Seminar: Search and Optimization

4. An Introduction to Revision Control with Mercurial

Gabi Röger

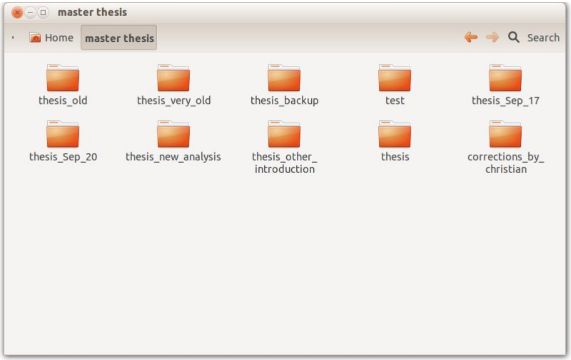
Universität Basel

October 10, 2013

Revision Control

What’s Revision Control?

Manage multiple versions of files



Why should we use it?

- Track the history: Who made when what changes?
- Manage easily multiple versions of your work (e.g. when refactoring)
- Collaboration with others: Merging your work
- Backup in case of mistakes

Revision Control Systems

- CVS
  - old-style centralized revision control
  - cons: outdated dinosaur (don't use it)
- Subversion (svn)
  - old-style centralized revision control
  - pros: fine-grained access rights
  - cons: painful merging of changes; needs access to central server
- Git and Mercurial (Hg)
  - distributed revision control
  - pros: fast, flexible, intelligent merging, allows different models of collaboration
  - cons: not meant for fine-grained access-rights or sub-repositories (albeit possible)

Installing Mercurial

- Linux (Ubuntu):
  - Necessary: Mercurial
    `sudo apt-get install mercurial`
  - Optional: GUI
    TortoiseHg: `sudo apt-get install tortoisehg`
  - Optional: Graphical merge tool
    Meld: `sudo apt-get install meld` or
    Kdiff3: `sudo apt-get install kdiff3`
- Windows: TortoiseHg `http://tortoisehg.bitbucket.org/`
- Mac: for example MacHg
  `http://jasonfharris.com/machg/`

Test installation with `hg --version`

First steps

Creating a repository

- `hg init [DEST]`
  initialize new repository
  (create subdirectory `.hg` in [DEST])

Example

```
$ hg init
```

make current directory a repository

```
$ hg init project
```

start a repository in directory `project`
(create it if it does not exist)

Before we begin

Who made what changes?

▷ Mercurial needs to know who you are

Edit configuration file

- `pathtorepository/.hg/hgrc` for local settings
- `~/.hgrc` for global settings

Example (pathtorepository/.hg/hgrc)

```
[ui]
username = Gabi Roeger <gabriele.roeger@unibas.ch>
```

Adding files and commiting changes

- `hg add [OPTION]... [FILE]...`
  Puts file under revision control
- `hg commit [OPTION]... [FILE]...`
  commit changes of the specified files or all outstanding changes
  Option `-m`: specify log message (otherwise opens a text editor)

Example

```
$ echo "realy elaborated text" > important_text
$ hg add important_text
$ hg commit -m "added important text"
$ sed -i -e 's/realy/really/' important_text
$ hg commit -m "fixed typo"
```

Deleting files

- `hg remove [OPTION]... [FILE]...`
  `hg rm [OPTION]... [FILE]...`
  deletes from file system and repository control
- `hg forget [FILE]...`
  removes files from repository control (on the next commit)

Example

```
$ touch file1 file2
$ hg add file1 file2
$ hg commit -m "added files"
$ hg rm file1
$ hg forget file2
$ hg commit -m "removed some files"
```

Status of the working directory

```
hg status [OPTION]... [FILE]...
hg st [OPTION]... [FILE]...
```

show changed files in the working directory

Important flags:

A	added
M	modified
R	removed
!	missing
?	not tracked

Ignoring files

Patterns in file `pathtorepository/.hgignore` describe files that should not be considered by hg commands (eg., `hg st`):

- Syntax regexp: regular expressions, Python/Perl syntax (default)
- Syntax glob: shell-style glob

Example (pathtorepository/.hgignore)

```
syntax: regexp
program
\.o$
```

Example (pathtorepository/.hgignore)

```
syntax: glob
program
*.o
```

Reverting uncommited changes

`hg revert [OPTION]... [FILE]`
restore files to their checkout state
Option `--all`: revert all changes

▷ Modified files are saved with a .orig suffix before reverting.

Example

```
$ hg st
M foo.txt
$ hg revert foo.txt
$ hg st
? foo.txt.orig
```

History

- `hg log [OPTION]... [FILE]`
  show revision history of entire repository or files

Example

```
$ hg log
changeset:   3:a4a8975c32a8
tag:         tip
user:        Gabi Roeger <gabriele.roeger@unibas.ch>
date:        Tue Sep 25 16:28:14 2012 +0200
files:       file1 file2
description:
removed some files


changeset:   2:cc210a3f1a3e
...
```

Inspecting changes

```
hg diff ([-c REV] | [-r REV1 [-r REV2]]) [FILE]...
```

show diff for repository (or files)

Option `-c`: change made in revision

Option `-r`: difference between revision and working copy/other rev.

- two revision arguments: compares those revisions
- one revision argument: compares the revision to the working directory
- no revision argument: compares the parent revision to the working directory

Moving through time

- `hg update [[-r] REV]`
  `hg up [[-r] REV]`
  Switch working directory to revision (or newest revision)
- `hg parents [-r REV] [FILE]`
  Show parent revisions of working directory or revision

Getting help

Most commands have much more options than shown:

- `hg help COMMAND`
  show documentation for command

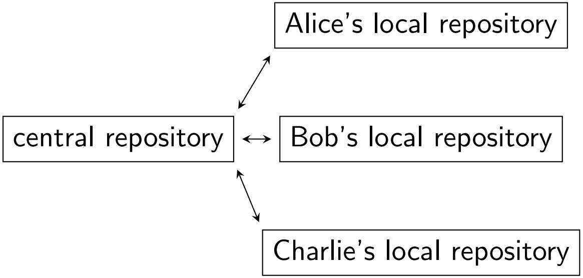
Example

```
$ hg help update
hg update [-c] [-C] [-d DATE] [[-r] REV]

aliases: up, checkout, co

update working directory (or switch revisions)

Update the repository's working directory to the
specified changeset. If no changeset is specified,
update to the tip of the current named branch.
(...)
```

Distributed development

Repository architecture

- Many possible alternatives
- Good option for small non-hierarchical group of developers:
  - ▷ One central repository:



Cloning

`hg clone SOURCE [DEST]`
create a copy of an existing repository

Example

```
$ hg clone ../project project-alice
$ hg clone http://hg.fast-downward.org fast-downward
```

Checking for incoming/outgoing changes

- `hg incoming [SOURCE]`
  `hg in [SOURCE]`
  show new changesets found in source
- `hg outgoing [DEST]`
  `hg out [DEST]`
  show changesets not found in the destination

Source or destination not specified
▷ default from `.hg/hgrc`

Transfering changes

- `hg pull [-u] [SOURCE]`
  pull changes from the specified source
  default: does not update the working directory
  option -u: automatically update after pulling
- `hg push [-f] [DEST]`
  push changes to the specified destination

Push aborts with error new remote head?

- Pull first and merge divergent changes (next slide)
- If you are sure that you actually want it and know why:
  Use `hg push -f` to force new head to destination repository

Resolving divergent history

If you have several heads in the repository (usually after a `pull`)

- `hg heads`
  show current repository heads
- `hg merge [REV]`
  update current working directory with all changes made in the requested revision since the last common predecessor.
  (If no revision is specified, the working directory's parent is a head revision, and the current branch contains exactly one other head, the other head is merged with by default.)
  - ▷ Automated merge if possible
  - ▷ Otherwise opens merge tool for manual merge
  - ▷ Don't forget to commit after merging

Finding the right contact person

`hg annotate [-u] [-n] [-r REV] FILE`
show changeset information by line for each file
Option `-u`: show user Option `-n`: show revision number

Example

```
$ hg annotate -un program.cpp
gabriele 1: #include <iostream>
gabriele 1:
gabriele 1: int main(int, char**)
     bob 5:     std::cout << "Bob and Alice say:";
     bob 8:     std::cout << "Hello world" << std::endl;
   alice 6:     std::cout << "The world says: Hello! ";
     bob 8:     std::cout << "Alice and Bob go home.";
gabriele 1:
```

Wrap-up

Characterization of commands

- Communicating with other repository
  - Only reporting: `incoming`, `outgoing`
  - Changing: `pull`, `push`
- Local commands
  - Only reporting: `annotate`, `diff`, `heads`, `help`, `id`, `log`, `status`
  - Changing: `add`, `commit`, `forget`, `init`, `merge`, `remove`, `revert`, `update`

Getting further

- Interesting next topics:
  - branching
  - tagging revisions
  - backout old changesets
- Tutorials and documentation:
  - `http://hginit.com`
    basic example-driven tutorial
  - `http://hgbook.red-bean.com`
    covering almost everything; also available as (printed) book
- Sharing a repository
  - Quick-and-dirty: `hg serve`
  - Long-term: Use hosting service (`https://bitbucket.org/`)
    or set up your own web-server accordingly