



25. JANUAR 2022

LINUS DUTTWEILER

BASH- PROJEKT

| | |
|----------------------|---|
| Projektanforderungen | 2 |
| Testing | 3 |
| Testarten | 3 |
| Testprotokoll | 3 |
| Tests | 3 |
| Realisierung | 5 |
| Benutzung | 5 |
| Pfadformat | 5 |
| Parameter | 6 |
| Screenshots | 6 |

PROJEKTANFORDERUNGEN

Ich nutze zuhause und im Geschäft Linux.

Dort habe ich gewisse Konfigurationsdateien und weitere Dateien welche ich bei beiden Maschinen brauche. z.B .vimrc oder .bashrc und viele mehr.

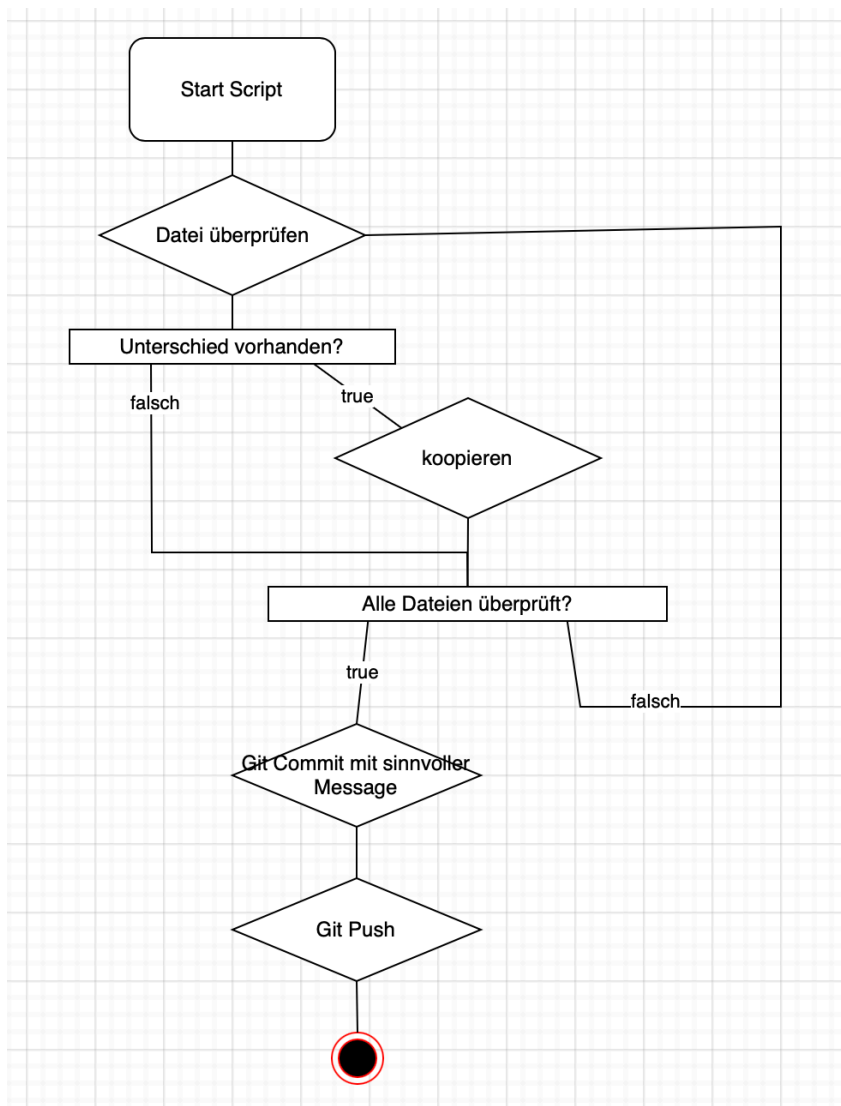
Ich möchte ein Script erstellen, mit welchem ich die Dateien Synchroneren kann.

Dabei soll das Script alle angegebenen Dateien in ein Unterordner im Home-Verzeichnis kopieren und Git-Commits erstellen. Diese werden dann auch auf ein Repository gepushed.

Somit kann ich dann die Dateien von zuhause alle in ein Repo pushen und im Geschäft auf sie zugreifen.

Die Commits und das Pushen soll auch selbständig gemacht werden.

Wenn ich genug weit komme, kann ich auch schauen, ob ich implementieren kann, dass das Script automatisch ausgeführt wird.



TESTING

Testarten

Ich kenne für Bash-Scripts nur wirklich die Möglichkeit sie von Hand zu testen. Das heisst, ich werde selbst das Script durch einige Testversuche untersuchen um sicherzustellen, dass die gesamte Funktionalität implementiert ist.

Testprotokoll

| Testfall | Beschreibung |
|-----------|--|
| Standard | Das Programm liest die Pfade aus dem File und kopiert ALLE angegebenen Dateien. Es reproduziert die Ordnerstruktur im aktuellen Verzeichnis. |
| Diff | Unterschiede in den Dateien werden erkannt. Bzw eine Veränderte Datei wird beim zweiten Durchlauf ersetzt. |
| Git | Die Commits werden erstellt, wenn das Programm laufen gelassen wird & es wird pushed. |
| Parameter | Die Parameter werden korrekt interpretiert. |

Tests

| Testfall | Beschreibung | Resultat |
|----------|--|---|
| Standard | Das Programm liest die Pfade aus dem File und kopiert ALLE angegebenen Dateien. Es reproduziert die Ordnerstruktur im aktuellen Verzeichnis. | Alles funktioniert wie erwartet, wird das Projekt in einem passenden Umfeld laufen gelassen. Es kopiert alle angegebenen Dateien in die neue Struktur im aktuellen Verzeichnis. |
| Diff | Unterschiede in den Dateien werden erkannt. Bzw eine Veränderte Datei wird beim zweiten Durchlauf ersetzt. | Diff erkennt die Unterschiede und die Dateien werden ersetzt, wird festgestellt wird das Programm ein zweites mal laufen gelassen, nach dem verändern eines Dokumentes. |
| Git | Die Commits werden erstellt, wenn das Programm laufen gelassen wird & es wird pushed. | Git funktioniert, jedoch musste ich es auskommentieren, da ich beim Entwickeln das Programm auch in einem Git-Repository habe. IN dieses soll es nicht pushen. Man muss die Zeilen auskommentiert, nach dem man das Script in einen anderen Ordner kopiert. |

| Testfall | Beschreibung | Resultat |
|-----------|---|--|
| Parameter | Die Parameter werden korrekt interpretiert. | Die Parameter werden gut geparsed und brechen das Programm ab, werden sie nicht erkannt. |

REALISIERUNG

Das Script kopiert die angegebenen Files, wenn sie noch nicht enthalten sind oder verändert wurden.

Die Orderstruktur wird dabei im aktuellen Verzeichnis abgebildet.

Heisst: kopiere ich etwas aus .config/temp wird .config/temp im Verzeichnis erstellt.

Dies ist eine gewollte Design Entscheidung.

Ich möchte, dass es eine Abbildung vom System ist, damit ich dann später weiss, von wo die Datei auf dem anderen Computer kam.

Benutzung

Das Script sollte in einem eigenen Ordner platziert werden.

Am bestem im User Verzeichnis: z.B /home/USER.

Der Ordner muss ein Git-Repository mit Upstream enthalten.

Das Script benötigt eine Datei, aus welcher es die Pfade lesen kann.

Die Datei muss einem Format Endsprechen. In meiner Abgabe ist ein file „paths“ als Beispiel.

Pfadformat

Pfad

- Dateiname

Beispiel

Wenn ich also eine Datei aus den Downloads und mehrere aus Documents möchte, notiert man das so:

~/Downloads

- downloaded_file.txt

~/Documents

- temp.cfg

- temp.txt

Das Programm sucht dann nach ~/Downloads/downloaded_file.txt, ~/Documents/temp.cfg und ~/Documents/temp.txt.

Parameter

Das Script hat zwei Parameter Optionen.

-v/-verbose und ein Pfad für das File, welches die weiteren Pfade mit dem Format enthält. Wird kein Pfad angegeben, nutzt sucht das Script die Datei im aktuellen Verzeichnis unter dem Namen „paths“.

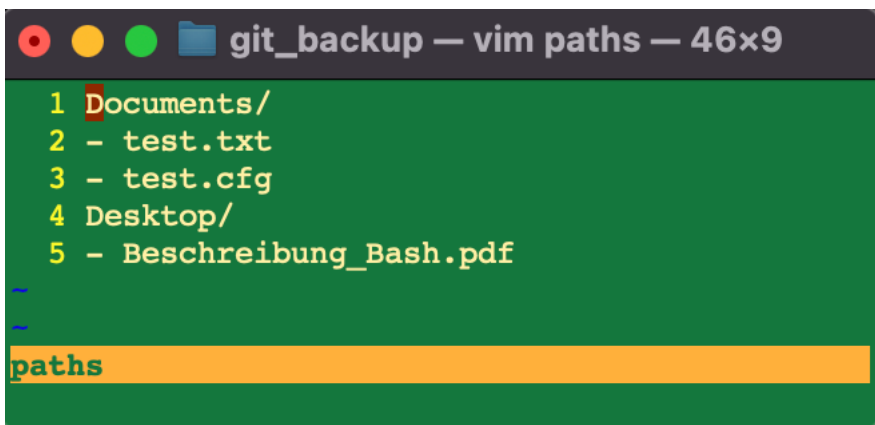
BEISPIEL

`./update -v` //Verbose

`./update -v /usr/temp/datei` //Verbose + Nutzt eigenen Pfad fuer Pfad-Dokument

`./update /usr/temp/datei` //Nutzt eigenen Pfad fuer Pfad-Dokument

Screenshots

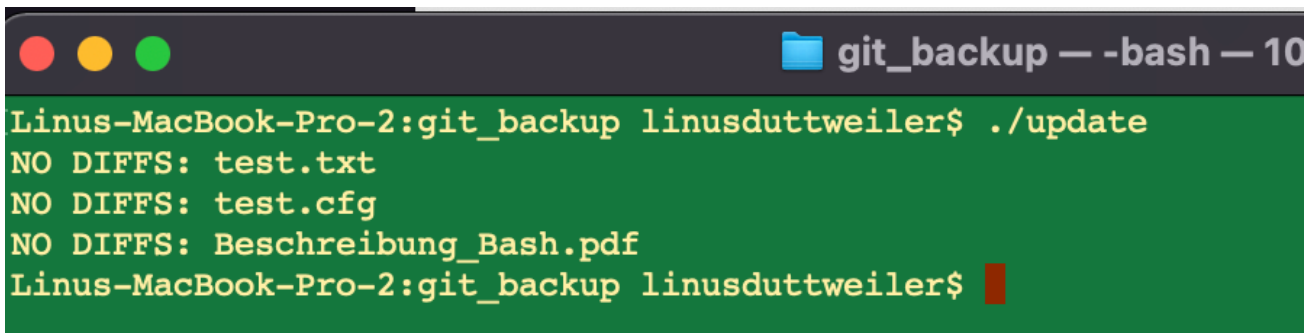


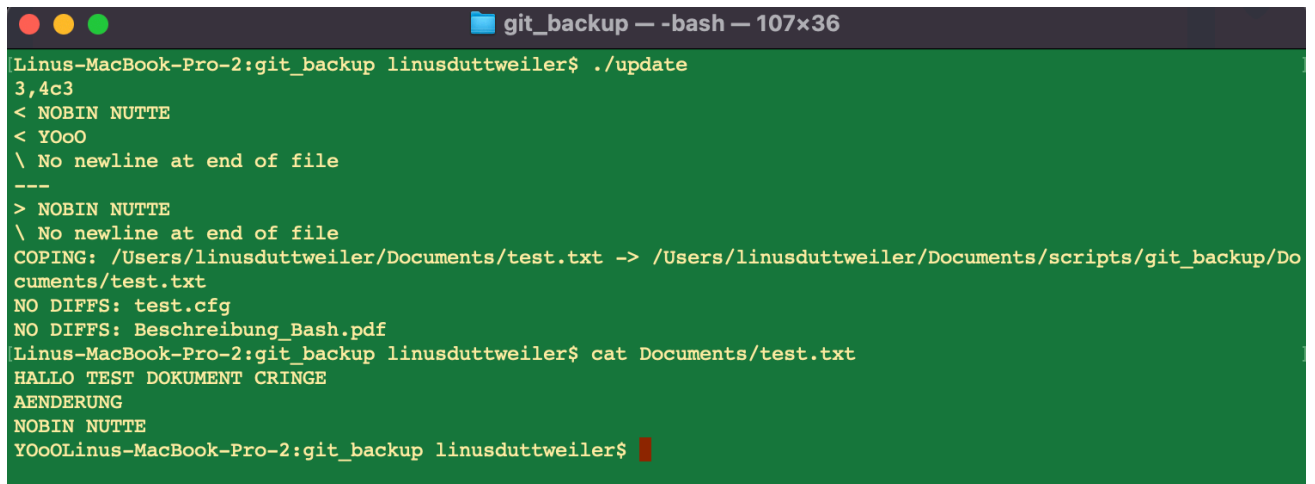
Das File in dem die Pfade mit meiner Formatierung definiert sind.

Diese Pfade werden dann im /home/user Verzeichnis gesucht: ~/Documents/text.txt

Und vergleicht es mit \$PWD/Documents/test.txt

Das Programm findet keinen Unterschied, da die Dokumente nicht verändert wurden.



A terminal window titled 'git_backup -- -bash -- 107x36' on a dark background. The text is yellow. It shows the output of a './update' command. The output includes file statistics (3,4c3), a comparison of 'NOBIN NUTTE' between two versions, a warning about a missing newline, a copy command for 'test.txt', and a list of files that are not different (test.cfg, Beschreibung_Bash.pdf). It then shows the content of 'Documents/test.txt' which is 'HALLO TEST DOKUMENT CRINGE' followed by 'AENDERUNG' and 'NOBIN NUTTE'. The prompt returns to 'Linus-MacBook-Pro-2:git_backup linusduttweiler\$' with a red cursor.

```
Linus-MacBook-Pro-2:git_backup linusduttweiler$ ./update
3,4c3
< NOBIN NUTTE
< Y0oO
\ No newline at end of file
---
> NOBIN NUTTE
\ No newline at end of file
COPING: /Users/linusduttweiler/Documents/test.txt -> /Users/linusduttweiler/Documents/scripts/git_backup/Do
cuments/test.txt
NO DIFFS: test.cfg
NO DIFFS: Beschreibung_Bash.pdf
Linus-MacBook-Pro-2:git_backup linusduttweiler$ cat Documents/test.txt
HALLO TEST DOKUMENT CRINGE
AENDERUNG
NOBIN NUTTE
Y0oOLinus-MacBook-Pro-2:git_backup linusduttweiler$
```

Nun verändere ich ~/Documents/text.txt und lasse das Programm wieder laufen.

Das Programm erkennt nun einen Unterschied und die ändern ist im \$PWD/Documents/test.txt enthalten.