Useful Tips About Some Tools

Yu Zhang (Clara)
clarazhang@gmail.com

June 6, 2011

1 GIT

GIT Reference: http://gitref.org/
git Manual Page
git-config Manual Page

- Start a new repository for an existing code base:
  `cd codebase`
  use `git init` to prepare codebase/.git directory
  use `git add ...` to add specified files to the index
  use `git commit -am '...'` to commit local repository
  use `git remote add origin git@korz...:`

- Clone the source repository, like this:
  `git clone git@korz.cs.yale.edu:pios`.

- Switch the specified branch.
  The default cloned branch is the `master` branch of the repository. You can use `git branch -a` to list all branches of the repository, where the remote branches are usually named with the prefix "origin". If you want to get the copy of another remote branch, you can type:
  `git checkout -b` *`newLocalBranch remoteBranch`*
  You can switch a branch by typing `git checkout <branch>`.

- You can create your new branch by typing:
  `git branch` *`newbranch`*

- You can update the remote-tracking branches for the repository you cloned from, then merge one of them into your current branch:
  `git pull`

- You can check what you have done on the branch:
  `git status -s`

- You can add file content by typing:
  `git add doc/*.txt` // add all txt files from subdirectories of doc/ directory
  `git add git-*.sh`

- You can check all the modified files you have and their status, and update in interactive mode by typing:
  `git add -i`
  See Interactive Adding

- You can config your name and email address by typing:
  `git config --global user.name "Your name"`
  `git config --global user.email you@example.com`

- If you finish some changes, and want to checkpoint your progress, you can *commit* your changes by running:
  `git commit -am 'info about this revision'`

- If the identity used for your commit is wrong, you can fix it with:
  `git commit --amend --author='Your name <you@example.com>'`

- You can check the commit logs by typing `git log`, see git diff ref 1 and 2
  `git log --oneline` used to see a more compact version
  `git log --oneline --graph` used to see when the history was branched and merged
  `git log --oneline --graph` used to see when the history was branched and merged
  `git log --oneline dcm ^master` used to see the commits that are in the 'dcm' branch that are not in the 'master' branch
  `git log --author=XXX --oneline -5` used to filter the commit history to only the ones done by a specific author (searching XXX in case sensitive), and to limit the results to the last 5 commits

- You can tag a specific commit with `git tag -a v1.0` so you can use it to compare to other commits in the future, where `-a` means *make an annotated tag*
  `git log --oneline --decorate --graph` used to see the commit history with tag info
  `git tag -a v1.0 3414308` used to tag the released commit `3414308`

- You can keep track of your changes by using the `git diff` command. Running `git diff` will display the changes to your code since your last commit, and `git diff remotebranch` will display the changes relative to the initial remote revision.
  You can first
  `git diff --no-prefix > patchfile`
  then apply the patch like:
  `patch -p0 < patchfile`

You can also reverse a patch like:
`patch -p0 -R < patchfile`

Other git diff commands include:
`git diff v1.0` used to see what has changed in our project since the v0.9 release
`git diff branchA branchB --stat` used to compare two divergent branches
`git diff branchA...branchB --stat`, where `...` means Git will automatically figure out what the common commit (otherwise known as the "merge base") of the two commit is and do the diff off of that

- You can push your local current branch to the remote ref matching the specified *branch* in the `origin` repository, just like:
  `git push origin branch`
- If you need to pull changes from `master` into your branch, then you can do so using `git pull` like so:
  `git pull origin`
  `git merge master`
  That command tells Git to pull all the changes from the `origin` repository (Gits name for the canonical remote repository) including all branches. Then, you merge the `master` branch into your branch.
- You can extract all commits that lead to *origin* since the inception of the project:
  `git format-patch -M -B origin`.
- You can undo your commits permanently, like this:
  `git reset --hard HEAD~2`
  where the last two commits (HEAD, HEAD^) were bad and you do not want to ever see them again.
- If you want to revert a remote branch to a specified commit, you can first switch your local branch to the branch which corresponds to the remote branch using
  `git checkout branch`,
  then decide which commit you want to revert using
  `git log`,
  after that you can use the last item, *i.e.,*
  `git reset --hard ...`
  to reset your local branch, finally you can do
  `git push origin branch`
  to push your local branch to the remote ref branch.

1.1 Important Issues

- *Symbolic links* If option `core.symlinks` of `git config` is false, symbolic links are checked out as small plain files that contain the link text. git-update-index(1) and git-add(1) will not change the recorded type to regular file. Useful on file systems like FAT that do not support symbolic links. The default is true, except git-clone(1) or git-init(1) will probe and set core.symlinks false if appropriate when the repository is created.

  Note: PIOS repository has some symbolic link files, such as `inc/types.h`. We need to modify the `.git/config` file and set the `symlinks` true, and check out the files again. Otherwise, we cannot build pios successfully when setting `LAB=9 SOL=9`.