TECHNISCHE UNIVERSITÄT
BERGAKADEMIE FREIBERG
Die Ressourcenuniversität. Seit 1765.

Versionsverwaltung mit Git

Kurze Einführung am Beispiel von LaTeX

E. Frank Sandig

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.

Geschichte

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

Geschichte

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)

ALTERNATIVEN

verteilte Versionsverwaltung:
- Monotone
- BitKeeper
- Mercurial
- Bazaar
- GNU arch

zentrale Versionsverwaltung:
- Apache Subversion (svn)

auch: lokale Versionsverwaltung, eher selten

Geschichte

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 LaTeX 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: $\rightarrow$ 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 verteilt 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 LaTeX-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 → 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 Repositorys. 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 htt-ps://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.

Eigenschaften

AUFBAU EINES LOKALEN REPOSITORYS

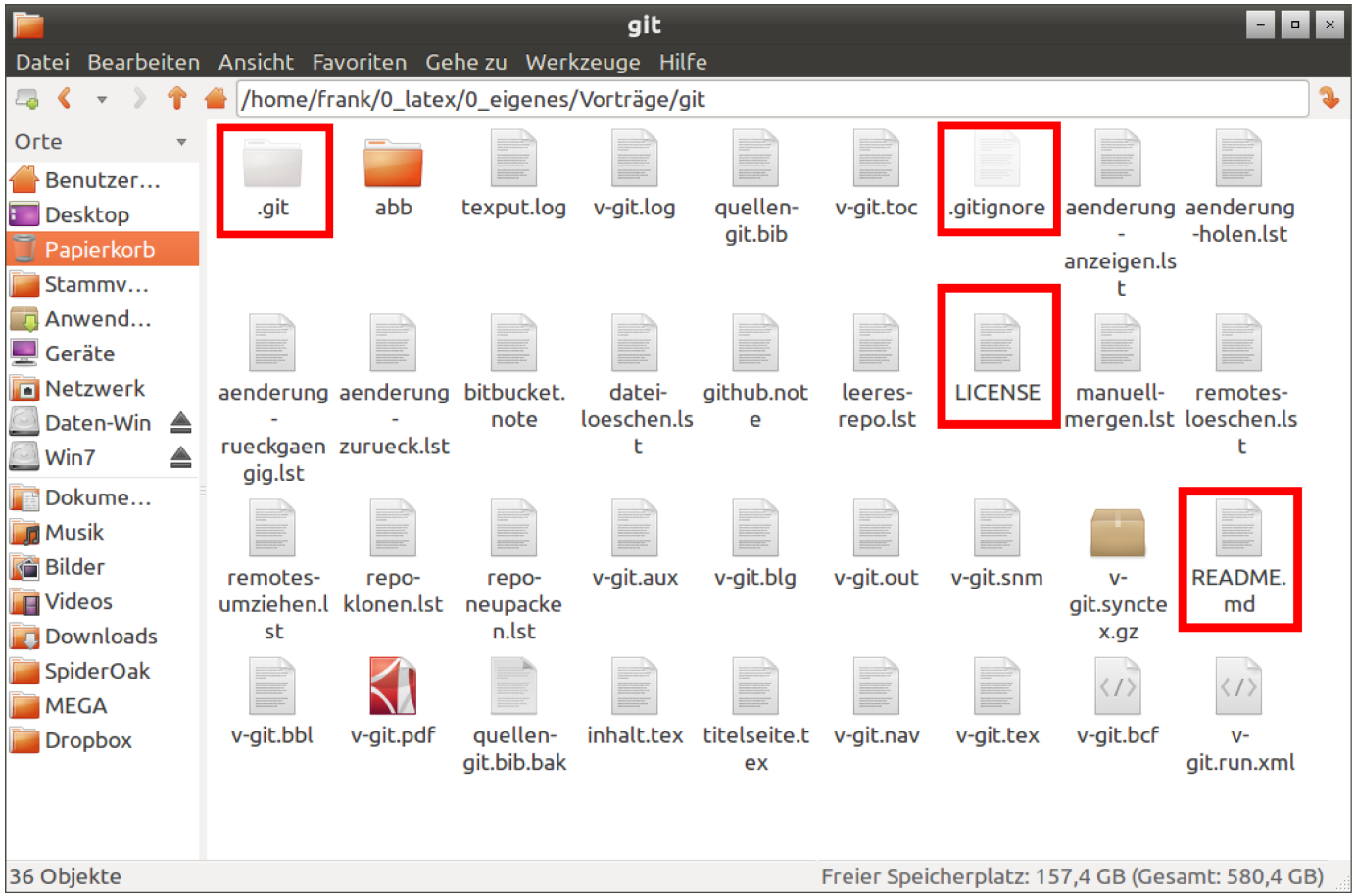

Abb. 1: lokales Repository dieses Vortrages

AUFBAU EINES LOKALEN REPOSITORYS II

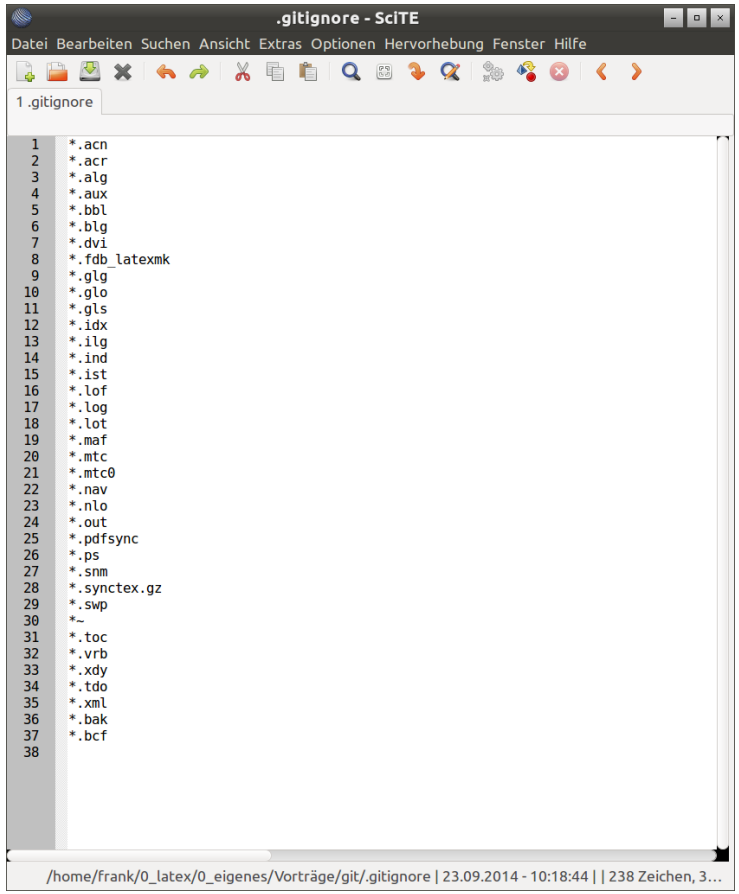

Abb. 2: die Datei .gitignore

AUFBAU EINES LOKALEN REPOSITORYS III

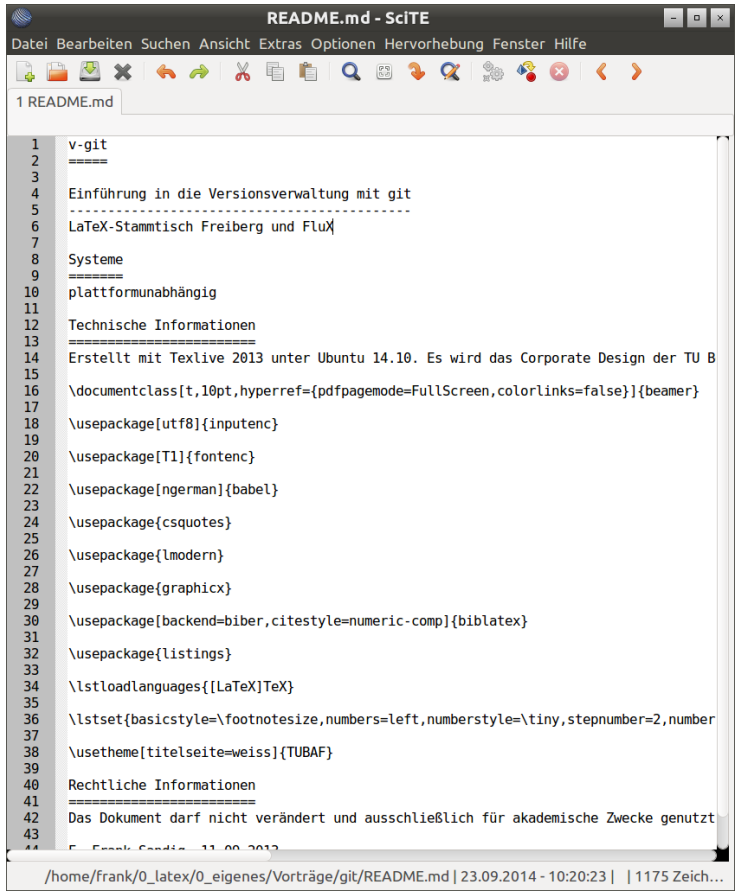

Abb. 3: die Datei README.md

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
```

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"
```

ÄNDERUNGEN ANZEIGEN III

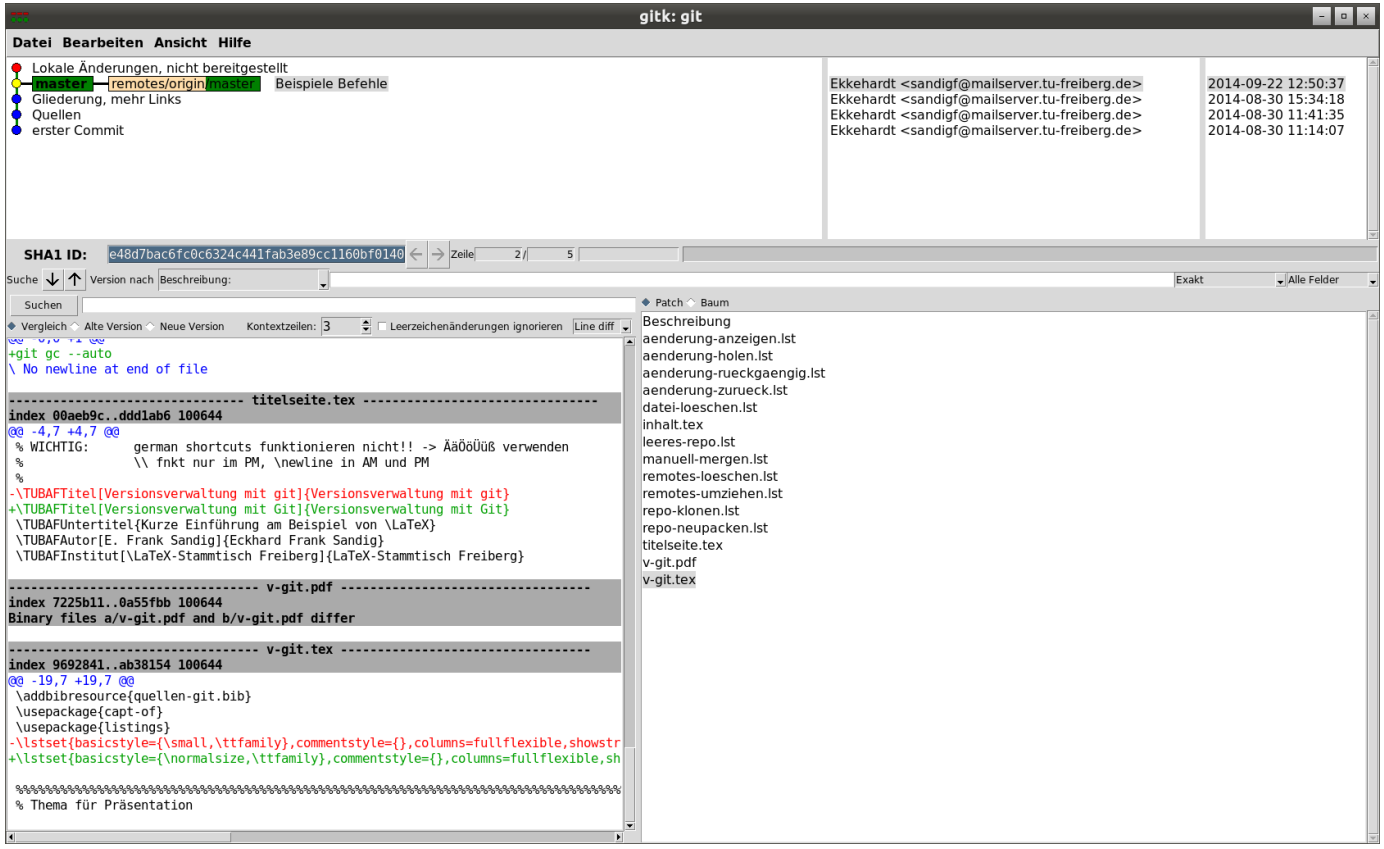

Abb. 4: Das Programm gitk

ÄNDERUNGEN ANZEIGEN III

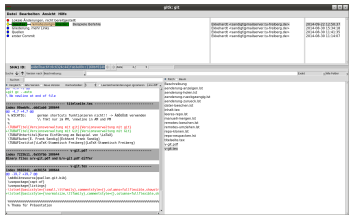

Abb. 4: Das Programm gitk

gitk muss in der Regel nachträglich installiert werden

ÄNDERUNGEN ANZEIGEN IV

Abb. 5: Ausgabe des Befehls `git log`

Beispiele

ÄNDERUNGEN ANZEIGEN V

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

ÄNDERUNGEN ANZEIGEN VI

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

ÄNDERUNGEN ANZEIGEN VII

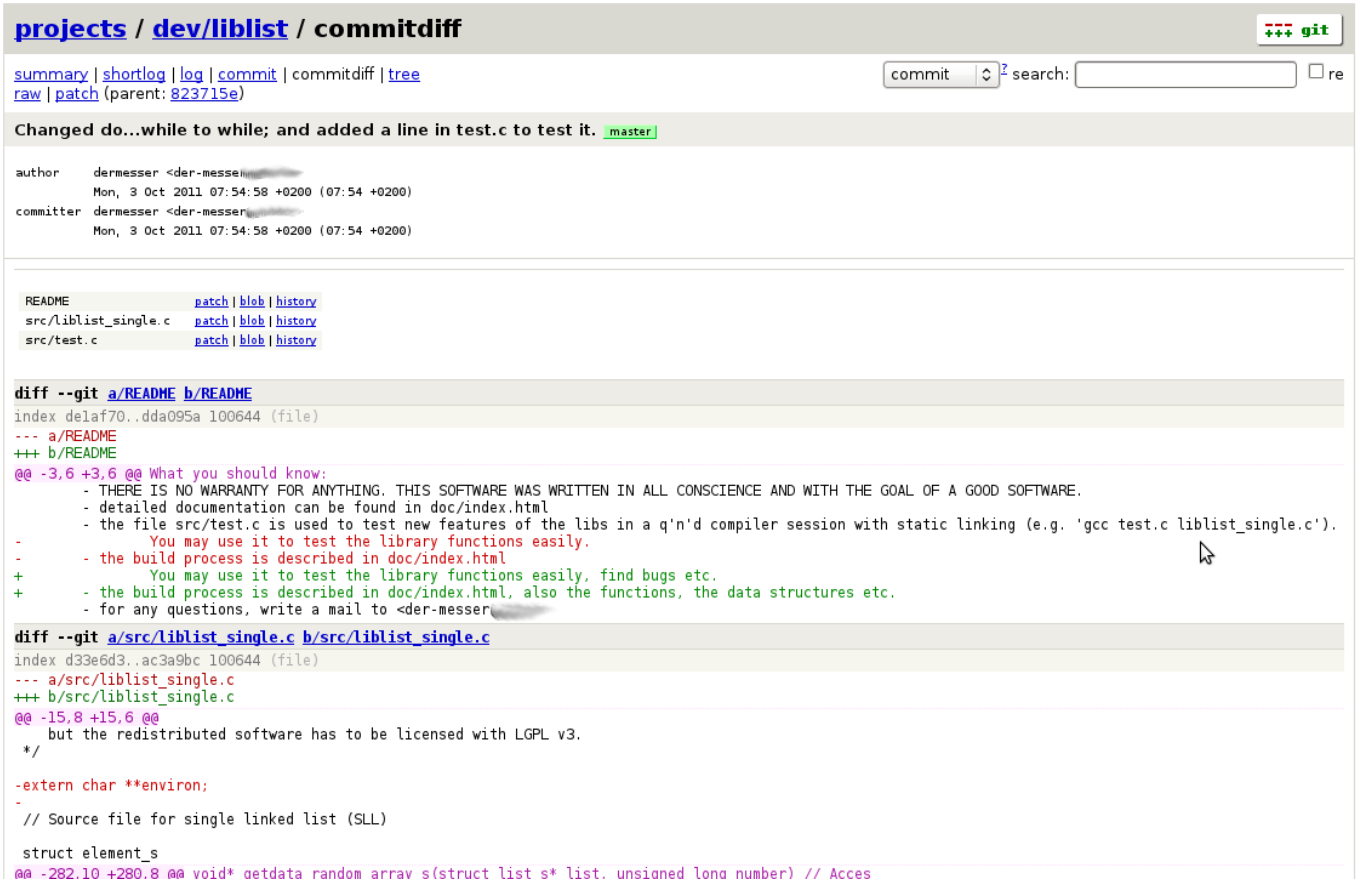

Abb. 8: Die Oberfläche gitweb

ÄNDERUNGEN RÜCKGÄNGIG MACHEN

```
unwiederbringlich zum angegebenen Stand zur
    ückkehren:
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