CS246 Mini-lab
Spring 2016
Revision Wars (in class only)

**Purpose**:

Revision control can be daunting. This lab is meant to introduce you to a safe place for experimentation, **the revisionWars repository**.

**Submit**: Each person **must** have committed and pushed at least 1 revision to /rd/cs246s2016/shared/revisionWars with a comment that has your **Full Name**, **username**, and a brief description of the changes made.

**Setup**:

- Groups of 4, two paired programming teams.
- Select a group name. Clone revisionWars from /rd/cs246s2016/shared/ Make a file <GroupName.cpp> in the revisionWars/ directory.
- One pair should make the file in their own revisionWars clone.
- Then after they commit and push, the other pair should make changes.
- **Copy the below main function to your file:**

```
int main(char** argv, int argc) {

    // Pair 1 partner A code

    // Pair 1 partner B code

    // Pair 2 partner A code

    // Pair 2 partner B code


}
```

Tasks:

Each pair should do a standard pair programming task where the first person

1. Fills in their spot.
2. commits the code
3. synchronizes the file
    a. pull,
    b. merge/update,
    c. commit,
    d. push

While the other person actively critiques.

(continues on next page…)

Once you've made the file and put some text into it, you should commit it, and push it to the server. Note: There may be a failure when you try to push to the server if there is a revision on the server that you haven't pulled yet. If that is the case, you will need to pull, and merge any files that have multiple versions. TortoiseHg Workbench or Synchronization can help you through this process. Some documentation is excerpted in the following pages. The quick start guide,
http://tortoisehg.readthedocs.org/en/latest/quick.html,
and TortoiseHg in daily use:
http://tortoisehg.readthedocs.org/en/latest/daily.html,
have the most useful information.

source:
http://tortoisehg.readthedocs.org/en/latest/quick.html#working-with-your-repository

# 4.9. Working with your repository

Suppose you've introduced some changes. It is easy to discover what pending changes there are in the repository.

Workbench: go to the Commit task tab and inspect the filelist at the left

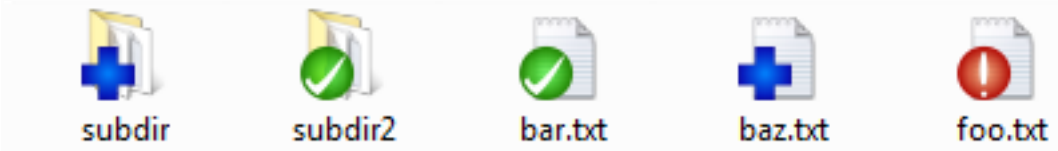
Any files marked with 'A' (added, green), with '?' (unversioned but not ignored, fuchsia), with 'M' (modified, blue), or with '!' (removed, red) indicate pending changes that should be committed.

The Commit task tab in the Workbench gives you a way to see differences within the files, or you can use your visual difference tool (kdiff3). Mercurial allows you to commit many changes before you decide to synchronize (share changes) with the group repository.

Explorer: inspect the icons on the folders and files in your repository

Folders or files in Explorer marked with one of the icons below are another way of indicating pending changes. You can traverse the directories to find specific

changes and commit them from Explorer. Though it might be quicker to do that from the Commit task tab in the Workbench.




Overlay Icons on Vista
Command line: type **thg commit**

When you're ready to publish your changes, you

1. Commit your changes to your local repository (see above).
2. Pull changes from the group repository into your repository using *TortoiseHg ‣ Workbench* or **thg log**, select the Sync task tab, choose the path to the group repository in the syncbar and then click the *Pull* button.
3. If some changesets were pulled, merge those changes with your local changes and then commit the merge into your local repository. From the revision history viewer (*TortoiseHg ‣ Workbench* or **thg log**) open the context menu over the changeset which you want to merge and select *Merge with local...*. Finally, in the merge dialog, press *Merge* and then *Commit*.
4. Ensure your merged work still builds and passes your extensive test suite.
5. Push your changes to the group repository, *TortoiseHg ‣ Workbench* or **thg log**, select the path to group repository and then click the *Push* button.

Which may sound complicated, but is easier than it sounds.

**Note**

Merges can be safely restarted if necessary.

Mercurial makes collaboration easy, fast, and productive. Learn more at Mercurial's wiki.

Source: http://tortoisehg.readthedocs.org/en/latest/workbench.html

## 5.4.4. Sync Toolbar

Sync toolbar

Synchronize your repository with other repositories.

***Incoming***

Download incoming changesets from the remote repository, store then in a temporary bundle file, then enter bundle preview mode with the incoming changes applied. Incoming changesets will be shown as normal, while others will be shown grayed in the revision graph. The buttons*Accept* and *Reject* are then shown at the top of the revision graph.

***Pull***

Pull incoming changesets from the remote repository, then apply after-pull effect (update, fetch, or rebase).

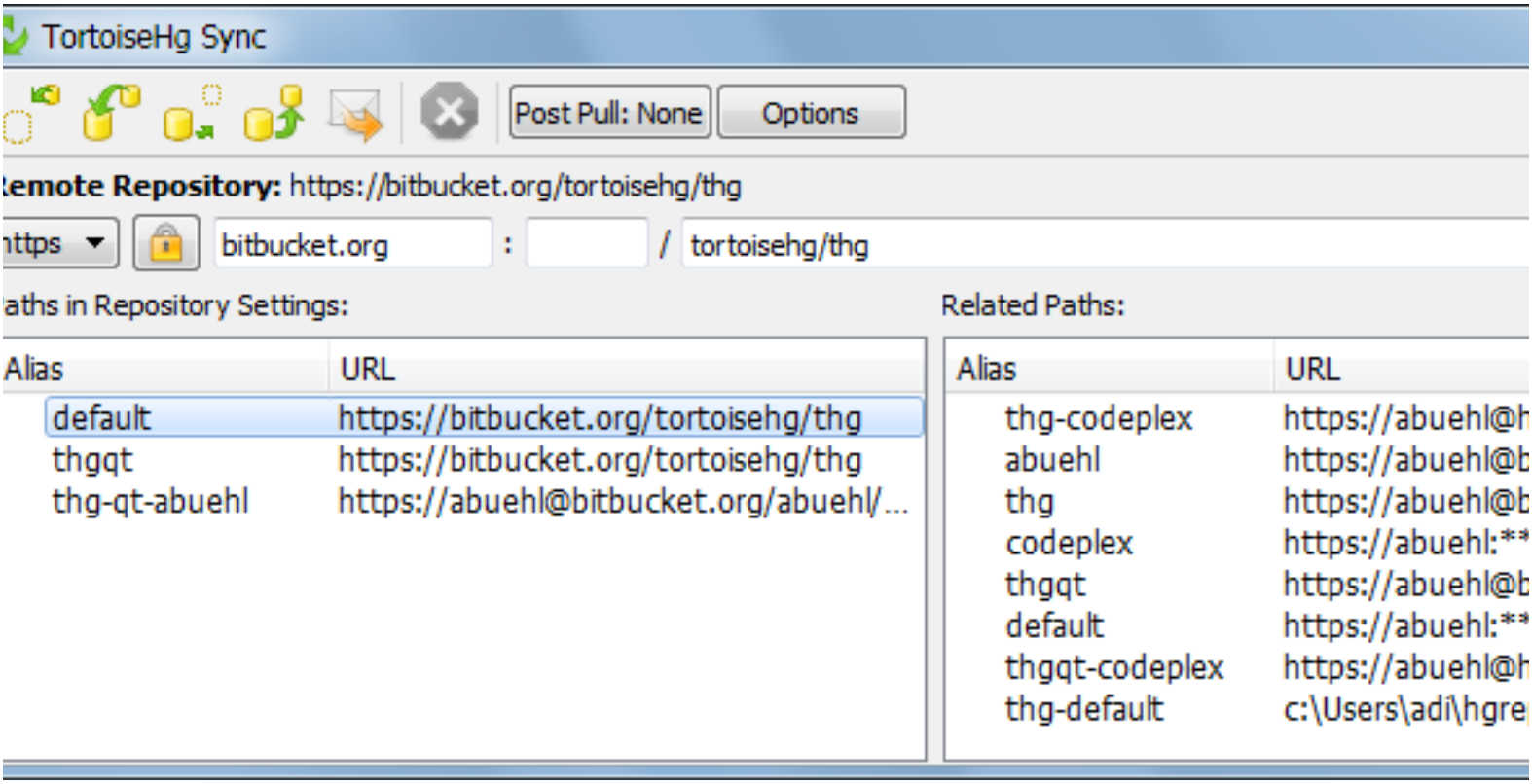
***Outgoing***

Determine outgoing changesets that would be pushed to the remote repository. Outgoing changesets will be shown as normal, while others will be shown grayed in the revision graph.

***Push***

Push outgoing changesets to the remote repository.

Source: http://tortoisehg.readthedocs.org/en/latest/sync.html




# 5.9. Synchronize

Synchronize dialog

The synchronize tool is used to transmit changesets between repositories or to email recipients.

***Incoming***

show changesets that would be pulled from target repository, the changes in the target repository that are not in local repository

***Pull***

pull incoming changesets from target repository

***Outgoing***

show changesets that would be pushed to target repository, the changes in the local repository that are not in target repository

***Push***

push outgoing changesets to target repository, make the local *tip* the new *tip* in the target repository

***Email***

send outgoing changesets (to target repository) as email

***Stop***

stop current operation

The *Post Pull* dialog contains radio buttons for selecting the operation which is performed after a pull. If you open the configuration tool, you can select a default behavior for your user account and override that selection on a per-repository basis.

***None***

No operations are performed after a pull. You will be allowed to view the pulled changesets in the log viewer, and you will have the option to update to the new tip if applicable.

***Update***

Automatically update to the current branch tip if, and only if, new revisions were pulled into the local repository. This could trigger a merge if the pulled changes conflict with local uncommitted changes.

***Fetch***

Equivalent to hg fetch. See the fetch extension documentation for its behavior. This feature is only available if the fetch extension has been enabled by the user.

***Rebase***

Equivalent to pull –rebase. See the rebase extension documentation for its behavior. This feature is only available if the rebase extension has been enabled by the user.

***Automatically resolve merge conflicts where possible***

If update or rebase are selected, a pull operation may result in a merge. If checked, Mercurial will try to resolve trivial merge conflicts without user interaction. If not checked, all merges will be interactive.

The *Options* dialog provides checkboxes for selecting infrequently used command options.

***Allow push of a new branch***
: allow a new named branch to be pushed

***Force pull or push***
: override warnings about multiple heads or unrelated repositories

***Recurse into subdirectories***
: incoming or outgoing commands can recurse into subdirectories and provide a full report

***Temporarily disable configured proxy***
: only sensitive when a web proxy is configured for the given repository. While checked it will disable that proxy.

***Remote Command***
: provides a –remotecmd argument

When the sync tool is opened within the Workbench, the toolbar has a *Target* checkbox. While checked, the target dropdown box is sensitive and the selected target revision, bookmark, or branch will be added to every synchronization command. When the sync tool is opened outside of the Workbench, the target checkbox and dropdown box is hidden. Clicking on a revision in the graph will update the values in the dropdown box.
Holding `Alt` while clicking on a revision will select the revision without switching away from the sync tool tab.

Below the toolbar is the currently selected URL. All synchronization commands will use this URL. The general effect of the toolbar is that it can be read as a Mercurial command line. The tool buttons select the command, the *Post Pull* and *Options* dialog specify options, the target dropdown box can specify revisions, and finally the URL completes the command.