Revision Control Systems

- Definition
- Quick recap of general use
- Terms used often
- History
- Two advanced concepts

Definition

- System used to reliably reproduce specific revision of software over time

Quick recap of general use

- scm checkout software-1.0
- Edit and test in your local sandbox
- scm checkin
  - “I added a new feature that does X”
- scm branch mainline software-2.1
- scm merge feature-branch software-1.1

Terms used often

- Repository
- Revision
- Checkout
- Commit

Terms used often (continued)

- Mainline *or* Trunk
- Branch
- Merge
- Tag

History

- SCCS
- RCS/CVS
- Subversion
- First Wave DVCS: Arch, Monotone
- Current Wave DVCS: Bazaar, Mercurial, Git

SCCS

- First revision control system
- Single file based
- Control records
- Delta table
- Revision information: who, when, changes

RCS

- Checkout needs locks
- Single file based, history stored in `filename,v`
- Tags for groups of files
- Reverse deltas and forward deltas
- Revision information: who, when, changes, branch, tags

CVS

- No locks
- Client / server
- Anonymous readonly access
- Previous de-facto open source RCS

CVS, continued

- Revision information: who, when, changes, branch, tags
- `cvs log boot.c`

```
RCS file: /cvs/src/sys/arch/sparc/stand/boot/boot.c,v
Working file: boot.c
head: 1.6
branch:
locks: strict
access list:
symbolic names:
   OPENBSD_4_7: 1.6.0.26
       [...]
keyword substitution: kv
total revisions: 12;   selected revisions: 1
description:
----------------------------
revision 1.3
date: 2002/03/14 01:26:44;  author: millert;  state: Exp;  lines: +5 -5
First round of __P removal in sys
=============================================================================
```

Subversion

- Mission: “Build a better CVS”
- Atomic commits
- Copy files and directories
- Better branching, merging and tagging
- Current de-facto open source RCS

Subversion, continued

- Revision information: who, when, changes, branch, tags
- `svn log -v https://svn.win.tue.nl/repos/MCRL2/trunk`

```
------------------------------------------------------------------------
r7524 | wieger | 2010-03-18 11:38:17 +0100 (Thu, 18 Mar 2010) | 1 line
Changed paths:
   M /trunk/tools/pbespareqelm/pbespareqelm.cpp

- Fixed typo in option string.
```

Arch

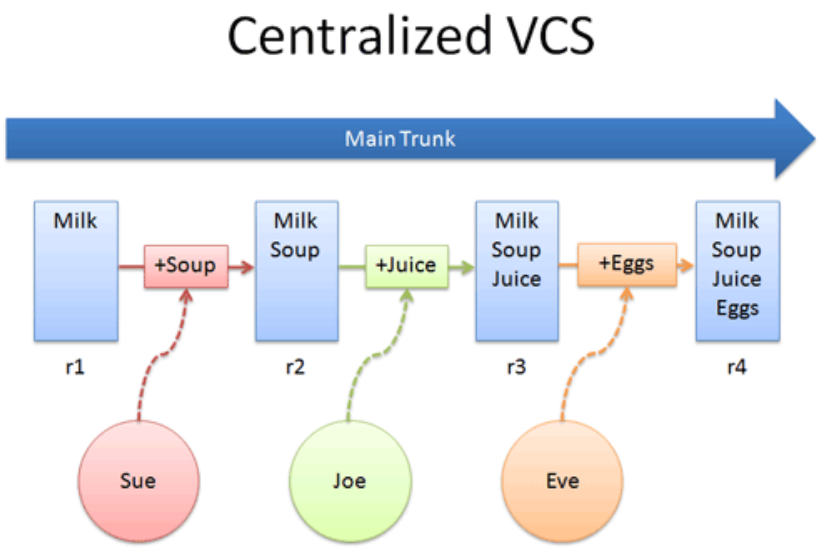
- Mission: “Replace CVS”
- Bizarre command set and file naming conventions
  - `$ tla changes -o ,,new-robot-comment`
- But: distributed version control!
- Not in use anymore

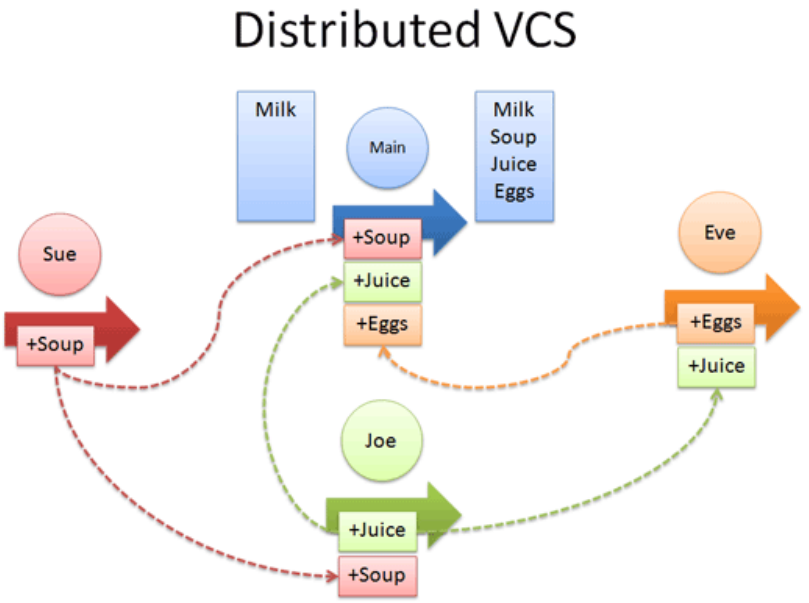
Monotone

- Mission: “Replace CVS”
- Correctness, verifiable history
- Initially slow
- But: distributed version control!
- Still in use

Distributed vs Centralized

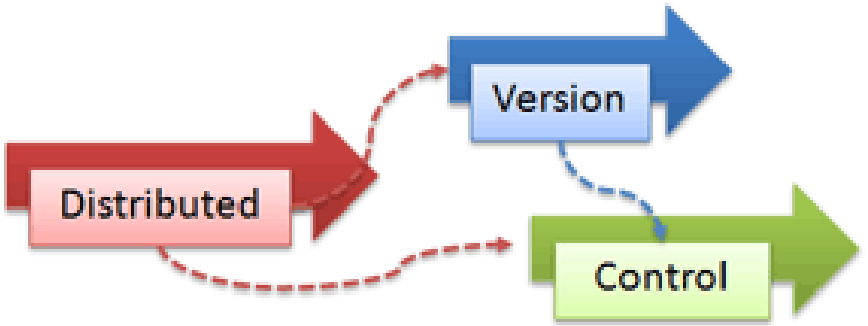
- Differences
- Distributed is “chaos”
- Resistance to change
- “Chaos” is manageable





Distributed vs Centralized, ct'd

- “Throw away” development
- Delegate subprojects
- “Central” release repository



The current big three

- Bazaar
- Mercurial
- Git
- A lot of overlap

Bazaar

- Evolved from Arch
- Mission: “Accessible DRCS”
- Cross-platform
- Slow initially, better now
- Projects: MySQL, Debian APT

Unique features of Bazaar

- Branches are simply new clones of repositories
- Very good support for graphical UIs on all platforms
- Tracks both file and directory renames explicitly

Bazaar log

- bzr log lp:bzr-svn

```
------------------------------------------------------------
revno: 3274
committer: Jelmer Vernooij <jelmer@samba.org>
branch nick: trunk
timestamp: Tue 2010-03-09 13:56:24 +0100
message:
  Provide BranchConfig._get_change_editor().
------------------------------------------------------------
revno: 3273
committer: Jelmer Vernooij <jelmer@samba.org>
branch nick: trunk
timestamp: Tue 2010-03-09 00:47:01 +0100
message:
  Fix use of svn:author=author.
```

Mercurial

- Mission: replace Bitkeeper
- Inspiration from Monotone
- Cross platform
- Projects: SUN Java, Mozilla Firefox, Python

Unique features of Mercurial

- Global branches
- Local revision numbers
- Built in server

Mercurial log

- hg log mozilla

```
changeset:   15262:263749849d0b
tag:         tip
user:        Marco Zehe <marco.zehe@googlemail.com>
date:        Thu Jun 05 11:49:51 2008 +0200
summary:     b=432467, firefox segfaults in plone kupu editor
[@ nsDocAccessible::FlushPendingEvents], on Tablet PC
[@arena_dalloc_small] (steps to reproduce in comment #26),
p=Ginn Chen <ginn.chen@sun.com>, r=surkov

changeset:   15261:49cdeb4f8144
user:        Simon Montagu <smontagu@smontagu.org>
date:        Thu Jun 05 04:05:25 2008 -0700
summary:     Reftest for bug 229764
```

Git

- Mission: replace Bitkeeper
- Inspiration from Monotone
- Linux dependant at first
- Projects: Linux Kernel, Android, Wine, Perl

Unique features of Git

- Tracks file contents, not files
- Internal details highly visible

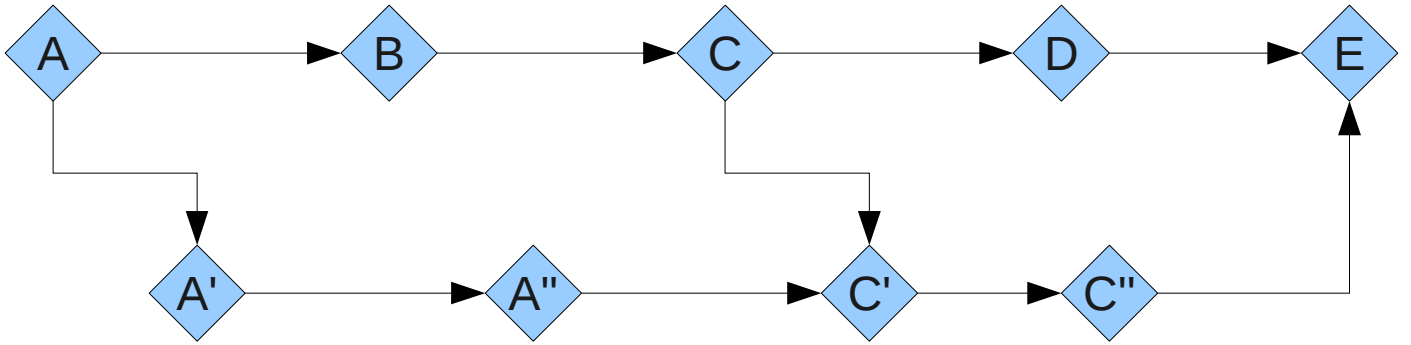
Git log

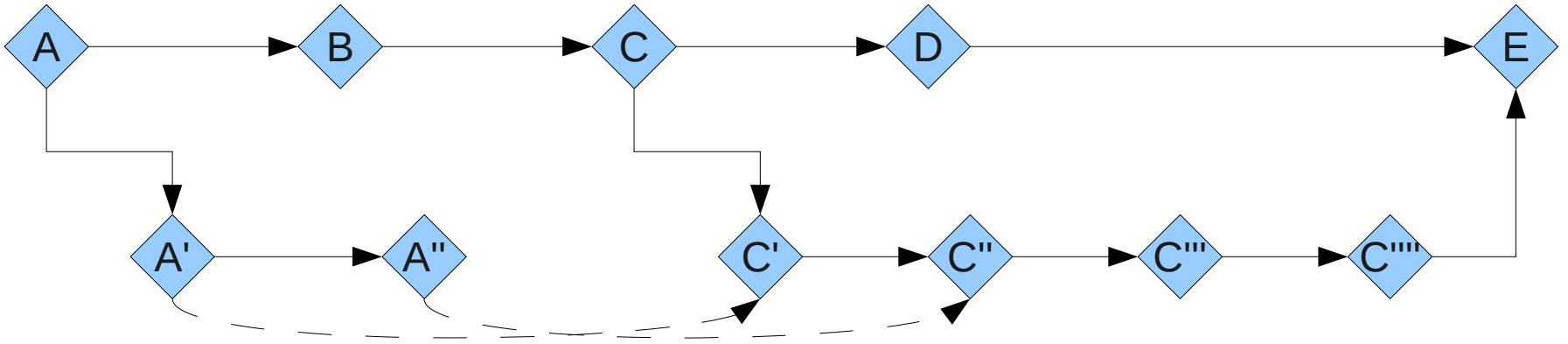
- git log HEAD

```
commit ef36c9a3b2828f5a11feda9e4d2708bf3a4a7a52
Author: Eric Anholt <eric@anholt.net>
Date:   Wed Mar 17 12:46:21 2010 -0700

    intel: Install the header file in the libdrm/ directory.

    Suggested-by: Rémi Cardona <remi@gentoo.org>
    Signed-off-by: Eric Anholt <eric@anholt.net>
```

Advanced concepts

- Rebase versus merge





Advanced concepts, ct'd

- Octopus merge

Further reading

- Original SCCS paper:
  http://basepath.com/aup/talks/SCCS-Slideshow.pdf
- History of RCS:
  http://www.burlingtontelecom.net/~ashawley/rcs/tichy1985rcs/rcs.html#id2504601
- Linux Kernel and BitKeeper:
  http://lkml.indiana.edu/hypermail/linux/kernel/0504.0/1540.html

  http://lkml.org/lkml/2005/4/7/150
- Distributed vs Centralized:
  http://blog.ianbicking.org/distributed-vs-centralized-scm.html

  http://web.mit.edu/ghudson/thoughts/bitkeeper.whynot

Revision Control Systems

- Definition
- Quick recap of general use
- Terms used often
- History
- Two advanced concepts

Definition

- System used to reliably reproduce specific revision of software over time

Quick recap of general use

- scm checkout software-1.0
- Edit and test in your local sandbox
- scm checkin
  - “I added a new feature that does X”
- scm branch mainline software-2.1
- scm merge feature-branch software-1.1

Checking out is getting a local sandbox copy

Checking in registers your name (author), time and date, the changes, and a log message. Maybe even more properties depending on SCM.

Branching is often used for parallel development of next version vs released versions, or for developing of larger features for the software

Merging can be done to merge features back into mainline or released versions

Terms used often

- Repository
- Revision
- Checkout
- Commit

Repository is the place where the historic data is stored.

A revision is the name for the software at a specific point in time

A checkout is a local copy of a revision; it is the basis for further work

A commit is the act of adding modifications to the repository, thereby creating more history

Terms used often (continued)

- Mainline *or* Trunk
- Branch
- Merge
- Tag

The mainline or trunk is the main development revision path of a software system

A branch is a continuation of development of the software in parallel to another such path

To merge is to integrate changes on multiple development paths back into a single path, the opposite of a branch

A tag is a symbolic name given to a specific revision of the software

History

- SCCS
- RCS/CVS
- Subversion
- First Wave DVCS: Arch, Monotone
- Current Wave DVCS: Bazaar, Mercurial, Git

Revision control systems have been here for a few decades already

Before the first such system people managed by carefully making copies of all files at certain points

This led to lots of difficulties in managing history of development, hence an RCS

The history I will show here consists of about 5 phases, with appropriate RCSes as shown on the sheet

There are of course lots more systems available

SCCS

- First revision control system
- Single file based
- Control records
- Delta table
- Revision information: who, when, changes

Problems SCCS tried to help with, as described in the original paper: amount of space needed to store source code (!), fixes that fail to make all versions, tell details about changes, and identify versions of software in use by customers

Control records govern the insertion or deletion of a block of lines – a concept used by many other RCSes down the line

The delta table associates control records with revisions, on a per file basis

RCS

- Checkout needs locks
- Single file based, history stored in `filename,v`
- Tags for groups of files
- Reverse deltas and forward deltas
- Revision information: who, when, changes, branch, tags

Locks were needed to avoid conflicts in editing files where two people edited the same file and overwrote each others changes

Single file based like SCCS, stored revisions in filename,v

Revisions for normal mainline development are stored as reverse deltas, revisions for branches are stored as forward deltas from the branchpoint

A symbolic name is handy to refer to a specific released version (customer-1.0, etc)

CVS

- No locks
- Client / server
- Anonymous readonly access
- Previous de-facto open source RCS

Without locks the problem of overwriting each others changes arises. CVS circumvents this by only allowing commits based on the absolute latest version on a branch

Although SCCS was implemented as a client/server model as well, CVS was made to be much more accessible to various clients

The server could be told to allow anonymous access to revisions, which helped for remote people interested in development of software

CVS was in use for a long time; circa 1985 – 2005 and it is still used by large projects, such as OpenBSD

The “pressure” of better proprietary systems (Perforce, Bitkeeper, MS SourceSafe) caused open source enthusiasts on the internet to start developing alternatives

CVS, continued

- Revision information: who, when, changes, branch, tags
- `cvs log boot.c`

```
RCS file: /cvs/src/sys/arch/sparc/stand/boot/boot.c,v
Working file: boot.c
head: 1.6
branch:
locks: strict
access list:
symbolic names:
    OPENBSD_4_7: 1.6.0.26
        [...]
keyword substitution: kv
total revisions: 12;   selected revisions: 1
description:
----------------------------
revision 1.3
date: 2002/03/14 01:26:44;  author: millert;  state: Exp;  lines: +5 -5
First round of __P removal in sys
=============================================================================
```

Subversion

- Mission: “Build a better CVS”
- Atomic commits
- Copy files and directories
- Better branching, merging and tagging
- Current de-facto open source RCS

The explicit mission of making a better CVS was not universally accepted (as made famous by Linus Torvalds; more on that later)

CVS commits on a large source tree suffered from a race problem: every file committed separately.

Subversion did do a lot better on a lot of things though, such as tracking copies of files and directories which are represented in CVS as delete/add.

Also a lot better support branching and merging, simply because a changeset is recorded for multiple files at once.

Although, from my highly unscientific POV, a lot of projects are moving to one of the big 3 distributed RCSs

But it also tracks copies and recently became a lot better at tracking merges.

Subversion, continued

- Revision information: who, when, changes, branch, tags
- `svn log -v https://svn.win.tue.nl/repos/MCRL2/trunk`

```
------------------------------------------------------------------------
r7524 | wieger | 2010-03-18 11:38:17 +0100 (Thu, 18 Mar 2010) | 1 line
Changed paths:
   M /trunk/tools/pbespareqelm/pbespareqelm.cpp

- Fixed typo in option string.
```

Arch

- Mission: “Replace CVS”
- Bizarre command set and file naming conventions
  - `$ tla changes -o ,,new-robot-comment`
- But: distributed version control!
- Not in use anymore

Again, due to pressure from proprietary systems, and explicitly not repeating the CVS model

Although it might have made sense in exploring distributed version control, for people not really interested in RCSes this was a turn off

We will see what that means shortly; but this was Arch's main draw

Monotone

- Mission: “Replace CVS”
- Correctness, verifiable history
- Initially slow
- But: distributed version control!
- Still in use

Just like Arch, the Monotone developers set out to build a better RCS but not necessarily with all CVS features

Monotone was set up with a heavy emphasis on correctness and verifiable history. It uses certificates to sign revisions and log messages, as well as to identify remote repositories.

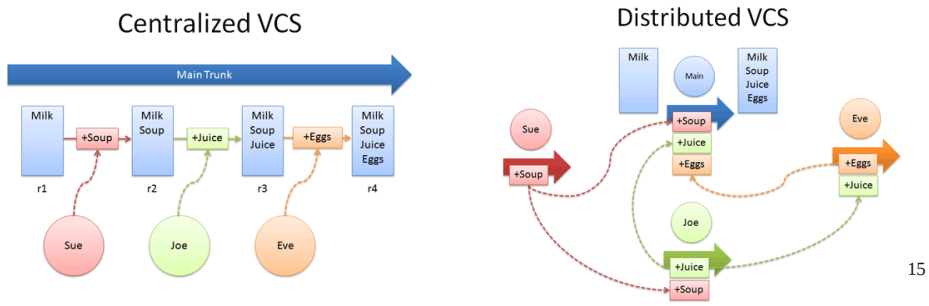
The emphasis on correctness and verifiable history made the initial implementation very slow, and the key management did not help either; so that turned people off

Again, like Arch, Monotone was set up for distributed version control from the start as it's main draw

The instant messenger client Pidgin still uses Monotone, not much else visible

Distributed vs Centralized

- Differences
- Distributed is “chaos”
- Resistance to change
- “Chaos” is manageable



15

Distributed revision control simply means that every developer has a complete copy of the repository available

As opposed to centralized, where there is a central repository server. This also implies that operations on the repository need access to the central server

Distributed development is potential chaos because every developer is essentially working on their own private copy, so features in one developer's trunk may not be present in anothers

Release managers make sure that releases are predictable and reproduceable, so having 100 copies is a no-no

It turns out that good development practices make it manageable

Distributed vs Centralized, ct'd

- “Throw away” development
- Delegate subprojects
- “Central” release repository

16

The fact is, that no code is ever committed 100% right the first time. Also, working on features that might not be interesting until there is a working prototype

Wouldn't it be nice to develop your good idea while still having history of your changes ready?

Or maybe you can delegate making a new feature to a sub team while keeping the trunk going; the sub team can then use their local copy to develop their own feature, granting other people access to their repository without impact on the main one

Note that all these features CAN be implemented using centralized RCS; however distributed makes it a lot easier

For release management, simply choose one of the distributed repositories and make a release there. De-facto “release” repositories will probably arise.

The current big three

- Bazaar
- Mercurial
- Git
- A lot of overlap

All are distributed
All have excellent branching and merging support, with only subtle differences
All systems store who, when, what, committer and author separately, tags, branches

Bazaar

- Evolved from Arch
- Mission: “Accessible DRCS”
- Cross-platform
- Slow initially, better now
- Projects: MySQL, Debian APT

Bazaar was sponsored by Canonical (of Ubuntu fame) when they recognized the potential for distributed development. It evolved from Arch through intermediate form called “Baz”

Bazaar was intended to convert users of centralized systems (notably CVS and Subversion) to distributed systems

Since it is implemented in Python it is cross-platform for free

The 1.x series was slow, which turned some people off again. This has been corrected in the 2.x releases.

Unique features of Bazaar

- Branches are simply new clones of repositories
- Very good support for graphical UIs on all platforms
- Tracks both file and directory renames explicitly

Bazaar log

- bzr log lp:bzr-svn

```
------------------------------------------------------------
revno: 3274
committer: Jelmer Vernooij <jelmer@samba.org>
branch nick: trunk
timestamp: Tue 2010-03-09 13:56:24 +0100
message:
  Provide BranchConfig._get_change_editor().
------------------------------------------------------------
revno: 3273
committer: Jelmer Vernooij <jelmer@samba.org>
branch nick: trunk
timestamp: Tue 2010-03-09 00:47:01 +0100
message:
  Fix use of svn:author=author.
```


Not shown here, and not every project uses this, but Bazaar can also track IDs for bugs solved internally.

Mercurial

- Mission: replace Bitkeeper
- Inspiration from Monotone
- Cross platform
- Projects: SUN Java, Mozilla Firefox, Python

Bitkeeper was the system of choice for a while for Linus Torvalds, creator of the Linux kernel. This was controversial because Bitkeeper was not open source software. In April 2005 the company behind Bitkeeper stopped the use of the license for many people developing the kernel. Hence a replacement was needed

Lots of concepts from Monotone are stolen by Mercurial, for example using SHA-1 hashes to identify a revision

Since Mercurial is written in Python, just like Bazaar it is cross-platform for free

Unique features of Mercurial

- Global branches
- Local revision numbers
- Built in server

Mercurial's “heavy” branches are global even though it is a distributed system. This is both a blessing and a curse since it means that the names of branches can conflict between multiple repositories. On the other hand it is easy to refer to a specific branch regardless of repository

For local reference, Mercurial also assigns a revision a number as well as the unique SHA1 hash

In order to quickly work together, and allow quick clones of a repository, it is easy to start a network server to allow people to do so

Mercurial log

- hg log mozilla

```
changeset:   15262:263749849d0b
tag:         tip
user:        Marco Zehe <marco.zehe@googlemail.com>
date:        Thu Jun 05 11:49:51 2008 +0200
summary:     b=432467, firefox segfaults in plone kupu editor
[@ nsDocAccessible::FlushPendingEvents], on Tablet PC
[@arena_dalloc_small] (steps to reproduce in comment #26),
p=Ginn Chen <ginn.chen@sun.com>, r=surkov

changeset:   15261:49cdeb4f8144
user:        Simon Montagu <smontagu@smontagu.org>
date:        Thu Jun 05 04:05:25 2008 -0700
summary:     Reftest for bug 229764
```

Git

- Mission: replace Bitkeeper
- Inspiration from Monotone
- Linux dependant at first
- Projects: Linux Kernel, Android, Wine, Perl

Just like Mercurial, Git was started due to the announcement that Bitkeeper was not free to use anymore

Also just like Mercurial, Git uses some concepts from Monotone like SHA-1 hashes for revisions

Git was implemented on Linux, initially using a lot of different languages. This made Git very hard to port to other platforms, especially the Windows platform. Nowadays Git has been slowly rewritten in C thereby making a Windows port available.

Unique features of Git

- Tracks file contents, not files
- Internal details highly visible

While Subversion introduced tracking file copies and renames, Git tracks contents instead. This allows Git to heuristically determine a function “move” from file1.c to file2.c and therefore show complete history for a function.

One of the most critized aspects of Git is that it is hard to learn, which is true. A lot of details about it are highly visible in commands. This has improved in recent times though.

Git log

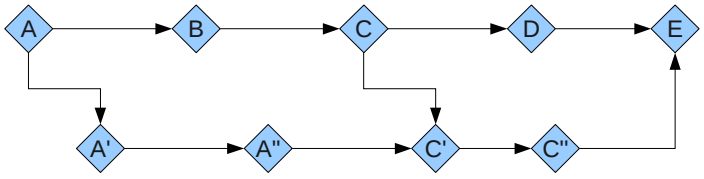
- git log HEAD

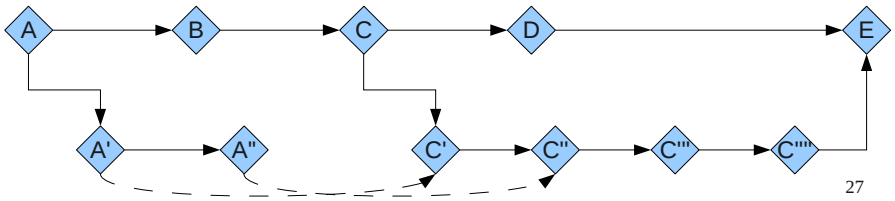
```
commit ef36c9a3b2828f5a11feda9e4d2708bf3a4a7a52
Author: Eric Anholt <eric@anholt.net>
Date:   Wed Mar 17 12:46:21 2010 -0700

    intel: Install the header file in the libdrm/ directory.

    Suggested-by: Rémi Cardona <remi@gentoo.org>
    Signed-off-by: Eric Anholt <eric@anholt.net>
```


Note that “suggested-by” and “signed-off-by” are not really git tracked fields as such; they are merely a loose convention that is used by multiple open source projects.

Advanced concepts

- Rebase versus merge





27

Although the term “rebase” is not always used in exactly the same way, it is usually a different concept than a merge.

Rebase “loses” information on which branch a feature is developed. Merge “remembers” the start of development on a branch.

Both are useful although not in the same scenario.

For “public” branches, that is branches that are consumed by other people instead of only one person or team, a merge is recommended.

For “private” branches or new, “separate” feature branches, a rebase is recommended.

This is because the system can use the merge information to make future merges easier. Rebasing is easier for private branches because the private history doesn't get “polluted” with merges that do not contribute to history information.

Advanced concepts, ct'd

- Octopus merge

Both Git and Bazaar can do so-called “octopus” merges. These are merges with multiple parent branches at the same time.

Mercurial users must merge branches two at a time.

Note that this is not a feature used often.

Further reading

- Original SCCS paper:
  http://basepath.com/aup/talks/SCCS-Slideshow.pdf
- History of RCS:
  http://www.burlingtontelecom.net/~ashawley/rcs/tichy1985rcs/rcs.html#id2504601
- Linux Kernel and BitKeeper:
  http://lkml.indiana.edu/hypermail/linux/kernel/0504.0/1540.html

  http://lkml.org/lkml/2005/4/7/150
- Distributed vs Centralized:
  http://blog.ianbicking.org/distributed-vs-centralized-scm.html

  http://web.mit.edu/ghudson/thoughts/bitkeeper.whynot