git

Linus' new version control system

CVS problems

- slow, cumbersome
- tied to one central computer
- can’t easily change names
- can’t easily rearrange directories
- file oriented

git improvements

- repository is distributed (copied everywhere)
- fast (partly because repository is local)
- distribution oriented: changes across directories are checked in at once with one message
- most renames and reorganizations are easy
- work conveniently unconnected from network
- branches are easy, quick, and convenient
- much more capable than cvs

The price?

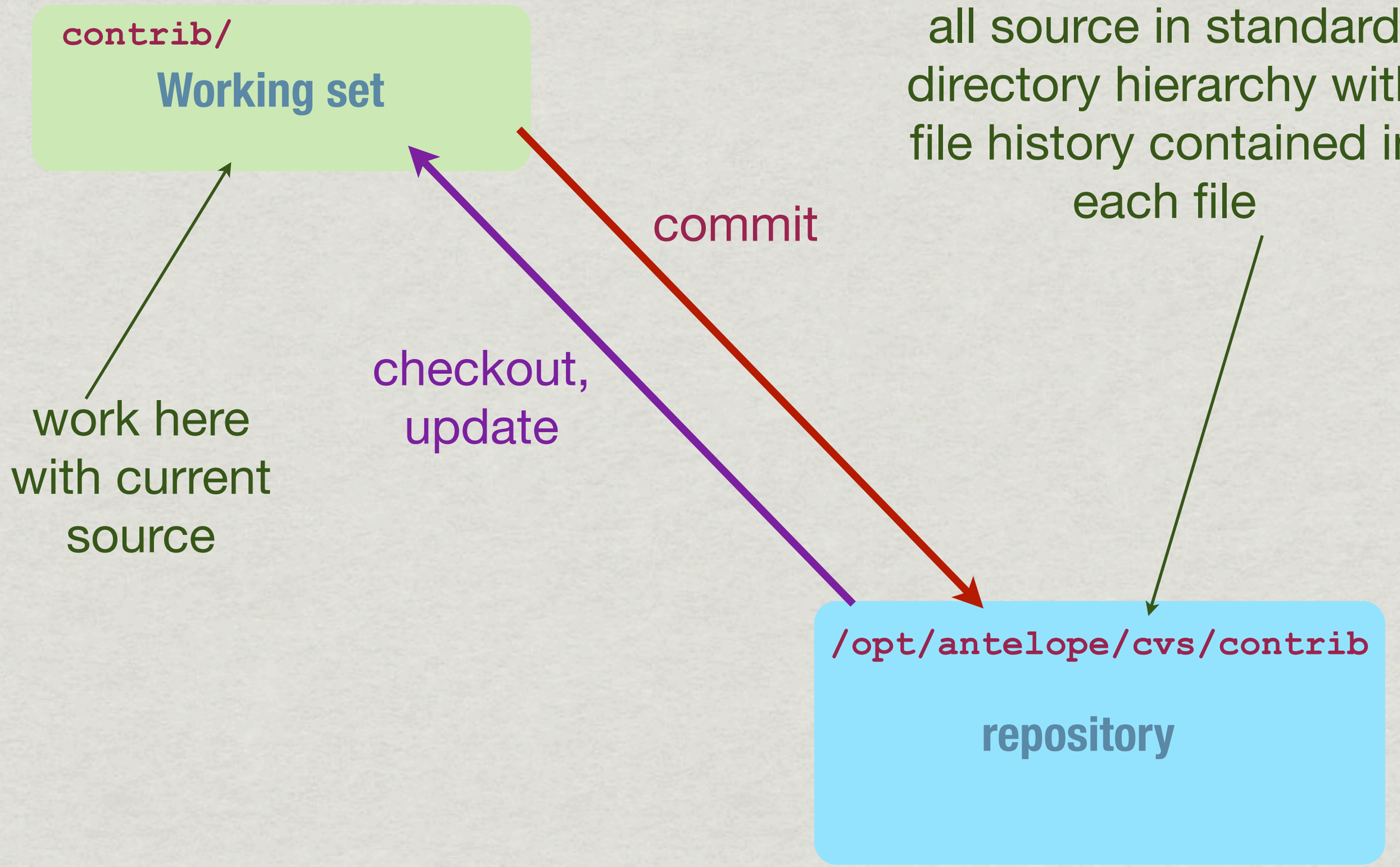
- much more complex

  you have to spend time understanding model and learning commands

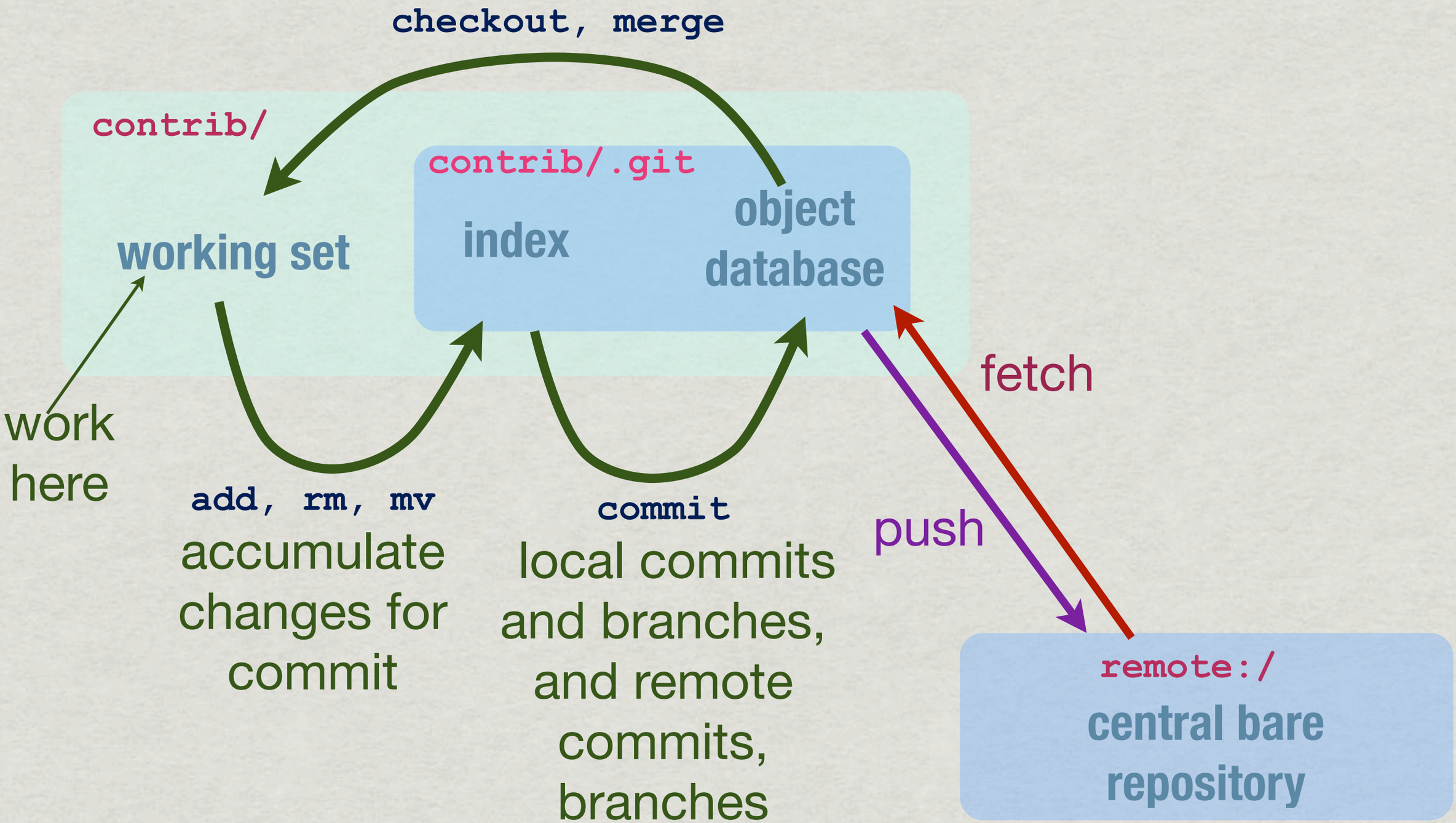
- less mature

  changes seem to be fast and furious at git central. Linus is very much in the thick of the discussion.

CVS


contrib/
Working set
commit
checkout,
update
work here
with current
source
all source in standard
directory hierarchy with
file history contained in
each file
/opt/antelope/cvs/contrib
repository

git is more complex



git

- your working directory also contains repository in toplevel .git directory
- your repository is co-equal with all other repositories; any *special* repository is only by convention

Minor point

- ***git-command*** is same as ***git command***
- but ***man git-command***

Getting started

- `git init`

- `git cvsimport -A authors -mukiv -d epicenter:/opt/antelope/cvs contrib`

  `user=129.657 sec, system=243.870 sec, elapsed=1:21:23.82, cpu use=7.6%`

  `cvs: 110 Mbytes`

  `git:  27 Mbytes`

- `(maybe) git repack -a -d -f --depth=250 --window=250`

- `git config core.bare true  # name “contrib.git” is equivalent`

- `git config core.sharedRepository group`

“bare” central repository: no working set

Translate login ids to name, email

```
% cat authors
chad=Chad Trabant <chad@iris.washington.edu>
danny=Danny Harvey <danny@brtt.com>
danq=Daniel Quinlan <danq@brtt.com>
eakins=Jennifer Eakins <jeakins@ucsd.edu>
foley=Steve Foley <sfoley@ucsd.edu>
glennt=Glenn Thompson <glenn@giseis.alaska.edu>
glp=Gary Pavlis <pavlis@indiana.edu>
horn=Nikolaus Horn <nikolaus.horn@zamg.ac.at>
kent=Kent Lindquist <kent@lindquistconsulting.com>
lindquis=Kent Lindquist <kent@lindquistconsulting.com>
pavlis=Gary Pavlis <pavlis@indiana.edu>
rnewman=Robert Newman <rnewman@ucsd.edu>
tmulder=Taimi Mulder <tmulder@ucsd.edu>
tshansen=Todd Hansen <tshansen@gmail.com>
vernon=Frank Vernon <vernon@brtt.com>
flvernon=Frank Vernon <vernon@brtt.com>
vonseg=David Vonseggern <vonseg@seismo.unr.edu>
```

personal setup

- git author/committer name/email

```
% git config --global user.name "Daniel Quinlan"
% git config --global user.email "danq@brtt.com"
% cat ~/.gitconfig
[user]
        name = Daniel Quinlan
        email = danq@brtt.com
```

Don’t work in central repository!!

```
% git clone windom:/d/day/contrib
Initialized empty Git repository in /d/week/contrib/.git/
remote: Generating pack...
remote: Done counting 28757 objects.
remote: Deltifying 28757 objects...
remote:  100% (28757/28757) done
remote: Total 28757 (delta 18160), reused 28757 (delta 18160)
Receiving objects: 100% (28757/28757), 25.33 MiB | 6938 KiB/s,
done.
Resolving deltas: 100% (18160/18160), done.
Checking out files: 100% (4851/4851), done.

% ls contrib
bin/            data/           java/           junkyard/
lib/            test_programs/
```

Add a program to index

```
% cd contrib/bin/utility
% mkdir hello
% cd hello
% vi hello.c


% git status
# On branch master
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       ./
nothing added to commit but untracked files present (use "git add" to track)


% git add .
% git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       new file:   hello.c
```

Commit to local repository

```
% git commit -m "add new program"
Created commit ef4b130: add new program
 1 files changed, 7 insertions(+), 0 deletions(-)
 create mode 100644 bin/utility/hello/hello.c
```

- git-commit brings up editor
  - -m is a shortcut
  - -v shows diffs
- good practice to make 1st line summary, for gui

push changes to origin

```
% git push
Counting objects: 9, done.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 527 bytes, done.
Total 6 (delta 3), reused 0 (delta 0)
refs/heads/master: 2b7395de8747df6c5f716d2df755c9e0535fd039 ->
ef4b1306e6afe21a69b798d63e30d4516336fff6
To windom:/d/day/contrib
   2b7395d..ef4b130  master -> master
```

Complications

```
% vi hello.c Makefile
% make
cc -g -i -D_REENTRANT -m64 -I/opt/antelope/dev-64/include -xcode=pic32 -L/opt/antelope/dev-64/lib -L/opt/antelope/dev-64/static
-R/opt/antelope/dev-64/local/lib -R/opt/antelope/dev-64/lib -o hello hello.c
```

```
% git status
# On branch master
# Changed but not updated:
#   (use "git add <file>..." to update what will be committed)
#
#       modified:   hello.c
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       .make.state
#       .optimize.dir
#       .optimize.pag
#       Makefile
#       hello
no changes added to commit (use "git add" and/or "git commit -a")
```

fix with .gitignore

```
% cat .gitignore
.make.state
.optimize.*
hello

% git status
# On branch master
# Changed but not updated:
#   (use "git add <file>..." to update what will be committed)
#
#       modified:   hello.c
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       .gitignore
#       Makefile
no changes added to commit (use "git add" and/or "git commit -a")
```

- hierarchy of .gitignore up to top level
- $GIT_DIR/info/exclude
- man gitignore
- watch out for space at end of line

push fails

- if git push fails

```
% git push
To git@github.com:user/repo.git
! [rejected] branchname -> branchname (non-fast forward) error:
failed to push some refs to 'git@github.com:user/repo.git'
```

- Try git pull

```
% git pull
Automatic merge failed; fix conflicts and then commit the result.
```

- Try edit, git commit, then git push
- probably shouldn’t try “git push -f”

Summary

- use ***git-clone*** to start
- edit, then use ***git-add*** to put things into index
- use ***git-rm*** to remove files, ***git-mv*** to rename
- repeat as desired
- add ***.gitignore*** to ignore local products
- use ***git-commit*** to register a new waypoint locally
- use ***git-push*** to push changes to remote origin
- may need to use ***git-pull*** first to merge intermediate changes (like cvs, but not file-wise)

Other complications

- conflicts are resolved like cvs, though some better gui merge tools may exist
- can exchange updates between different repositories directly, rather than through central repository
- use branches to isolate different speculative lines of development: much simpler, easier, faster than cvs
- seen ~10 basic commands (but not options), only about 140 more to go.

Repository in more detail

```
% mkdir example
% cd example
% git init
Initialized empty Git repository in .git/
% ls .git
HEAD        config    description  hooks/    info/      objects/  refs/
```

configuration

scripts

directory/file data

status info

version references into objects

```
% cat .git/description
Unnamed repository; edit this file to name it for gitweb.
```

```
% cat .git/HEAD
ref: refs/heads/master
```

http://www.kernel.org/pub/software/scm/git/docs/repository-layout.html

Add a file to an empty repository

```
% git init
% mkdir hello
% cd hello
% vi hello.c

% git ls-files
```

```
% git add .
% git ls-files
hello.c
```

What changed in .git?

```
% cmpdirs -rc .git ../copy/.git
  present in .git, not in ../copy/.git:
    index
  present in .git/objects, not in ../copy/.git/objects:
    20/
```

```
% git ls-files -s
100644 20c482a916253f619b0123df5280b942643828da 0   hello/hello.c
```

```
% ls -l .git/objects/20/c482a916253f619b0123df5280b942643828da
-r--r--r--  1 danq  admin  102 May 30 09:29 .git/objects/20/
c482a916253f619b0123df5280b942643828da
% file .git/objects/20/c482a916253f619b0123df5280b942643828da
.git/objects/20/c482a916253f619b0123df5280b942643828da: VAX COFF executable not
stripped
```

```
% git cat-file -p 20c4
#include <stdio.h>
int
main (int argc, char **argv)
{
    printf("Hello, world!\n" ) ;
}
```

Notice!!

- *git add* is different from *cvs add*: it copies files to object database, (leaving references in the index file).

- This is called ***staging*** the files, into the ***cache*** or ***index***.

- The next step is a ***commit***.

Now commit index to local

```
% git commit -m "first commit of hello.c"
Created initial commit 406053a: first commit of hello.c
 1 files changed, 8 insertions(+), 0 deletions(-)
 create mode 100644 hello/hello.c


% cmpdirs -rc .git ../copy2/.git
different: .git/index                    ../copy2/.git/index
  present in .git, not in ../copy2/.git:
    logs/
  present in .git/objects, not in ../copy2/.git/objects:
    40/ 66/ f6/
present in .git/refs/heads, not in ../copy2/.git/refs/heads:
  master
```

more detail

```
% find .git/objects -type f
.git/objects/20/c482a916253f619b0123df5280b942643828da
.git/objects/40/6053a22104d6e0334ba2243335003fd59b55ba
.git/objects/66/719022e6659bc26e9af52cd16bb4324cec1e38
.git/objects/f6/e87e0cdf38264737e0404f2feb61fedcc10f9d

% foreach i ( 20c4 4060 6671 f6e8 )
? echo $i `git cat-file -t $i`
? end
20c4 blob
4060 commit
6671 tree
f6e8 tree

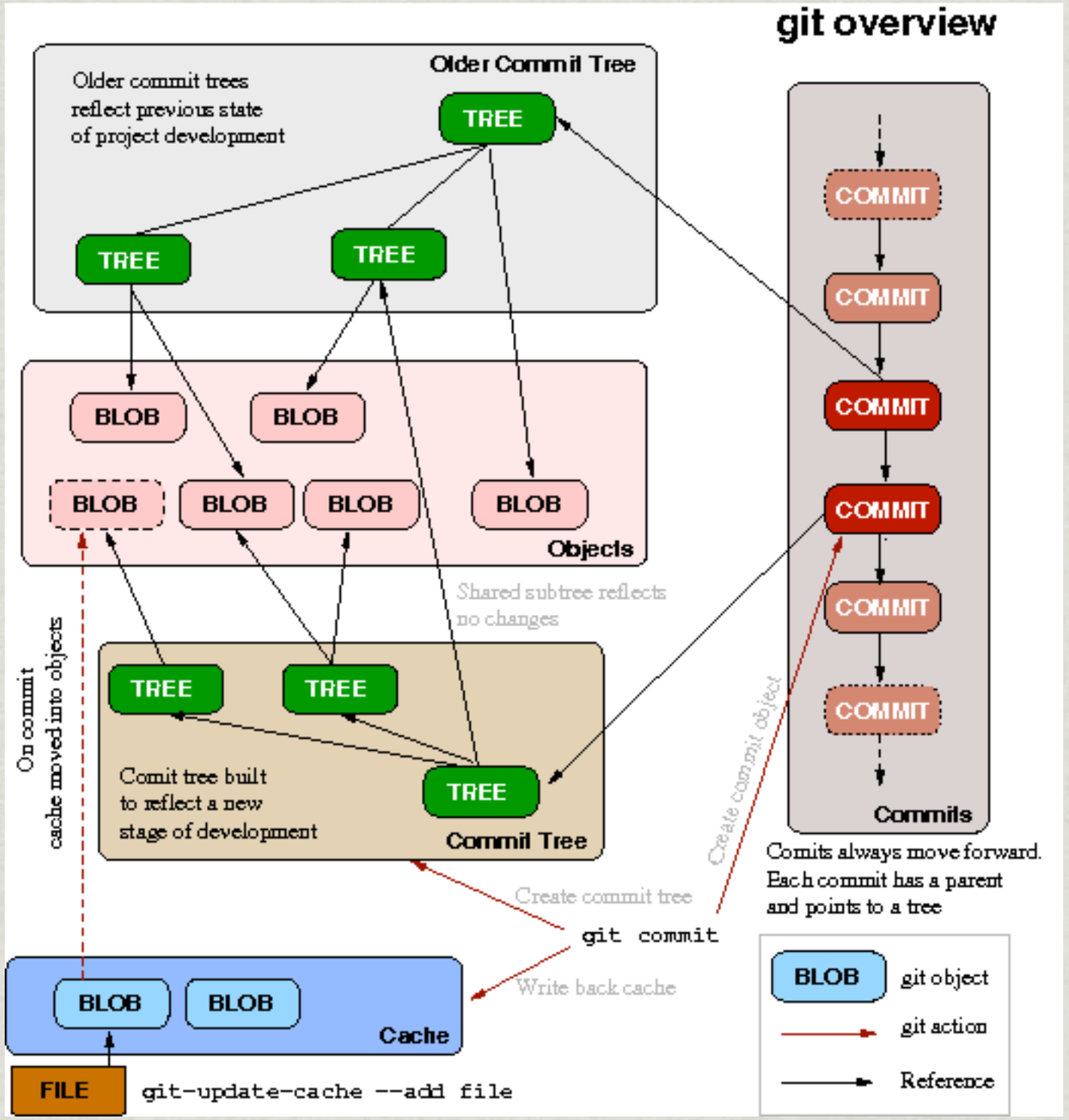
% git cat-file -p 4060
tree 66719022e6659bc26e9af52cd16bb4324cec1e38
author Daniel Quinlan <danq@windom.brtt.com> 1212162757 -0600
committer Daniel Quinlan <danq@windom.brtt.com> 1212162757 -0600

first commit of hello.c

windom% git cat-file -p 6671
040000 tree f6e87e0cdf38264737e0404f2feb61fedcc10f9dhello

windom% git cat-file -p f6e8
100644 blob 20c482a916253f619b0123df5280b942643828dahello.c
```

git operation overview



from **www.technovelty.org/ code/linux/**

moving directory the wrong way

```
% mv hello hello-world

% git status
# On branch master
# Changed but not updated:
#   (use "git add/rm <file>..." to update what will be
committed)
#
#  deleted:    hello/hello.c
#
# Untracked files:
#   (use "git add <file>..." to include in what will be
committed)
#
#  hello-world/
no changes added to commit (use "git add" and/or "git commit -
a")

% mv hello-world hello

% git status
# On branch master
nothing to commit (working directory clean)
```

moving directory the right way

```
% git mv hello hello-world

% git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#   renamed:    hello/hello.c -> hello-world/hello.c
#

% git commit -m 'renamed hello to hello-world'
Created commit 89d3668: renamed hello to hello-world
 2 files changed, 6 insertions(+), 6 deletions(-)
 create mode 100644 hello-world/hello.c
 delete mode 100644 hello/hello.c

% git cat-file -p 89d3668
tree 3927d29bdb0090c891e16099840178bc6963fe78
parent 6384d0fde4ec5ade4f639b6f5070f9aff5b5cee4
author Daniel Quinlan <danq@windom.brtt.com> 1212015158 -0600
committer Daniel Quinlan <danq@windom.brtt.com> 1212015158 -0600

renamed hello to hello-world

% git cat-file -p 3927d29bdb0090c891e16099840178bc6963fe78
040000 tree 2e54ff61570198d4cd3cc7267ff64859e2b62cf9    hello-world
```

Other Problems

- rcs tags: $Revision:$ $Date:$
  - `% git show-ref -s HEAD`

    `fe22414b4f316807a8d88792ede6309f625d46b8`
  - `% git diff --shortstat HEAD`

    `7 files changed, 15 insertions(+), 10 deletions(-)`
  - `% git log -n 1 --pretty=format:%ci deposit.1`

    `2006-11-06 16:32:13 +0000`
- no empty directories

Other Problems

- tkdiff

```
chomp(my $prefix = `git-rev-parse --show-prefix`) ;
$opt_r = "HEAD" if ! defined $opt_r ;
system ("git show '$opt_r:$prefix$filename' > $tmpfile" ) ;
system ( "tkdiff $tmpfile $filename" ) ;
```

- Recover original file

```
% git checkout file
```

- “push” mail

```
### copy new version to .git/hooks/post-receive
% chmod +x .git/hooks/post-receive
```

- add to previous commit (before push!!!)

```
### fix problems (edit, git add, whatever)
% git commit --amend
```

Further exploration

- git-gui, gitk, qgit, gct, git mergetool
- branches: explicit merges
- reset, rebase, fast-forward
- stash
- bisect
- blame (annotate)
- cherry-pick
- ```
  git log --since="June 5, 2005"
    --grep="find this"
    --author="bob@domain.com"
    --pretty=oneline some-branch
    -- crypto/*.c
  ```
  ```

Getting git

- macports: port install git-core
- http://code.google.com/p/git-osx-installer/
- sun:

  ftp://ftp.sunfreeware.com/pub/freeware/sparc/10/git-1.5.4.2-sol10-sparc-local.gz

  (get man pages separately -- I copied from mac installation)

- linux: probably installed already.

References

- Linus at Google
  http://youtube.com/watch?v=4XpnKHJAok8
- Git User Manual
  http://www.loria.fr/~molli/pmwiki/uploads/Main/gitmanual.pdf
- A tutorial introduction to git
  http://www.kernel.org/pub/software/scm/git/docs/tutorial.html
- Git Magic
  http://www-cs-students.stanford.edu/~blynn/gitmagic/ch01.html
- git for CVS users
  http://www.kernel.org/pub/software/scm/git/docs/cvs-migration.html
- Git FAQ
  http://git.or.cz/gitwiki/GitFaq
- intro slides
  http://www.jukie.net/~bart/slides/intro-to-git/intro-to-git.pdf
- podcast ($9), pdf
  http://peepcode.com/products/git-internals-pdf
  http://peepcode.com/products/git

WITH THE LAST GIT RELEASE, `GIT-REBASE` GAINED A NEW OPTION: `--INTERACTIVE`.

IF YOU ALREADY HAD THE FEELING THAT IN A PATCH SERIES OF YOURS YOU SHOULD HAVE ORDERED PATCHES DIFFERENTLY, OR MERGED SOME, THEN THIS COMMAND IS WHAT YOU DREAMED OF. HERE IS HOW IT WORKS…

LET'S PRETEND YOU WANT TO REWORK YOUR LAST 10 PATCHES, YOU'LL RUN:

```
$ GIT REBASE -I HEAD~10
```

IT WILL LAUNCH YOUR `$EDITOR` AND YOU'LL SEE SOMETHING LIKE:

```
# REBASING 16D3800..14F3D11 ONTO 16D3800
#
# COMMANDS:
#  PICK = USE COMMIT
#  EDIT = USE COMMIT, BUT STOP FOR AMENDING
#  SQUASH = USE COMMIT, BUT MELD INTO PREVIOUS COMMIT
#
# IF YOU REMOVE A LINE HERE THAT COMMIT WILL BE LOST.
#
PICK 6270640 SIMPLIFY WRITE_TREE USING STRBUF'S.
PICK 27C528A FURTHER STRBUF RE-ENGINEERING.
PICK FD82C9A ERADICATE YET-ANOTHER-BUFFER IMPLEMENTATION IN BUITIN-RERERE.C
PICK EEE488F MORE STRBUF USES IN CACHE-TREE.C.
PICK 16878B5 ADD STRBUF_RTRIM AND STRBUF_INSERT.
PICK E9081AF CHANGE SEMANTICS OF INTERPOLATE TO WORK LIKE SNPRINTF.
PICK 99C3EF5 REWORK PRETTY_PRINT_COMMIT TO USE STRBUFS INSTEAD OF CUSTOM BUFFERS.
PICK 203DB5D USE STRBUF_READ IN BUILTIN-FETCH-TOOL.C.
PICK A20D939 USE STRBUFS TO IN READ_MESSAGE (IMAP-SEND.C), CUSTOM BUFFER--.
PICK 14F3D11 REPLACE ALL READ_FD USE WITH STRBUF_READ, AND GET RID OF IT.
~
~
~
~
~
~[1]
```

THEN YOU CAN REWRITE "`PICK`" INTO "`EDIT`" IF YOU WANT TO CHANGE SOMETHING IN A COMMIT, OR "`SQUASH`" IF YOU WANT TO MERGE IT WITH THE ONE FROM THE LINE BEFORE.

WHAT THE SMALL HELP DOESN'T SAY IS THAT YOU CAN ACTUALLY **REORDER** YOUR COMMITS, AND IT WILL DO WHAT YOU EXPECT IT TO DO. I USED IT 10 MINUTES AGO, BECAUSE I HAVE THIS STRING BUFFER MODULE I EXTEND ON A REGULAR BASIS, I SQUASHED EVERY API EXTENSION OF THAT MODULE IN ONE COMMIT USING THAT.

EACH TIME ONE CHANGE NEEDS YOU TO EDIT ANYTHING BECAUSE EITHER YOU ASKED FOR IT, OR THAT ONE OF THE CHANGE YOU ASKED FOR GENERATED A CONFLICT, THEN AS USUAL THE REBASE WILL STOP. YOU WILL BE PROMPTED TO MAKE THE CHANGE, OR FIX THE CONFLICT, OR MERGE COMMENTS (IN CASE OF A SQUASH), AND WHEN ALL IS IN ORDER, YOU JUST NEED TO:

```
$ GIT REBASE --CONTINUE
```

THIS IS JUST **AWSOMELY SIMPLE AND INTUITIVE**

Other git users

- Cairo (2D graphics library with support for multiple output devices)
- Debian buildd-tools (buildd, sbuild, schroot)
- Erlware (Erlang/OTP development tools)
- Fedora
- GNU Autoconf
- GNU Automake
- GNU core utils
- Mesa3D (graphics library, software OpenGL implementation)
- One Laptop Per Child (OLPC)
- Ruby on Rails (The popular MVC framework for Ruby)
- Samba (provides file and print services to all manner of SMB/CIFS clients)
- Slash - the discussion system that powers Slashdot
- Wine (implementation of the Windows API on top of X and Unix)
- X.Org (X server, X libraries, classic X applictions, drivers)