



Versionsverwaltung mit Git

Kurze Einführung am Beispiel von LaTeX

E. Frank Sandig



Versionsverwaltung mit Git

WORUM ES HEUTE GEHT

Motivation

Geschichte

Begriffe

Eigenschaften

Beispiele

Stoff zum Lesen



MOTIVATION

Größere Projekte:

- Verfolgen der Projektgeschichte
- Zurücknehmen von Änderungen
- verteiltes, kollaboratives Arbeiten am Projekt
- effiziente Verwaltung
- unkomplizierte Veröffentlichung des Quelltextes

└ Motivation

└ Motivation

MOTIVATION

Größere Projekte:

- Verfolgen der Projektgeschichte
- Zurücknehmen von Änderungen
- verteiltes, kollaboratives Arbeiten am Projekt
- effiziente Verwaltung
- unkomplizierte Veröffentlichung des Quelltextes

Mehr als drei Dateien und mehr als 1000 Worte.



NAMENSHERKUNFT

“I’m an egotistical bastard, and I name all my projects after myself. First ‘Linux’, now ‘Git’.”

Linus Torvalds

“The joke ‘I name all my projects for myself, first Linux, then git’ was just too good to pass up. But it is also short, easy-to-say, and type on a standard keyboard. And reasonably unique and not any standard command, which is unusual.”

Linus Torvalds



ENTSTEHUNG

- 2005 von Linus Torvalds gestartet
- Anlass: bis dahin für Linux (Kernel) verwendete Versionsverwaltung BitKeeper bekam neue Lizenz, es wären Gebühren angefallen
- Hauptentwickler heute: Junio Hamano
- Anforderungen:
 1. verteilte Arbeitsweise, Abläufe ähnlich BitKeeper
 2. sehr hohe Sicherheit gegen unbeabsichtigte und böswillige Verfälschung
 3. hohe Effizienz
- Punkt 3 wurde nach Torvalds' Ansicht vom bereits bestehenden Projekt Monotone nicht erfüllt

ENTSTEHUNG

- 2005 von Linus Torvalds gestartet
- Anlass: bis dahin für Linux (Kernel) verwendete Versionsverwaltung BitKeeper bekam neue Lizenz, es wären Gebühren angefallen
- Hauptentwickler heute: Junio Hamano
- Anforderungen:
 1. verteilte Arbeitsweise, Abläufe ähnlich BitKeeper
 2. sehr hohe Sicherheit gegen unbeabsichtigte und böswillige Verfälschung
 3. hohe Effizienz
- Punkt 3 wurde nach Torvalds' Ansicht vom bereits bestehenden Projekt Monotone nicht erfüllt

Arten der Effizienz:

- effiziente Befehle (wenig bewirkt viel)
- effiziente Speicherung (geringer Overhead der Verwaltung); Commits, Branches, ...
- effiziente interne Arbeitsweise (schnelle Ausführung)



ALTERNATIVEN

verteilte Versionsverwaltung:

- Monotone
- BitKeeper
- Mercurial
- Bazaar
- GNU arch

zentrale Versionsverwaltung:

- Apache Subversion (svn)

└─ Geschichte

└─ Alternativen

ALTERNATIVEN

verteilte Versionsverwaltung:

- Monotone
- BitKeeper
- Mercurial
- Bazaar
- GNU arch

zentrale Versionsverwaltung:

- Apache Subversion (svn)

auch: lokale Versionsverwaltung, eher selten



VERWENDUNG

zahlreiche Opensource-Projekte, z. B.:

Amarok, **Android**, BusyBox, CMake, **Debian**, DragonFly BSD, Drupal, Eclipse, Erlang, Fedora, **Git** selbst, **Gnome**, Joomla, jQuery, JUnit, KDE, **LibreOffice**, LilyPond, **Linux-Kernel**, Linux Mint, MediaWiki, node.js, One Laptop per Child, OpenFOAM, Perl 5, Parrot und Rakudo (Perl 6), **PHP**, phpBB, Plone, PostgreSQL, Qt, Ruby on Rails, Ruby, Samba, Scala, TaskJuggler, TYPO3, **VLC** media player, Wine, x264 und **X.org** uvm.

neu Nov. 2014: GNU **Emacs**, Google Go



BEGRIFFE

Textdatei

Eine Datei, welche unformatierten Text im ASCII-Format (oder kompatibel), also eine direkt druckbare Zeichenfolge, enthält. Insbesondere existieren Zeilenumbrüche. Bei L^AT_EX primär der Quelltext, ferner auch .bib-Datenbanken, Hilfs- und Logdateien und ähnliches. Menschenlesbar.

Binärdatei

Maschinenlesbare Datei. Benötigt zur Anzeige für den Menschen einen Interpreter (fertige PDF-Datei) oder ist selbst eine ausführbare Datei (.exe). Weiter gefasst alles, was kein reiner Text ist.

Beide Arten werden von Git unterschiedlich behandelt: → Commit.



BEGRIFFE II

Repository

Lager, Depot. Übertragen: Projektarchiv. Verwaltetes Verzeichnis zur Speicherung und Beschreibung von digitalen Objekten, also z. B. Ordner mit Dateien + Verwaltungsinformationen hinter einer Infrastruktur. Der Speicherort kann lokal oder entfernt sein. Der Befehl `git init` macht einen Ordner durch erstellen der Verwaltungsinformationen zum Repository.



BEGRIFFE III

Fork

Gabel. Übertragen: Abspaltung. Vollständige Kopie eines Projektes, welche unabhängig weiterentwickelt werden kann. Insbesondere mindestens ein vollständiges Repository mit untergliederten → Branches, welches die gesamte Projektgeschichte enthält. Der Fork kann dem ursprünglichen Projekt über → Pull Requests und/oder einen → Upstream Branch verbunden bleiben. Forks sind im Rahmen verteilter entwickelter Projekte ein reguläres Arbeitsmittel (Fork = Arbeitskopie). Linus Torvalds' Betriebssystemkern Linux wurde auf GitHub über 7500 Mal (01/2015) geforkt, in der Regel, um Änderungen zum Projekt beizutragen, nicht, um ein neues Projekt zu gründen.



BEGRIFFE IV

Branch

Zweig. Jede verwaltete Ableitung des Arbeitsverzeichnisses ist technisch ein Branch. In der Regel eine Abspaltung innerhalb des Projektes, z.B. zum Erstellen einer neuen Hauptversion oder zur getrennten Entwicklung einer bestimmten Funktion. In großen L^AT_EX-Projekten können einzelne Kapitel in getrennten Zweigen entwickelt werden. Zweige eines Projektes können verschmolzen werden: → Merge.



BEGRIFFE V

Remote

Entferntes Repository. Dies kann ein Remote Tracking Branch zur Veröffentlichung und Sicherung des Quelltextes sein (lesen und schreiben) oder ein → Upstream Repository (nur lesen). Ein Remote ist immer auch ein Branch.

Commit

Begehen, einprägen. Übertragen: Beitrag, Revision. Gesamtheit der gemeinsam übergebenen (und freigeschalteten) Änderungen einer Arbeitsphase. Dazu zählen die zeilenweisen Änderungen der Textdateien und die vollständigen veränderten Binärdateien.



BEGRIFFE VI

Origin

Bezeichnet den primären Remote Tracking Branch eines Repositorys.

Master

Ausgangszweig eines Projektes. Wird beim initialisieren automatisch erstellt. Primärer Arbeitszweig.



BEGRIFFE VII

Upstream

Stromaufwärts. Übertragen: Sender/Quelle. Ein Repository, auf welches vom Arbeitsverzeichnis aus nur lesend zugegriffen wird. Änderungen können von Upstream direkt in Master verschmolzen werden. Umgekehrt ist ein \rightarrow Pull (Request) nötig. In der Regel wird nach einem Fork das Ausgangsrepository als Upstream definiert.

Fetch

Abholen. Kopieren von Commits aus einem entfernten Repository (z. B. Upstream oder Origin) in den gleichnamigen Zweig des lokalen Repositories. Beispiel: `git fetch upstream`



BEGRIFFE VIII

Push

Schieben. Senden von Commits aus einem Zweig des lokalen Repositorys in ein entferntes Repository. Beispiel: `git push origin master`

Pull

Ziehen. Übernahme von Commits aus einem Fork in das Ausgangsprojekt durch den Besitzer des Ausgangsrepositorys. Verbunden mit einem `→ Merge`.



BEGRIFFE IX

Pull Request

Zug-Anfrage. Anfrage eines Autors von Änderungen, diese aus dessen Fork in das ursprüngliche Projekt zu übernehmen. Wird in der Regel über das Webinterface des Git-Servers ausgelöst.

Diff

Differenz. Zeilenweises Anzeigen von Unterschieden in Textdateien des Projektes und Anzeige der geänderten Binärdateien. Der Stil entspricht dem Unix-Programm diff. Es ist die Anzeige bezüglich einzelner Commits, einzelner Zweige, eines Stichdatums oder eines Zeitraumes möglich.



BEGRIFFE X

Merge

Verschmelzen. Übernahme sämtlicher Commits eines Zweiges in einen anderen, wonach die Zweige synchron sind. Beispiel: `git merge upstream/master`

Clone

Klonen. Erstellen einer vollständigen Kopie eines anderen Repositorys. Die Quelle kann lokal oder entfernt sein, das Ziel ist standardmäßig ein Unterordner im aktuellen Verzeichnis. Notwendig, um eine lokale Arbeitskopie eines geforkten Projekts zu erstellen. Die angegebene Quelle wird automatisch als Origin definiert. Beispiel: `git clone https://Ekkehardt@github.com/Ekkehardt/linux.git`



BEGRIFFE XI

Hash

Zeichenfolge fester Länge, welche aus Eingabedaten beliebiger Länge über eine Hashfunktion berechnet wird. Kleine Änderungen an den Eingabedaten bewirken große Änderungen im Hash. Dient zum Nachweis der Integrität der Daten und zur Identifikation der einzelnen Commits. Entspricht folglich einer Prüfsumme und Revisionsnummer.



EIGENSCHAFTEN

- **nicht-lineare Entwicklung**

Erstellen und Verschmelzen von Entwicklungszweigen; effiziente Implementierung der Zweige.

- **kein zentraler Server**

Jeder Entwickler besitzt eine Arbeitskopie mit der gesamten Projektgeschichte. Alle Remotes und lokale Kopien sind technisch gleich.

- **Datentransfer zwischen Repositorys**

file://, git://, ssh://, http://, https://, ftp:// oder rsync://

- **Kryptographische Sicherheit der Projektgeschichte**

Der Hash eines einzelnen Commits basiert auf der vollständigen Geschichte, die zu diesem Commit geführt hat; markieren und signieren (GPG) einzelner Commits.



EIGENSCHAFTEN II

■ **Speichersystem und Dateiversionierung**

Es erhält nicht jede Datei eine Revisionsnummer, wie bei CVS, sondern es werden Verweise ähnlich einem Dateisystem gespeichert. Ändert sich eine Datei nicht, ist kein erneutes Speichern nötig.

■ **Säubern des Repositorys**

Daten aller Commits und Zweige bleiben bis zum expliziten löschen vorhanden, um Änderungen zurücknehmen zu können.

■ **Interoperabilität**

Über Hilfsprogramme: Austausch mit GNU arch (git-archimport), CVS (git-cvsexportcommit, git-cvsimport und git-cvsserver), Darcs (darcs-fastconvert, darcs2git und andere), Quilt (git-quiltimport) und Subversion (git-svn).



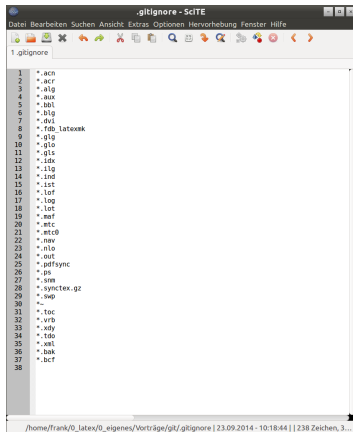
EIGENSCHAFTEN III

■ **Web-Interface**

Gitweb, git-tf, GitHub, bitbucket, ...

- Läuft auf fast allen modernen unixartigen Systemen, wie Linux, Solaris, Mac OS X, FreeBSD, DragonFly BSD, NetBSD, OpenBSD, AIX, IRIX und Haiku.
- Unter Microsoft Windows mit dem Client von GitHub, mit Hilfe der Cygwin-Umgebung, mit Msysgit oder der TortoiseGit-Shell-Erweiterung anwendbar.

AUFBAU EINES LOKALEN REPOSITORYS II



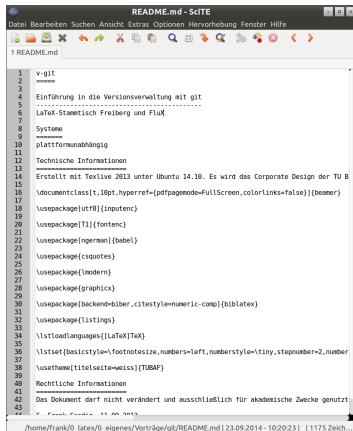
```

1 *.acn
2 *.acr
3 *.alg
4 *.aux
5 *.bbl
6 *.blg
7 *.dvi
8 *.fdb_latexmk
9 *.glg
10 *.glo
11 *.gls
12 *.idx
13 *.ilg
14 *.ind
15 *.ist
16 *.lof
17 *.log
18 *.lot
19 *.maf
20 *.mtc
21 *.mtc0
22 *.nav
23 *.nlo
24 *.out
25 *.pdfsync
26 *.ps
27 *.sim
28 *.synctex.gz
29 *.swp
30 ~
31 *.toc
32 *.vrb
33 *.xdy
34 *.tdo
35 *.xsl
36 *.bak
37 *.bct
38
  
```

/home/frank/0_latex/0_eigenes/Vorträge/git/.gitignore | 23.09.2014 - 10:18:44 | | 238 Zeichen, 3...

Abb. 2: die Datei .gitignore

AUFBAU EINES LOKALEN REPOSITORYS III



```

1 v-git
2 =====
3
4 Einführung in die Versionsverwaltung mit git
5 .....
6 LaTeX-Stammtisch Freiberg und FluX
7
8 Systeme
9 =====
10 plattformunabhängig
11
12 Technische Informationen
13 .....
14 Erstellt mit Texlive 2013 unter Ubuntu 14.10. Es wird das Corporate Design der TU B
15
16 \documentclass[t,10pt,hyperref={pdfpagemode=FullScreen,colorlinks=false}]{beamer}
17
18 \usepackage{utf8}{inputenc}
19
20 \usepackage[T1]{fontenc}
21
22 \usepackage[ngerman]{babel}
23
24 \usepackage{csquotes}
25
26 \usepackage{lmodern}
27
28 \usepackage{graphicx}
29
30 \usepackage[backend=biber,citestyle=numeric-comp]{biblatex}
31
32 \usepackage{listings}
33
34 \lstloadlanguages{[LaTeX]TeX}
35
36 \lstset{basicstyle=\footnotesize,numbers=left,numberstyle=tiny,stepnumber=2,number
37
38 \usesthemetitleseite=weiss}[TUBAF]
39
40 Rechtliche Informationen
41 .....
42 Das Dokument darf nicht verändert und ausschließlich für akademische Zwecke genutzt
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
  
```

Abb. 3: die Datei README.md



MIT LEEREM REMOTE-REPO BEGINNEN

- repo anlegen (Homepage)
- in Ordner mit zu verwaltenden Daten wechseln

```
git init
git add -A
git commit -m "erster Commit"
git remote add origin https://User@bitbucket.org/
    User/repo.git
git push -u origin master
```



REMOTES LÖSCHEN

```
git remote remove origin  
Version 1.7.10 und vorher: git remote rm origin
```



REMOTES UMZIEHEN

- neues Remote anlegen (Homepage)

```
git remote set-url origin https://new.url.here
git push -u origin master
```

- altes Remote löschen (Homepage)



REPO KLONEN

- ggf. zuerst auf HP forken

```
git clone https://github.com/User/repo.git
```



ÄNDERUNGEN ZURÜCKGEBEN

```
git add -A  
git commit -m "Update message"  
git push origin master
```


└ Beispiele

└ Änderungen zurückgeben

ÄNDERUNGEN ZURÜCKGEBEN

```
git add -A  
git commit -m "update message"  
git push origin master
```

Origin ist bereits definiert; evtl Pull-Request über Homepage



ÄNDERUNGEN VOM UR-REPO HOLEN

```
git remote add upstream https://github.com/  
    andererUser/ursrungsrepo.git  
git fetch upstream  
git merge upstream/master
```



MANUELL MERGEN

`git mergetool`

- z. B. opendiff, kdiff3, tkdiff, xxdiff, meld, tortoisemerge, gvimdiff, diffuse, diffmerge, ecmerge, p4merge, araxis, bc3, codecompare, emerge, vimdiff



ÄNDERUNGEN ANZEIGEN

Zustand des Repos:

```
git status
```

grafisch:

```
gitk
```

Liste aller Commits mit ihrem SHA1-Hash:

```
git log
```

in einem bestimmten Commit:

```
git diff "@{e1b9}"
```

seit gestern:

```
git diff "@{yesterday}"
```



ÄNDERUNGEN ANZEIGEN II

seit dem 2. Januar 1970:
`git diff "@{1970-01-02}"`

zwischen irgendeiner Version und der vorvorletzten:
`git diff 1b6d "master~2"`

alle Commits der letzten zwei Wochen:
`git whatchanged --since="2 weeks ago"`



Beispiele

ÄNDERUNGEN ANZEIGEN III

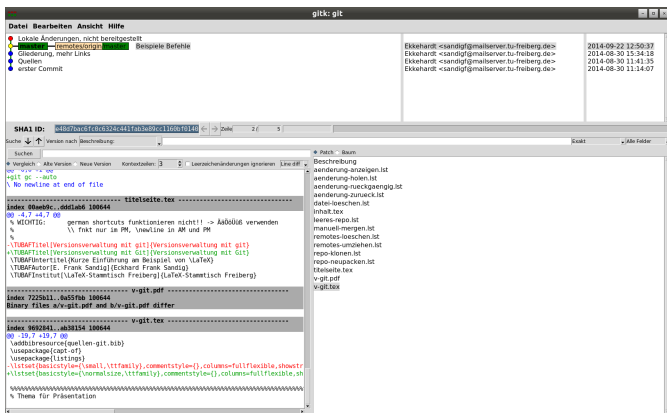


Abb. 4: Das Programm git

└ Beispiele

└ Änderungen Anzeigen III

ÄNDERUNGEN ANZEIGEN III



Abb. 6 Das Programm gitk

gitk muss in der Regel nachträglich installiert werden

ÄNDERUNGEN ANZEIGEN IV

```

frank@orcrlist:~/0_latex/0_eigenes/Vorträge/git
commit a21eaf234a5684ba4a14b4dc421555be168ad475
Merge: d1de0b4 c4902c4
Author: Ekkehardt <sandigf@mailserver.tu-freiberg.de>
Date: Tue Sep 23 10:30:28 2014 +0200

    Merge remote-tracking branch 'origin/master'

commit c4902c4e1ddce9c122489733e1da65eab46ebca0
Author: Ekkehardt <sandigf@mailserver.tu-freiberg.de>
Date: Tue Sep 23 10:28:23 2014 +0200

    TEST

commit d1de0baf57ab2480ed2d17af4c2d63513ce453e4
Author: Ekkehardt <sandigf@mailserver.tu-freiberg.de>
Date: Tue Sep 23 10:35:25 2014 +0200

    erste_Bildschirmfotos

commit 4c150ec91910f678bdaf5b4245b34253040bf72a
Author: Ekkehardt <sandigf@mailserver.tu-freiberg.de>
Date: Tue Sep 23 10:25:25 2014 +0200

    erste_Bildschirmfotos

commit e48d7bac6f0c6324c441fab3e89cc1160b16140
Author: Ekkehardt <sandigf@mailserver.tu-freiberg.de>
Date: Mon Sep 22 12:50:37 2014 +0200

    Beispiele_Befehle

commit 413514f600f052fd750e904e1054690bf745194
Author: Ekkehardt <sandigf@mailserver.tu-freiberg.de>
Date: Sat Aug 30 15:34:18 2014 +0200

    Gliederung, Mehr Links

commit 429f02038b132f2622827e8bba4a1e92a4217d88
Author: Ekkehardt <sandigf@mailserver.tu-freiberg.de>
Date: Sat Aug 30 11:41:35 2014 +0200

    Quellen
  
```

Abb. 5: Ausgabe des Befehls `git log`

ÄNDERUNGEN ANZEIGEN V

```

frank@orcrlist:~/0_latex/0_eigenes/Vorträge/git
diff --git a/.gitignore b/.gitignore
index 3e4b780..61c120c 100644
+++ a/.gitignore
++++ b/.gitignore
@@ 25,6 +25,7 @@
+ .xml
+ .bak
+ .bcf
+verlage.karl
+ .note

diff --git a/README.md b/README.md
index 7cb468f..f47d90c 100644
--- a/README.md
+++ b/README.md
@@ 1,7 +1,7 @@
- git
+ Einführung in die Versionsverwaltung mit git
+
+ LATEX-Stammtisch Freiberg und BauK am BATUM
+ LATEX-Stammtisch Freiberg und Fluss
+
+ Systeme
+
diff --git a/abb/README.png b/abb/README.png
new file mode 100644
index 0000000..17bb3b6
Binary files /dev/null and b/abb/README.png differ
diff --git a/abb/gitignore.png b/abb/gitignore.png
new file mode 100644
index 0000000..759baad
Binary files /dev/null and b/abb/gitignore.png differ
diff --git a/abb/gitk.png b/abb/gitk.png
new file mode 100644
index 0000000..d023ae5
Binary files /dev/null and b/abb/gitk.png differ
diff --git a/abb/repo.png b/abb/repo.png
new file mode 100644
index 0000000..23cd354
Binary files /dev/null and b/abb/repo.png differ
diff --git a/aenderung-anzeigen.lst b/aenderung-anzeigen.lst
new file mode 100644
<+> Viewing <+>stream
  
```

Abb. 6: Ausgabe des Befehls `git diff "@{yesterday}"`



ÄNDERUNGEN ANZEIGEN VI

```

frank@orcrlist:~/0_latex/0_eigenes/Vorträge/git
$ git whatchanged --since="2 weeks ago"
commit c4902c4e1ddce9c12248973c1da65eab46ebca0
Author: Ekkehardt <sandig@mailserver.tu-freiberg.de>
Date: Tue Sep 23 10:28:23 2014 +0200

    TEST

100644 000000 e69de29..f000000 0 bitbucket.note
100644 000000 e69de29..0000000 0 github.note
100644 000000 e69de29..0000000 0 README.md

commit d1debbef57ab2460ed2d1f0f4c2da3513ce453e4
Author: Ekkehardt <sandig@mailserver.tu-freiberg.de>
Date: Tue Sep 23 10:25:25 2014 +0200

    erste Bilderfotos

100644 100644 3e4b780..61c320c M .gitignore
100644 100644 7cb400f..f47d98c M README.md
100644 100644 0000000..170b306 A abb/README.png
100644 100644 0000000..739baad A abb/gitignore.png
100644 100644 0000000..4623aa5 A abb/gitk.png
100644 100644 0000000..23cd354 A abb/repo.png
100644 100644 3e08303..d46176c M inhalt.tex
100644 100644 ddd1ab6..01fc3af M titelseite.tex
100644 100644 0a551bb..1c931ff M w915.pdf
100644 100644 a033194..f916320 M w916.tex

commit 4c15bec919101670bda150a245b3425304dbf729
Author: Ekkehardt <sandig@mailserver.tu-freiberg.de>
Date: Tue Sep 23 10:25:25 2014 +0200

    erste Bilderfotos

100644 100644 3e4b780..61c320c M .gitignore
100644 100644 7cb400f..f47d98c M README.md
100644 100644 0000000..170b306 A abb/README.png
100644 100644 0000000..739baad A abb/gitignore.png
100644 100644 0000000..4623aa5 A abb/gitk.png
100644 100644 0000000..23cd354 A abb/repo.png
100644 100644 0000000..e69de29 A bitbucket.note
100644 100644 0000000..f000000 A github.note
100644 100644 3e08303..d46176c M inhalt.tex
100644 100644 ddd1ab6..01fc3af M titelseite.tex
  
```

Abb. 7: Ausgabe des Befehls `git whatchanged --since="2 weeks ago"`

ÄNDERUNGEN ANZEIGEN VII



projects / dev/liblist / committiff git

[summary](#) | [shortlog](#) | [log](#) | [commit](#) | [committiff](#) | [tree](#)
[raw](#) | [patch](#) (parent: [@23715a](#)) commit search:

Changed do...while to while; and added a line in test.c to test it. master

author [dmeszner <der-messner@...>](#)
 Mon, 3 Oct 2011 07:54:58 +0200 (07:54 +0200)
 committer [dmeszner <der-messner@...>](#)
 Mon, 3 Oct 2011 07:54:58 +0200 (07:54 +0200)

REAME [patch](#) | [blob](#) | [history](#)
 src/liblist_single.c [patch](#) | [blob](#) | [history](#)
 src/test.c [patch](#) | [blob](#) | [history](#)

```
diff --git a/README b/README
index de1af70..dda095a 100644 (file)
--- a/README
+++ b/README
@@ -3,6 +3,6 @@ What you should know:
- THERE IS NO WARRANTY FOR ANYTHING. THIS SOFTWARE WAS WRITTEN IN ALL CONSCIENCE AND WITH THE GOAL OF A GOOD SOFTWARE.
- detailed documentation can be found in doc/index.html
- the file src/test.c is used to test new features of the libs in a q'n'd compiler session with static linking (e.g. 'gcc test.c liblist_single.c').
+   You may use it to test the library functions easily.
-   the build process is described in doc/index.html
+   You may use it to test the library functions easily, find bugs etc.
+   but the build process is described in doc/index.html, also the functions, the data structures etc.
+   for any questions, write a mail to <der-messner@...>
```

```
diff --git a/src/liblist_single.c b/src/liblist_single.c
index d33e6d3..ac3a9bc 100644 (file)
--- a/src/liblist_single.c
+++ b/src/liblist_single.c
@@ -15,8 +15,6 @@
but the redistributed software has to be licensed with LGPL v3.
*/

-extern char **environ;
-
// Source file for single linked list (SLL)

struct element_s
@@ -282,10 +280,8 @@ void* oetdata random array s(struct list s* list, unsigned long number) // Acces
```

Abb. 8: Die Oberfläche gitweb



ÄNDERUNGEN RÜCKGÄNGIG MACHEN

unwiederbringlich zum angegebenen Stand zurückkehren:

```
git reset --hard 766f
```

angegebenen Commit rückgängig machen (als neuen Commit):

```
git revert 1b6d
```

letzten veröffentlichten Stand wiederherstellen:

```
git fetch origin  
git merge origin/master
```



REPO NEU PACKEN

```
git gc --auto
```



DATEI AUS ALLEN COMMITS LÖSCHEN

```
git filter-branch --prune-empty --index-filter "git
  rm --cached -f \---ignore-unmatch Dateiname.pdf
  " -- --all
```

└ Beispiele

└ Datei aus allen Commits löschen

DATEI AUS ALLEN COMMITS LÖSCHEN

```
git filter-branch --prune-empty --index-filter 'git
rm --cached -f \' --ignore-unmatch Dateiname.pdf
' -- --all
```

Nach git filter-branch muss das remote gelöscht und neu angelegt werden, danach ein push. Sonst entstehen unlösbare Konflikte.



ARBEITEN MIT ZWEIGEN

Zweige anzeigen:
`git branch`

Zweig anlegen:
`git branch NAME`

in Zweig wechseln:
`git checkout NAME`

Anlegen und Wechseln gleichzeitig:
`git checkout -b NAME`



ARBEITEN MIT ZWEIGEN II

Zweige verschmelzen:

```
git merge QUELLE/ZIEL
```

Zweige löschen:

```
git branch -d NAME
```

WEITERE INFORMATIONEN

- [1] URL: <https://www.atlassian.com/de/git/tutorial/git-basics>.
- [2] URL: <http://git-scm.com/book/de>.
- [3] URL: <http://git-scm.com/documentation>.
- [4] URL: <http://rogerdudler.github.io/git-guide/index.de.html>.
- [5] URL:
<http://www-cs-students.stanford.edu/~blynn/gitmagic/intl/de>.
- [6] URL: http://mbork.pl/2014-10-18_Version_Control_Systems.
- [7] URL: <http://svij.org/blog/2014/10/25/git-fur-einsteiger-teil-1>.
- [8] URL: <http://svij.org/blog/2014/11/01/git-fur-einsteiger-teil-2>.
- [9] URL: <http://svij.org/blog/2015/01/05/was-ist-git/>.
- [10] URL: <http://svij.org/blog/2015/01/12/git-fur-einsteiger-teil-3/>.



WEITERE INFORMATIONEN II

- [11] URL: <http://wiki.ubuntuusers.de/Git>.
- [12] URL: <http://git-scm.com>.
- [13] URL: <http://de.wikipedia.org/wiki/Git>.



Ende der Präsentation

VIELEN DANK! – NOCH FRAGEN?

- schriftsatz@sandig-fg.de
- <http://github.com/Ekkehardt>
- Ekkehardt in #latex-de auf irc.freenode.net, auf <http://texwelt.de> und im Forum unter <http://www.golatex.de>
- Präsentation unter <http://sandig-fg.de> und in Kürze unter <http://www.suedraum.de/latex/stammtisch>