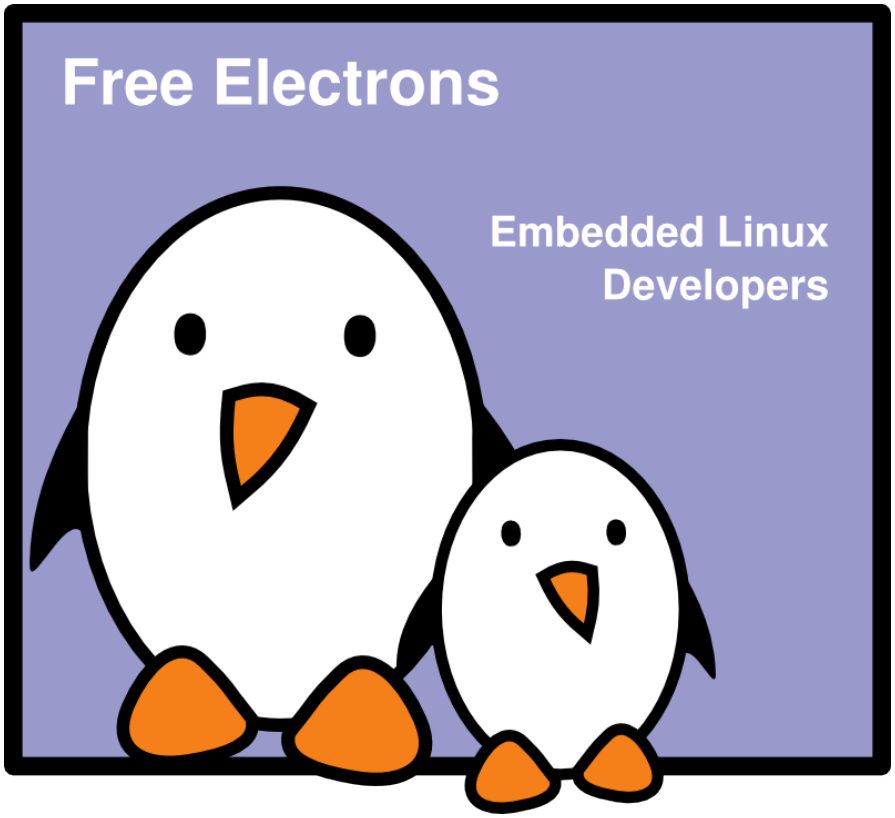
Introduction to Git

Thomas Petazzoni
Free Electrons



Latest update: Dec 24, 2009,
Document sources, updates and translations:
http://free-electrons.com/docs/git
Corrections, suggestions, contributions and translations are welcome!

What is Git?

- A version control system, like CVS, SVN, Perforce or ClearCase
- Originally developed for the Linux kernel development, now used by a large number of projects, including U-Boot, GNOME, Buildroot, uClibc and many more
- Contrary to CVS or SVN, Git is a *distributed* version control system
  - No central repository
  - Everybody has a local repository
  - Local branches are possible, and very important
  - Easy exchange of code between developers
  - Well-suited to the collaborative development model used in open-source projects

Install and setup

- Git is available as a package in your distribution
  `sudo apt-get install git-core`
- Everything is available through the `git` command
  - `git` has many commands, called using `git <command>`, where `<command>` can be `clone`, `checkout`, `branch`, etc.
  - Help can be found for a given `command` using
    `git help <command>`
- Setup your name and e-mail address
  - They will be referenced in each of your commits
  - `git config --global user.name 'My Name'`
  - `git config --global user.email me@mydomain.net`

Clone a repository

- To start working on a project, you use Git's `clone` operation.
- With CVS or SVN, you would have used the `checkout` operation, to get a working copy of the project (latest version)
- With Git, you get a full copy of the repository, including the history, which allows to perform most of the operations offline.
- Cloning Linus Torvalds' Linux kernel repository
  ```
  git clone git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux-2.6.git
  ```
- `git://` is a special Git protocol. Most repositories can also be accessed using `http://`, but this is slower.
- After cloning, in `linux-2.6/`, you have the repository and a working copy of the *master* branch.

Explore the history

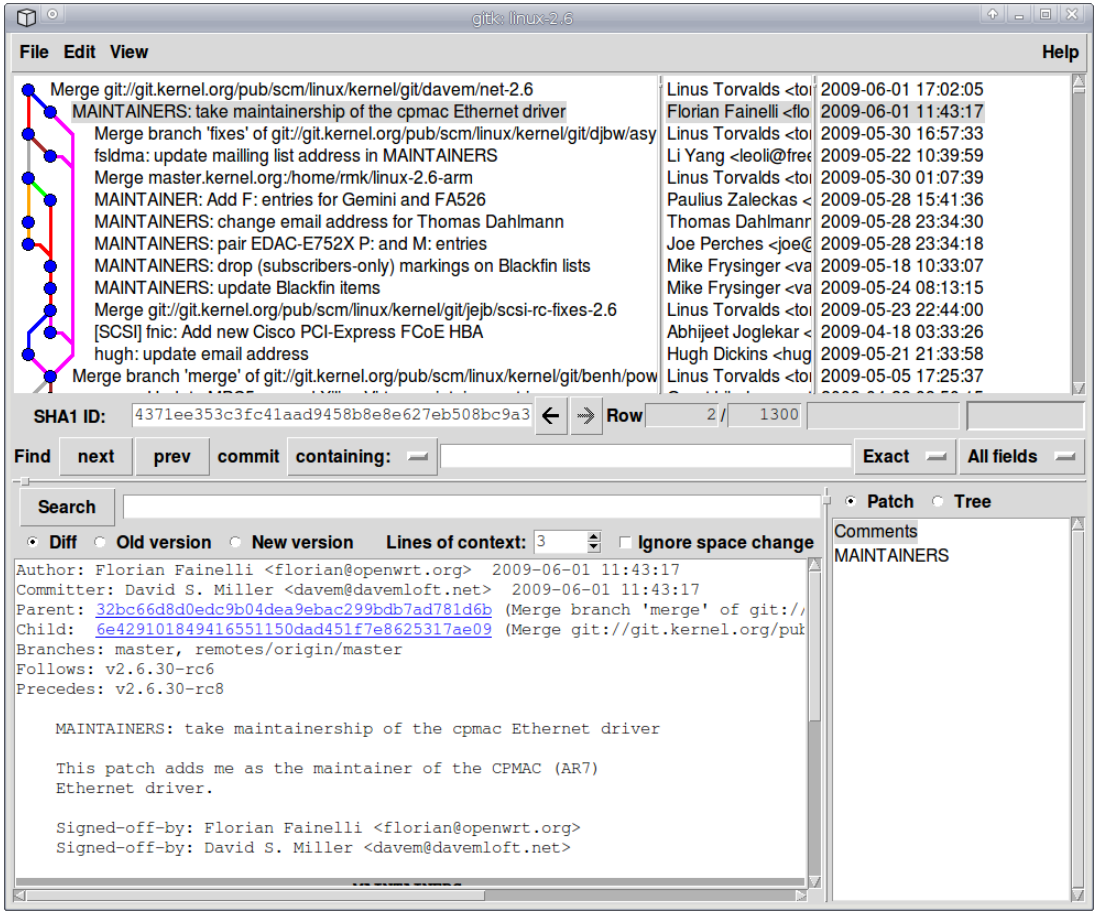
- `git log` will list all the commits. The latest commit is the first.

```
commit 4371ee353c3fc41aad9458b8e8e627eb508bc9a3
Author: Florian Fainelli <florian@openwrt.org>
Date:   Mon Jun 1 02:43:17 2009 -0700

    MAINTAINERS: take maintainership of the cpmac Ethernet driver

    This patch adds me as the maintainer of the CPMAC (AR7)
    Ethernet driver.

    Signed-off-by: Florian Fainelli <florian@openwrt.org>
    Signed-off-by: David S. Miller <davem@davemloft.net>
```

- `git log -p` will list the commits with the corresponding diff
- The history in Git is not linear like in CVS or SVN, but it is a graph of commits
  - Makes it a little bit more complicated to understand at the beginning
  - But this is what allows the powerful features of Git (distributed, branching, merging)

Visualize the history

- *gitk* is a graphical tool that represents the history of the current Git repository
- Can be installed from the `gitk` package



Visualize the history

- Another great tool is the Web interface to Git. For the kernel, it is available at http://git.kernel.org/

/pub/scm / linux/kernel/git/torvalds/linux-2.6.git / commitdiff git

summary | shortlog | log | commit | commitdiff | tree
raw (merge: 8623661 84047e3) commit search:

Merge branch 'tracing-urgent-for-linus' of git://git.kernel.org/pub/scm/linux/kernel ... master

Linus Torvalds [Thu, 11 Jun 2009 02:58:10 +0000 (19:58 -0700)]

```
* 'tracing-urgent-for-linus' of git://git.kernel.org/pub/scm/linux/kernel/git/tip/linux-2.6-tip:
  function-graph: always initialize task ret_stack
  function-graph: move initialization of new tasks up in fork
  function-graph: add memory barriers for accessing task's ret_stack
  function-graph: enable the stack after initialization of other variables
  function-graph: only allocate init tasks if it was not already done

Manually fix trivial conflict in kernel/trace/ftrace.c
```

kernel/fork.c	patch | blob | history
kernel/trace/ftrace.c	patch | blob | history
kernel/trace/trace_functions_graph.c	patch | blob | history

diff --git a/kernel/fork.c b/kernel/fork.c

```
index 5449efb..bb762b4 100644 (file)
--- a/kernel/fork.c
+++ b/kernel/fork.c
@@ -981,6 +981,8 @@ static struct task_struct *copy_process(unsigned long clone_flags,
        if (!p)
                goto fork_out;

+       ftrace_graph_init_task(p);
+
        rt_mutex_init_task(p);

 #ifdef CONFIG_PROVE_LOCKING
```

Update your repository

- The repository that has been cloned at the beginning will change over time
- Updating your local repository to reflect the changes of the remote repository will be necessary from time to time
- `git pull`
- Internally, does two things
  - Fetch the new changes from the remote repository (`git fetch`)
  - Merge them in the current branch (`git merge`)

Tags

- The list of existing tags can be found using
  `git tag -l`
- To check out a working copy of the repository at a given tag
  `git checkout <tagname>`
- To get the list of changes between a given tag and the latest available version
  `git log v2.6.30..master`
- List of changes with diff on a given file between two tags
  `git log v2.6.29..v2.6.30 MAINTAINERS`
- With gitk
  `gitk v2.6.30..master`

Branches

- To start working on something, the best is to make a branch
  - It is local-only, nobody except you sees the branch
  - It is fast
  - It allows to split your work on different topics, try something and throw it away
  - It is cheap, so even if you think you're doing something small and quick, do a branch
- Unlike other version control systems, Git encourages the use of branches. Don't hesitate to use them.

Branches

- Create a branch
  `git branch <branchname>`
- Move to this branch
  `git checkout <branchname>`
- Both at once (create and switch to branch)
  `git checkout -b <branchname>`
- List of local branches
  `git branch -l`
- List of all branches, including remote branches
  `git branch -a`

Making changes

- Edit a file with your favorite text editor
- Get the status of your working copy
  `git status`
- Git has a feature called the *index*, which allows you to stage your commits before committing them. It allows to commit only part of your modifications, by file or even by chunk.
- On each modified file
  `git add <filename>`
- Then commit. No need to be online or connected to commit.
  `git commit`
- If all modified files should be part of the commit
  `git commit -a`

Sharing changes by e-mail

- The simplest way of sharing a few changes is to send patches by e-mail
- First step is to generate the patches
  `git format-patch -n master..<yourbranch>`
  - Will generate one patch for each of the commits done on `<yourbranch>`
  - The patch files will be 0001-...., 0002-...., etc.
- Second step is to send these patches
  `git send-email --compose --to email@domain.com 00*.patch`
  - Assumes that the local mail system is properly configured
    Needs the `git-email` package in Ubuntu.
  - Or `git config` allows to set the SMTP server, port, user and password if needed

Sharing changes: your own repository

- If you do a lot of changes and want to ease collaboration with others, the best is to have your own repository
- Create a *bare* version of your repository
```
cd /tmp
git clone --bare ~/project project.git
touch project.git/git-daemon-export-ok
```
- Transfer the contents of `project.git` to a publicly-visible place (reachable read-only by HTTP for everybody, and read-write by you through SSH)
- Tell people to clone http://yourhost.com/path/to/project.git
- Push your changes using
```
git push ssh://yourhost.com/path/toproject.git
srcbranch:destbranch
```

Tracking remote trees

- In addition to the official Linus Torvalds tree, you might want to use other development or experimental trees
  - The OMAP tree at `git://git.kernel.org/pub/scm/linux/kernel/git/tmlind/linux-omap-2.6.git`
  - The realtime tree at `git://git.kernel.org/pub/scm/linux/kernel/git/rostedt/linux-2.6-rt.git`
- The `git remote` command allows to manage remote trees

```
git remote add rt \
git://git.kernel.org/pub/scm/linux/kernel/git/rostedt/linux-2.6-rt.git
```

- Get the contents of the tree

```
git fetch rt
```

- Switch to one of the branches

```
git checkout rt/master
```

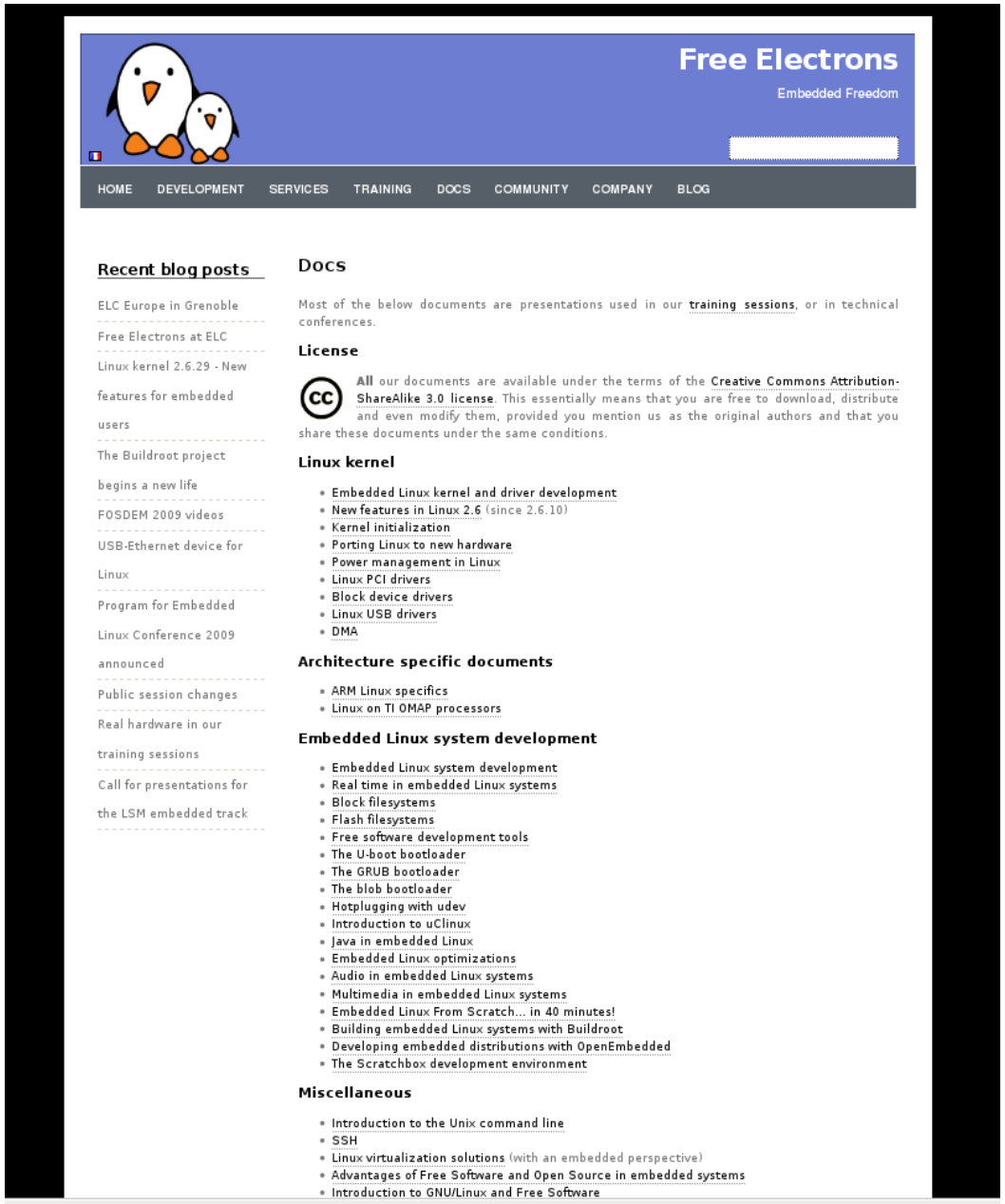
About Git

- We have just seen the very basic features of Git.
  A lot more interesting features are available (rebasing, bisection, merging and more)
- References
  - Git Manual
    http://www.kernel.org/pub/software/scm/git/docs/user-manual.html
  - Git Book
    http://book.git-scm.com/
  - Git official website
    http://git-scm.com/
  - James Bottomley's tutorial on using Git
    http://free-electrons.com/pub/video/2008/ols/ols2008-james-bottomley-git.ogg

Practical lab – Git

- Clone a Git repository and explore history
- Make and share changes to a project managed in Git

Related documents



Free Electrons
Embedded Freedom

HOME DEVELOPMENT SERVICES TRAINING DOCS COMMUNITY COMPANY BLOG

Recent blog posts

- ELC Europe in Grenoble
- Free Electrons at ELC
- Linux kernel 2.6.29 - New features for embedded users
- The Buildroot project begins a new life
- FOSDEM 2009 videos
- USB-Ethernet device for Linux
- Program for Embedded Linux Conference 2009 announced
- Public session changes
- Real hardware in our training sessions
- Call for presentations for the LSM embedded track

Docs

Most of the below documents are presentations used in our training sessions, or in technical conferences.

License

All our documents are available under the terms of the Creative Commons Attribution-ShareAlike 3.0 license. This essentially means that you are free to download, distribute and even modify them, provided you mention us as the original authors and that you share these documents under the same conditions.

Linux kernel

- Embedded Linux kernel and driver development
- New features in Linux 2.6 (since 2.6.10)
- Kernel initialization
- Porting Linux to new hardware
- Power management in Linux
- Linux PCI drivers
- Block device drivers
- Linux USB drivers
- DMA

Architecture specific documents

- ARM Linux specifics
- Linux on TI OMAP processors

Embedded Linux system development

- Embedded Linux system development
- Real time in embedded Linux systems
- Block filesystems
- Flash filesystems
- Free software development tools
- The U-boot bootloader
- The GRUB bootloader
- The blob bootloader
- Hotplugging with udev
- Introduction to uClinux
- Java in embedded Linux
- Embedded Linux optimizations
- Audio in embedded Linux systems
- Multimedia in embedded Linux systems
- Embedded Linux From Scratch... in 40 minutes!
- Building embedded Linux systems with Buildroot
- Developing embedded distributions with OpenEmbedded
- The Scratchbox development environment

Miscellaneous

- Introduction to the Unix command line
- SSH
- Linux virtualization solutions (with an embedded perspective)
- Advantages of Free Software and Open Source in embedded systems
- Introduction to GNU/Linux and Free Software

All our technical presentations on http://free-electrons.com/docs

- Linux kernel
- Device drivers
- Architecture specifics
- Embedded Linux system development

How to help

You can help us to improve and maintain this document...

- By sending corrections, suggestions, contributions and translations
- By asking your organization to order development, consulting and training services performed by the authors of these documents (see http://free-electrons.com/).
- By sharing this document with your friends, colleagues and with the local Free Software community.
- By adding links on your website to our on-line materials, to increase their visibility in search engine results.

Linux kernel

Linux device drivers
Board support code
Mainstreaming kernel code
Kernel debugging

Embedded Linux Training

All materials released with a free license!

Unix and GNU/Linux basics
Linux kernel and drivers development
Real-time Linux, uClinux
Development and profiling tools
Lightweight tools for embedded systems
Root filesystem creation
Audio and multimedia
System optimization

Free Electrons

Our services

Custom Development

System integration
Embedded Linux demos and prototypes
System optimization
Application and interface development

Consulting and technical support

Help in decision making
System architecture
System design and performance review
Development tool and application support
Investigating issues and fixing tool bugs