts/truetype/msttcorefonts/Verdana.ttf \
-pointsize 50 \
alt.png \
-fill none \
-stroke black \
-strokewidth 3 \
-annotate +50+100 'Hier steht der Text' \
-fill none \
-stroke white \
-strokewidth 1 \
-annotate +50+100 'Hier steht der Text' \
neu.png
```

Dieses Beispiel erzeugt eine doppelte schwarze Rahmenlinie mit weißer Füllung.

Text mit Mahjongg-Effekt

```
convert
-font @/usr/share/fonts/truetype/msttcorefonts/Verdana.ttf \
-pointsize 50 \
-fill white \
alt.png \
-stroke black \
-strokewidth 25 \
-annotate +50+100 'Hier steht der Text' \
-stroke white \
-strokewidth 20 \
-annotate +50+100 'Hier steht der Text' \
-stroke black \
-strokewidth 15 \
-annotate +50+100 'Hier steht der Text' \
-stroke white \
-strokewidth 10 \
-annotate +50+100 'Hier steht der Text' \
-stroke black \
-strokewidth 5 \
-annotate +50+100 'Hier steht der Text' \
-stroke none \
-strokewidth 0 \
-annotate +50+100 'Hier steht der Text' \
neu.png
```

abc

abc