Git is MacGyver
Managing Kernel sources with Git

Chemnitzer Linux-Tage
13th and 14th March 2010
Kernel Track

Robert Richter <rric@kernel.org>

Git is MacGyver*

Who is MacGyver?

*Thanks to Patrick Thomson for the inspiration to this title
http://importantshock.wordpress.com/2008/08/07/git-vs-mercurial/

What has Git to do with MacGyver?

Maybe we all know this American television cult series from the 80's. MacGyver is the man with the Swiss army knife...

"MacGyver besitzt einen unglaublichen Einfallsreichtum, wenn es darum geht, mit Hilfe seiner handwerklichen Begabung und Alltagsgegenständen immer wieder Dinge zu basteln, mit denen er sich aus schwierigen Situationen befreit." -- de.wikipedia.org

"He prefers non-violent conflict resolution where possible, and refuses to use a gun. MacGyver works as a problem solver ..." -- en.wikipedia.org

Same is true for Git.

Content

- Git and the kernel
- Getting started with Git
- How Git works - some internals
- Working with kernel sources
- Kernel development with Git
- Cool stuff

Git is essential for kernel development

Per kernel version we have:

- more than 10.000 patches
- more than 1.000 developers
- almost 10.000 changed files, 1.000.000 new lines and 500.000 deleted lines

All these changes must be managed in an efficient way.

This is what Git is for...

Source code management (SCM) for kernel development written by Linus Torvalds in 2005

Git is used in kernel development to

- get kernel sources,
- track kernel changes,
- do kernel development.

All of this is also known as *version control*.

But Git is more, Git is a culture. It is how development is done.

Kernel statistics again

Per kernel version we have:

- more than 10.000 patches
- more than 1.000 developers
- almost 10.000 changed files, 1.000.000 new lines and 500.000 deleted lines

How to get these Numbers?

Getting the statistics ...

... is a Git command line one-liner:

Number of patches:

```
$ git log --pretty=oneline v2.6.32..v2.6.33 | wc -l
11684
```

Number of developers:

```
$ git shortlog --email v2.6.32..v2.6.33 -s | wc -l
1330
```

Number of changes (files, new and deleted lines):

```
$ git diff --shortstat v2.6.32..v2.6.33
 9673 files changed, 859407 insertions(+), 479401 deletions(-)
```

One more statistic slide

Top 20 developer of v2.6.33:

```
$ git shortlog --email v2.6.32..v2.6.33 -sn | head -n 20
   470        Linus Torvalds <torvalds@linux-foundation.org>
   152        Frederic Weisbecker <fweisbec@gmail.com>
   137        Arnaldo Carvalho de Melo <acme@redhat.com>
   130        Luis R. Rodriguez <lrodriguez@atheros.com>
   129        Masami Hiramatsu <mhiramat@redhat.com>
   124        Bartlomiej Zolnierkiewicz <bzolnier@gmail.com>
   115        Takashi Iwai <tiwai@suse.de>
   110        Ben Hutchings <bhutchings@solarflare.com>
   110        Eric Dumazet <eric.dumazet@gmail.com>
   110        Paul Mundt <lethal@linux-sh.org>
   104        Alan Cox <alan@linux.intel.com>
   104        David S. Miller <davem@davemloft.net>
   102        Manu Abraham <abraham.manu@gmail.com>
   101        Thomas Gleixner <tglx@linutronix.de>
  ...
```

Getting started- installing Git

Project page: http://git-scm.com/

Current version: v1.7.0.1

Distro packages:

- Debian based: git-core, gitk, (gitweb)
- Fedora: git, gitk, (git-email), (gitweb), git-all (git-arch git-cvs git-email git-gui gitk git-svn perl-Git)
- OpenSuse: git/git-core
- Gentoo: dev-util/git (includes gitk and gitweb)

man pages: git, gittutorial, gitglossary

Git is rich in commands

git-add git-am git-annotate git-apply git-archimport git-archive git-bisect git-blame git-branch git-bundle git-cat-file git-check-attr git-checkout git-checkout-index git-check-ref-format git-cherry git-cherry-pick git-citool git-clean git-clone git-commit git-commit-tree git-config git-count-objects git-cvsexportcommit git-cvsimport git-cvsserver git-daemon git-describe git-diff git-diff-files git-diff-index git-difftool git-diff-tree git-fast-export git-fast-import git-fetch git-fetch-pack git-filter-branch git-fmt-merge-msg git-for-each-ref git-format-patch git-fsck git-gc git-get-tar-commit-id git-grep git-gui git-hash-object git-help git-http-backend git-http-fetch git-http-push git-imap-send git-index-pack git-init git-instaweb gitk git-log git-lost-found git-ls-files git-ls-remote git-ls-tree git-mailinfo git-mailsplit git-merge git-merge-base git-merge-file git-merge-index git-merge-one-file git-mergetool git-merge-tree git-mktag git-mktree git-mv git-name-rev git-notes git-pack-objects git-pack-redundant git-pack-refs git-parse-remote git-patch-id git-peek-remote git-prune git-prune-packed git-pull git-push git-quiltimport git-read-tree git-rebase git-receive-pack git-reflog git-relink git-remote git-repack git-replace git-repo-config git-request-pull git-rerere git-reset git-revert git-rev-list git-rev-parse git-rm git-send-email git-send-pack git-shell git-shortlog git-show git-show-branch git-show-index git-show-ref git-sh-setup git-stash git-status git-stripspace git-submodule git-svn git-symbolic-ref git-tag git-tar-tree git-unpack-file git-unpack-objects git-update-index git-update-ref git-update-server-info git-upload-archive git-upload-pack git-var git-verify-pack git-verify-tag git-whatchanged git-write-tree

... but only 15 commands to know

```
git config
git log
git init
git add
git grep
git commit
git clone
gitk
```

```
git fetch
git merge
git branch
git blame
git rebase
git show
git diff
```

Quick start - creating local repositories

Create a new repository:

```
$ cd foobar
$ git init
$ git add .
$ git commit -sm 'Initial checkin'
```

Remote repositories can be added later too.

Quick start - clone an existing repository

Clone kernel mainline:

```
$ git clone \
  git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux-2.6.git
$ ... # edit some files
$ git add <changed file>
$ git commit -m "Don't explain what, explain why."
$ git format-patch origin/master
```

see man gittutorial, http://git-scm.com/

Configuring Git

Environment variables:

```
$ printenv | grep GIT
GIT_COMMITTER_NAME=Robert Richter
GIT_COMMITTER_EMAIL=robert.richter@amd.com
```

Files: .git/config, etc.

With git config (man git-config)

But... Better edit config files because they are supposed to be edited and git-config is supposed to be the documentation (man git-config).

Get status, browse history, show changes

```
$ git status
$ git show
$ git show --stat
$ git log
$ git log --pretty=oneline --abbrev-commit -10
$ git log -p
$ git shortlog v2.6.32..v2.6.33
$ git diff
$ git diff --cached # changes to be committed
$ git diff HEAD     # show all changes in the working tree
$ git diff --stat
$ git blame <file>  # find commits that changed <file>
```

Gitk - git repository browser

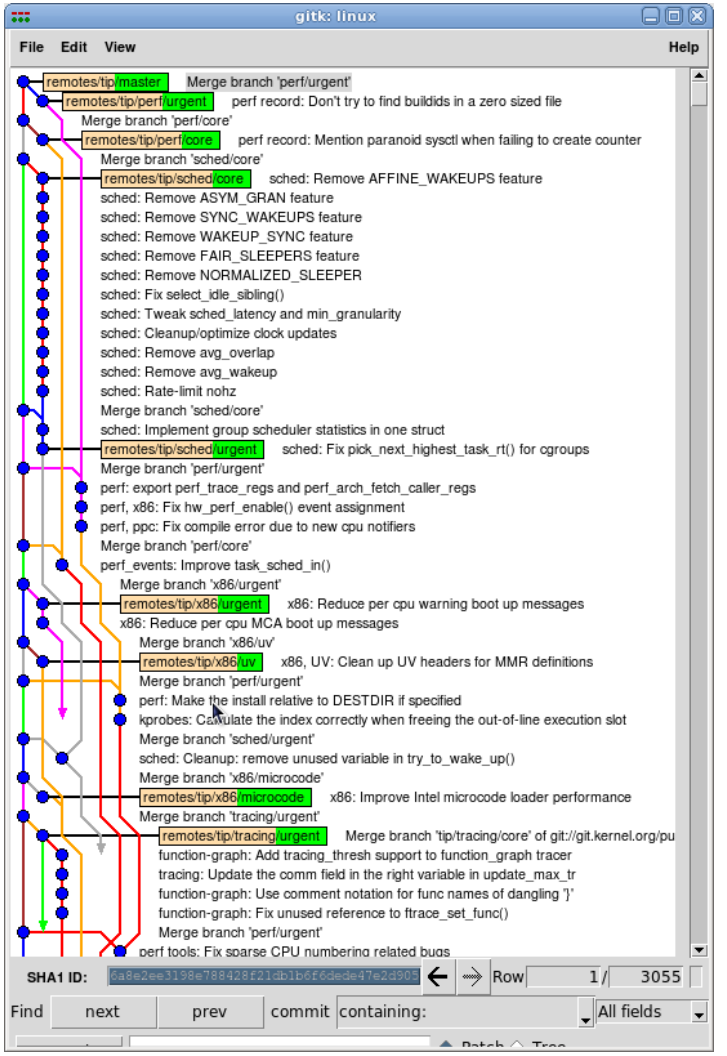
```
$ gitk --all

$ gitk ORIG_HEAD..

$ gitk linux-2.6/master..

$ gitk \
v2.6.33..origin/master

$ gitk \
linux-2.6/master..tip/master
```



Quick start - development with Git

Update and show branches:

```
$ git fetch origin # update remote branches from 'origin'
$ git branch -a    # show all local and remote branches
```

Develop on a feature branch:

```
$ git checkout -b new_feature  # create feature branch
$ ...                          # do some changes
$ git commit ...               # commit on branch new_feature
$ git checkout master
$ git merge new_feature    # merge new_feature into master
```

Rebase own changes to latest mainline kernel:

```
$ git rebase linux-2.6/master
```

And, that's it! This is daily Git usage.

Remote repositories

Git repos that can be fetched from and pulled to.

Repository URLs (see man git-clone):

```
git://host.xz[:port]/path/to/repo.git/
http://host.xz[:port]/path/to/repo.git/
https://host.xz[:port]/path/to/repo.git/
rsync://host.xz/path/to/repo.git/
ssh://[user@]host.xz[:port]/path/to/repo.git/
[user@]host.xz:/path/to/repo.git/
/path/to/repo.git/
file:///path/to/repo.git/
```

The working tree (local repository)

- contains checked out files (`GIT_WORK_TREE`)
- “clean” if no changes, otherwise “dirty”
- The root repository (.git is default) contains repository data (root repository, `GIT_DIR`).
- Bare repositories do not have a working tree.
- Use .gitignore to specify untracked files
- No, no, no. I don't want Git anymore!

```
$ rm -rf .git # Of course it is just a hypothetical construct
```

The Index

Contains changes to be commited.

Command that may change the index:

```
$ git add ...
$ git rm ...
$ git reset ...
$ git checkout ...
```

Commit changes:

```
$ git commit -s
$ git commit -c ...
$ git commit -asm "Don't explain what, explain why."
```

Reset changes

Reset repository revision, not the working tree:

```
$ git reset              # index only
$ git reset commitid     # repository revision
```

Reset the working tree

```
$ git checkout <commitid> file # set file to version
$ git reset --hard             # discard all changes
```

Reset the working tree and repository revision:

```
$ git reset --hard ORIG_HEAD
$ git reset --hard <commitid>
$ git checkout <branch>
```

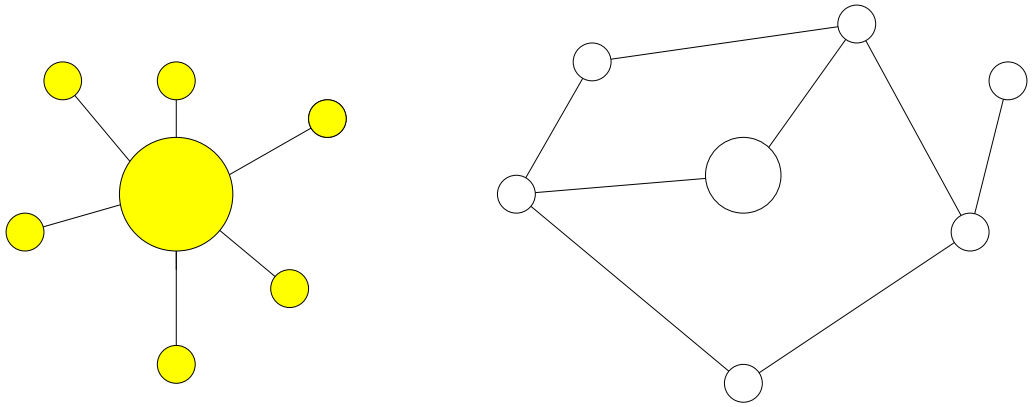
What is Git designed for?

"If you're not distributed, you are not worth using, it's that simple. If you perform badly, you are not worth using, it is that simple. And if you cannot guarantee that the stuff I put into an SCM comes out exactly the same, you are not worth using." --Linus Torvalds

- distributed
- efficient with good performance even on huge projects
- stable and worth to trust your data

Why distribution is so important

"Being distributed very much means that you do not have one central location that keeps track of your data." --Linus Torvalds



Distribution - How about pulling from me?

- more than just offline work
- "You don't have to worry about the commit access thing": If you have a distributed model, everybody has commit rights.
- You can do whatever you want with your project. You can do great or stupid work. How about pulling from me? ... and, people do!
- Distribution means branching. Every people has it's own branch. Branches are not global.

Distribution (cont'd) - Please, do not disturb others

- Commit changes without disturbing others
- You don't need commiting rules
- Others don't have to wait for you
- So you get small commits
- hundreds or thousands of people working on the same project
- get more trustworthy, no single point of failure, 'perfect' load balancing
- see Tech Talk: Linus Torvalds on git

Network of trust

There are 1000 of repositories, from whom to pull?

Network of trust is the only way you can do development.

"I don't have to trust you. I only need to trust 5, 10, 15 people... I can pull from them... They have other people... This is how it all comes together." --Linus Torvalds

You know that there are 5 branches that are really interesting to follow...

It fits how we are wired up.

If everybody has its own branch, which version do I have?

A repository state is identified by an SHA-1 hash:

```
$ git show v2.6.33~0
commit 60b341b778cc2929df16c0a504c91621b3c6a4ad
Author: Linus Torvalds <torvalds@linux-foundation.org>
Date:   Wed Feb 24 10:52:17 2010 -0800

    Linux 2.6.33

diff --git a/Makefile b/Makefile
index 12b1aa1..1b24895 100644
--- a/Makefile
+++ b/Makefile
@@ -1,7 +1,7 @@
 VERSION = 2
 PATCHLEVEL = 6
 SUBLEVEL = 33
-EXTRAVERSION = -rc8
+EXTRAVERSION =
 NAME = Man-Eating Seals of Antiquity

 # *DOCUMENTATION*
```

Naming commits to something human readable

Describing and specifying revisions:

- Built-in: **`HEAD, FETCH_HEAD, ORIG_HEAD and MERGE_HEAD`**
- Branches: **`git branch -a`**
- Tags:

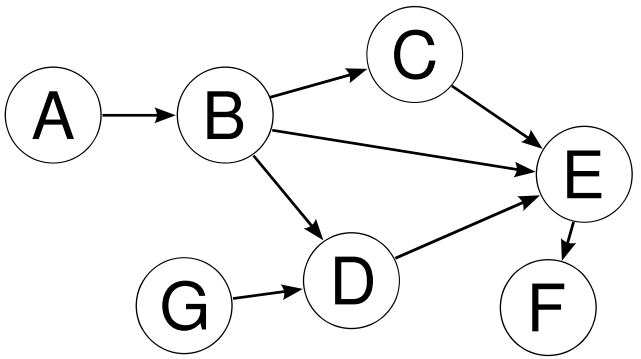
```
$ git tag
v2.6.33
[...]
v2.6.33-rc8
```

- Suffixes: master@{1}, tags/v2.6.32-rc3~5^2~2

see man git-rev-parse, git-name-rev

SHA-1 hashes and how Git works internally.

- binary objects in an object database
- identified, named and checked by its SHA-1
- Commit objects form a Directed Acyclic Graph (DAG, gerichteter azyklischer Graph)
- My history is gone! No, it isn't!



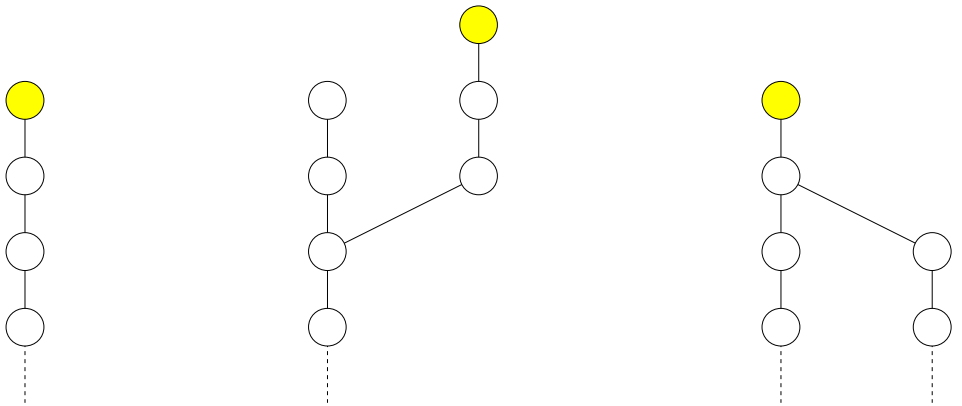
see:
http://eagain.net/articles/git-for-computer-scientists/
http://de.wikipedia.org/wiki/Graph_(Graphentheorie))

Git objects

- blob (content of a file)
- tree (file names and directory structure)
- commit (describes a repository state), contains a (possibly empty) list of parents
- tag (a ref pointing to a commit object)

Commit, branch, merge...

Commit parents and childrens:



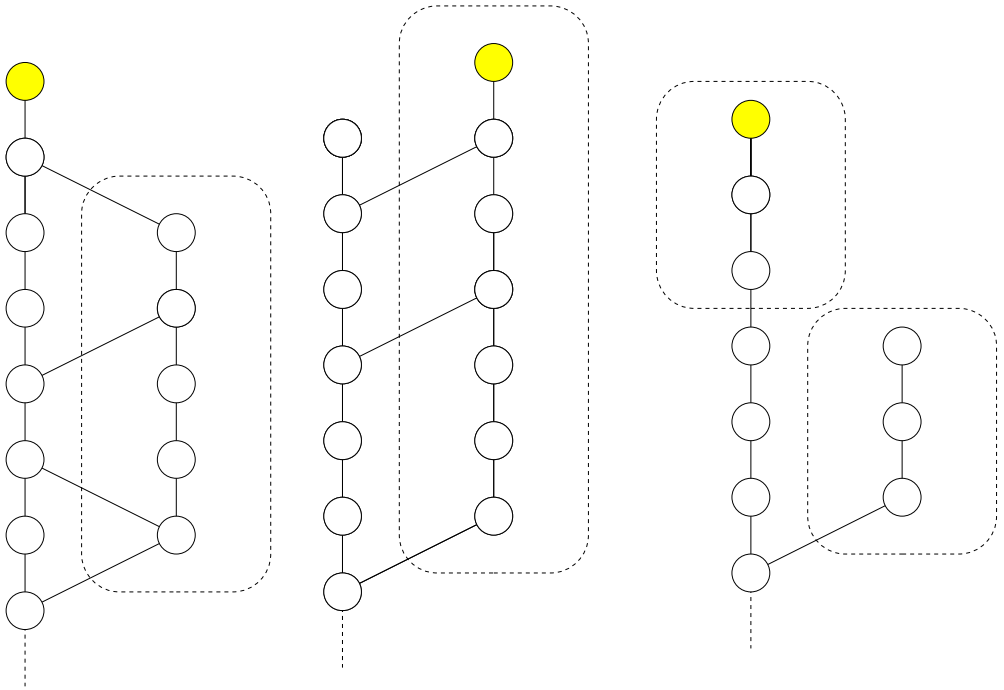
Rewriting History - Historical revisionism

"If all records told the same tale — then the lie passed into history and became truth." --Nineteen Eighty-Four, George Orwell

- This is not 1984; you cannot rewrite history. Of course you can, a rebase changes history.
- Patches must be rebased if there could be conflicts when applying.
- "Rebase is random things happen and I am not surprised." --Linus Torvalds

Why rebasing?

continuous patch sets, avoid conflicts, clean history



Other ways to rewrite history

Change or apply single commits:

```
git cherry-pick
git commit --amend
git commit --amend -a
git commit --amend --reset-author
git commit --amend -c COMMITID
```

When to merge and when to rebase?

Do not merge maintainers code.

Don't do merges downstream.

This avoids criss cross merges.

Rebase only as an endpoint person.

Do not rebase public development trees.

Resolving conflicts

"Git is much smarter than most people." --Linus Torvalds

Caused by: git merge, git rebase, git cherry-pick

Show conflict:

```
$ git status                    # show conflicting files
$ git diff                      # 3 way diff showing conflicts
```

Edit and add, then continue, commit or abort:

```
$ git add .
$ git rebase --continue         # if rebasing
$ git commit                    # merge, cherry-pick
$ git reset --hard              # merge not commited
$ git reset --hard ORIG_HEAD    # merge commited
$ git rebase --abort
```

Avoiding conflicts

- specify merge strategy
- small patch sets
- release early and often
- separate cleanups, make them in the beginning of the patch set
- apply major changes to both branches before doing a merge
- use tools (sed, perl, ...) to rename or cleanup automatically
- rebase in single step

Update from and to remote repositories

Fetch and pull again

```
$ git fetch                              # update remote branch
$ git pull --ff-only origin master # ... and HEAD
```

A pull is a fetch and a merge.

Rebase HEAD if non-fast forward of fetch-only:

```
$ git rebase origin/master
```

Push to remote repository

```
$ git push origin master
```

Fast-Forward ... we have not done anything

Non-fast Forward: The remote branch from which you want to pull is modified in non-linear ways such as being rewound and rebased frequently. A merge results in a 3-way merge.

This means, the new non-fast forward branch does not contain all commits of the old branch. These commits would get lost.

Interactive Git

“Killer feature”:

Join patches, edit commit id, reorder them:

```
$ git rebase -i
```

Add single hunks to be commited:

```
$ git add -i
```

Working with the kernel

Getting kernel sources, do you remember?

```
$ wget -nd \
http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.33.tar.bz2
$ tar -xzf linux-2.6.33.tar.bz2
```

Here comes git:

```
$ git fetch \
git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux-2.6.git \
master
$ git checkout v2.6.33
...
HEAD is now at 60b341b... Linux 2.6.33
```

Working with kernel repositories

Manage repositories with:

$ git remote ...

Important trees (see http://git.kernel.org/):

git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux-2.6.git

git://git.kernel.org/pub/scm/linux/kernel/git/stable/linux-2.6-stable.git

git://git.kernel.org/pub/scm/linux/kernel/git/sfr/linux-next.git

git://git.kernel.org/pub/scm/linux/kernel/git/tip/linux-2.6-tip.git

git://git.kernel.org/pub/scm/linux/kernel/git/kyle/linux-2.6-snaps.git

(mm-tree is not maintained with Git)

Who changed the code, when and why? What has been changed?

```
$ git blame ...

$ git annotate file

$ git show ...

$ git annotate <commitid> file

$ git show ...
```

You found a bug and want to fix it? Send a patch!

First, check who maintains the code (`MAINTAINERS`):

```
$ ./scripts/get_maintainer.pl -f drivers/oprofile/event_buffer.c
Robert Richter <robert.richter@amd.com>
oprofile-list@lists.sf.net
linux-kernel@vger.kernel.org
```

Use the maintainers repository:

```
description OProfile kernel tree
owner       Robert Richter
last change Mon, 1 Mar 2010 13:21:23 +0000
URL git://git.kernel.org/pub/scm/linux/kernel/git/rric/oprofile.git
    http://git.kernel.org/pub/scm/linux/kernel/git/rric/oprofile.git

$ git remote add oprofile \
  git://git.kernel.org/pub/scm/linux/kernel/git/rric/oprofile.git
$ git fetch oprofile
$ git rebase oprofile/core
```

Sign your work

Add a Signed-off-by tag to your commit messages (see Documentation/SubmittingPatches):

```
Signed-off-by: Robert Richter <robert.richter@amd.com>
```

The sign-off is a simple line at the end of the explanation for the patch, which certifies that you wrote it or otherwise have the right to pass it on as a open-source patch.

You can add a Signed-off-by to the current HEAD using:

```
$ git commit --amend -s
```

Build it, test it, send it

(in that order, if possible)

Compile and test your changes with various relevant configurations.

Submit the patch to the mailing list

```
$ git send-email ...
```

How the patch goes upstream

- Patch is reviewed and discussed on the mailing list
- May be an updated version is resubmitted
- Eventually the maintainer marks it for the stable tree (Cc: <stable@kernel.org>)
- The maintainer applies the patch to a proper branch
- The branch is merged into for-next to detect conflicts

How the patch goes upstream (cont'd)

- Somewhen this tree is merged in the maintainer's mainline
- A for-linus branch is created
- A [GIT PULL] is sent to Linus for updates during the merge window, or anytime for fixes
- Linus pulls it, the patch is upstream

Cool stuff

"A word that is a proper subsitute for any other word. Present most often in one sided conversations, or when a conversation is running dry,the random insertion of 'stuff' breaks the silence for as long as it takes to say the word." --www.urbandictionary.com

Shrinking object directory size alternative object directories:

```
$ cat .git/objects/info/alternates
/home/robert/dev/linux/.git/objects
$ git gc
$ du -s .git /home/robert/dev/linux/.git
39464                .git
707808               /home/robert/dev/linux/.git
```

Cool stuff (cont'd)

Apply patches with mutt:

```
|cd linux; git am -3 -
```

Single step rebase:

```
$ for NUM in $(seq 48 -1 0); \
  do git rebase tip/perf/core~$NUM  || break; \
  done
```

Commands I can't resist to tell you

```
$ git rebase some_branch HEAD~0         # working without branches
$ git rebase --onto fixes master~10 master~5 # rip out patches

$ git merge --no-commit master
$ git merge --ff-only

$ git push . HEAD:testing_branch -f     # "Copy" a branch
$ git push . :some_branch               # Delete a branch

$ git commit -c HEAD --reset-author -a # reusing a commit message

$ git bisect ...

$ git filter-branch ...
```

Using Git with other SCMs

"I know you're out there. I can feel you now. I know that you're afraid. You're afraid of us. You're afraid of change."
--Neo, The Matrix

Git is MacGyver, and Mercurial is James Bond, convert repsitories with:

```
hg-fast-export, Hg-Git plugin for Mercurial
```

If you have to use Subversion, CVS, etc. this is that you are looking for:

```
git-svn, git-cvsupdate, git quiltimport
```

Other Git use cases

Tracking /etc with Git

Synchronizing your own files on multiple computers (home, work, laptop, remote server, etc.)

Build regressions (synchronizing, report and archive build sources)

Operate a Git server with git-daemon and Gitweb as on git.kernel.org.

Now we know...

Git is a problem solver.

Git is Open Source.

Git is easy.

Git is distributed.

Git is efficient.

Git is stable.

Git is a culture.

Git is much smarter than most people.

Git is writing history.

Git is cool.

Git is kernel development.

Git is MacGyver

Index

version control

remote repository

working tree (local repository), clean/dirty

bare repositories

index

repository root (git dir)

commit, parent/child

fast-forward branches

References

- Tech Talk: Linus Torvalds on git, http://www.youtube.com/watch?v=4XpnKHJAok8
- man pages: git, gittutorial, gitglossary
- http://git-scm.com/
- wikipedia.org
- http://importantshock.wordpress.com/2008/08/07/git-vs-mercurial/
- http://www.computerworld.com/s/article/101207/After_open_source_controversy_Torvalds_turns_to_git_