Aufgabenblatt 34

Aufgabe 1

In dieser Aufgabe sollt ihr ein Makefile schreiben, welches das Verarbeiten von Bildern automatisiert. Es sollen alle Bilder mit der Endung **.jpg** oder **.tga** in das Netpbm-Format (mit der Dateiendung **.ppm**) umgewandelt und
skaliert werden.
Anschließend sollen die skalierten Bilder in das PNG-Format umgewandelt und zusammen in ein TAR-Archiv gepackt werden.

help-Target

Bei einem Aufruf von make help soll der folgende Hilfetext ausgegeben werden:

- Makefile to scale and convert jpg and tga files to png.
- VARIABLES
  - SIZE - specifies the largest image size (default=100)
- TARGETS
  - all - create tar archive and png files
  - tar - create tar archive
  - png - scale and convert all jpg and tga files to png
  - help - display this help and exit
  - clean - remove all generated files

png-Target

Mit dem Target png (bei einem Aufruf von make png) sollen alle Dateien im aktuellen Verzeichnis mit der Endung .jpg oder
.tga zunächst in das Portable Pixmap Format mit der Endung .ppm konvertiert werden. Hierbei sollen der Dateipfad und
Dateiname gleich bleiben und sich nur die Dateiendung ändern.

Dies kann mit dem Programm jpegtopnm bzw. tgatoppm aus dem netpbm-Paket erreicht werden.

Anschließend werden die .ppm Dateien auf eine Größe von SIZE x SIZE (siehe unten) skaliert und als neue Dateien mit
der Endung ***.scaled.ppm*** gespeichert. Hierbei ist zu beachten, dass bei nicht-quadratischen Bildern die längere Seite auf
die angegebene Länge skaliert wird und das ursprüngliche Seitenverhältnis des Bildes erhalten bleibt.

Dies kann mit dem Programm pnmscale (ebenfalls aus dem netpbm-Paket) erreicht werden. (Hinweis: Je nach Version des
netpbm-Pakets kann pnmscale auch pamscale heissen, orientiert euch dabei an der Version die im Rechenzentrum
installiert ist: Netpbm 10.0)

Zuletzt sollen die erzeugten ***.scaled.ppm*** Bilder in das PNG-Format konvertiert und mit der einfachen Endung ***.png***

gespeichert werden.

Dies kann mit dem Programm pnmtopng erreicht werden.

tar-Target

Bei einem Aufruf von make tar sollen alle erzeugbaren PNG-Dateien zusammen in ein TAR-Archiv im aktuellen Verzeichnis
mit dem Name ***png-files.tar.gz*** gepackt werden. Sollten diese noch nicht existieren, müssen sie vorher erzeugt
werden. Das Target tar soll nur ein “alias” für das ***png-files.tar.gz-Target*** sein (make ***png-files.tar.gz*** muss
ebenfalls funktionieren).

all-Target

Bei einem Aufruf von make all oder einfach nur make sollen sowohl die PNG-Dateien als auch das TAR-Archiv erzeugt werden.

clean-Target

Mit dem Target clean sollen alle duch make erzeugten Dateien wieder gelöscht werden. Hierzu zählen nebend den
Zwischenformaten (PPM) auch die PNG-Datei sowie das TAR-Archiv. Die Quelldateien (JPG und TGA) sollen in jedem Fall
erhalten bleiben.

%Regeln

Verwendet für die Konvertierung der Dateiformate die %-Regeln, sodass eine beliebige Datei auch mittels make
beliebige-datei.png oder make ***beliebige-datei.scaled.ppm*** erzeugt werden kann.

Aufräumen

Je nach Aufbau des Makefiles erkennt make temporäre Dateien (in unserem Fall .ppm Dateien) und löscht diese einfach
direkt wieder.
Da wir diese Dateien aber auch haben wollen (zumindest bis make clean wieder aufgerufen wird), müssen wir make dies
ggf. mit Hilfe des .SECONDARY Targets mitteilen (siehe auch: Chains of Implicit Rules).
Fügt daher einfach folgende Zeile in euer Makefile ein (damit wird die Liste der temporären Dateien als “leer” definiert):.SECONDARY:

Skalieren

Die Skalierung der Bilder soll mit Hilfe der Variablen SIZE konfigurierbar sein. SIZE gibt dabei die maximale Höhe/Breite der Zielbilder in Pixel an.
Wird make beispielsweise mit make png SIZE=250 aufgerufen, sollen die Bilder entsprechend auf eine maximale Größe
von 250 Pixel skaliert werden. Ist beim Aufruf von make kein anderer Wert für SIZE angegeben, wird der Default-Wert
verwendet: 100.

Sonstiges

Wir gehen für diese Aufgabe davon aus, dass alle Eingabe-Dateien die korrekte Dateiendung besitzen. D.h.: Alle JPG-Dateien enden mit .jpg und alle TGA Dateien mit .tga.

Umfang der Lösung

Diese Aufgabe ist mit einem 60 Zeilen langen Makefile problemlos lösbar (reines Make, keine Skriptsprachen). Sollte eure Lösung also viel umfangreicher sein, solltet ihr diese nochmal überarbeiten. Lösungen, welche mehr als 90 Zeilen lang sind (ohne Kommentare), werden wir nicht akzeptieren.

Erlaubte Tools

Ihr sollt in dieser Aufgabe ein Makefile entwickeln, daher sind Shell-Befehle nur für die Implementierung der einzelnen Targets erlaubt. Euer Programm soll im Wesentlichen ein Makefile sein. Das Finden der Quelldateien soll daher ebenfalls mit Make umgesetzt werden. Aufrufe von find oder ähnlichem sind daher verboten.
Für diese Aufgabe werden die netpbm-Tools benötigt, welche unter anderem Bilder konvertieren können.
Unter Ubuntu können diese mittels apt-get install netpbm installiert werden. In den Rechenzentren ist dies bereits geschehen.
Make kann mit dem Befehl apt-get install make installiert werden. Auch make ist in den Rechenzentren schon installiert.
Zum Betrachten und manuellen Bearbeiten von Bilder eignen sich beispielsweise display oder gimp.

Hinweise

Makefiles

Targets

In Makefiles werden Abhängigkeit über Targets definiert.
Ein Beispiel:

```
mytarget: mysource
command
```

Dieses Target besagt, dass die Datei mytarget abhängig von der Datei mysource ist und durch den Befehl commad
erzeugt werden kann. Üblicherweise wird im Befehl command die Datei mysource gelesen und die Datei mytarget
geschrieben.

Dabei wird vorher überprüft ob die Datei mytarget älter ist als mysource. Nur wenn dies der Fall ist, wird das Kommando tatsächlich ausgeführt.

Diese Abhängigkeiten können auch beliebig verkettet werden, z.B.:

b: a
cmd1
c: b
cmd2
d: c

cmd3

Um das Target d zu erzeugen würde zunächst cmd1, dann cmd2 und schließlich cmd3 ausgeführt werden, wobei jedesmal zunächst die oben genannte Überprüfung stattfindet.

Stringverarbeitung

Um alle Dateiennamen, die auf ein bestimmtes Muster passen in einer Variablen zu speichern, genügt folgende Zeile:
`SRC=$(wildcard *.c)`
Hieraus kann leicht eine Variable erstellt werden, die die gleichen Dateinamen mit anderer Endung enthält:
`OBJ=$(SRC:.c=.o)`

Shell

Die Variable SHELL definiert die in make genutzte Konsole. Manchmal ist es notwendig die Bash zu verwenden, dies sollte im Makefile dann explizit angegeben werden:
`SHELL=/bin/bash`

`Hier könnte deine Lösung stehen`