Assignment 4. Change management

Useful pointers

- Michael Johnson, Diff, Patch, and Friends, Linux Journal 28 (1996-08)
- Larry Ayers, Patch for Beginners, Linux Gazette 32 (1998-09)
- Linus Torvalds, Jun Hamano *et al*., Git - the fast version control system
- The official Git tutorial (2009) and Everyday Git with 20 commands or so (2009)
- Ben Collins-Sussman, Brian W. Fitzpatrick & C. Michael Pilato, Version control with Subversion (2008)
- David MacKenzie, Paul Eggert, and Richard Stallman, Comparing and merging files, version 2.8.1 (2002-04-05)

Laboratory: Backing out a change

As usual, keep a log in the file `lab4.txt` of what you do in the lab so that you can reproduce the results later. This should not merely be a transcript of what you typed: it should be more like a true lab notebook, in which you briefly note down what you did and what happened.

You're helping to develop an operating system and command set that as part of its acceptance test is supposed to pass a large test suite maintained by a large government agency. The test suite is pretty much carved in stone: it was written five or ten years ago and contains some bugs, but it takes months (maybe years) to get it fixed, because of all the bureaucracy involved in getting fixes approved for a benchmark used in competitive bidding processes. Your company's product is failing several of the tests, and you've been assigned to fix one of the failures.

You track your assigned failure down to the following test:

```
mkdir test
cd test
touch ./-ooops a b c
rm *
```

The test suite expects your implementation of `rm` to output the following:

```
rm: invalid option -- 'o'
Try `rm --help' for more information.
```

Instead, your `rm`, which is based on Coreutils 8.0, outputs an extra line, generating:

```
rm: invalid option -- 'o'
Try `rm ./-ooops' to remove the file `-ooops'.
Try `rm --help' for more information.
```

The extra (middle) line is supposed to be helpful, but the test suite doesn't expect it and it rejects your implementation of `rm`. This is a bug in the test suite (it should not be so picky), but fixing the bug will take a long time and in the meantime your product will get a lower score.

You search the Internet for ideas, and find a `coreutils` patch posted 2005-08-29 that explains why

Coreutils `rm` outputs the extra line. You decide to work around the bug by backing out this patch: that way, the modified version of `rm` will output the old message.

Back out the 2005-08-29 patch, as follows:

1. Download `coreutils` 8.0.
2. Use the `patch` command to back out the patch mentioned above.
3. Record any problems you had in applying the patch.
4. Fix any problems with the patch by hand, by using your favorite text editor.
5. Build the resulting modified version of `coreutils`, and verify that it does the right thing with the scenario illustrated above.
6. From within the `coreutils-8.0` directory, generate a tentative version of your patch by using `diff` with the `-u` option to compare the original version of `rm.c` with your modified version. Put this tentative patch into the file `lab4.diff`.

Homework: Verifying and publishing the backed-out change

You're happy with the backed-out patch that you prepared in your assignment, but now you'd like to publish this patch, in a form similar to that presented in the original patch, so that others can use it.

1. Maintain a file `hw4.txt` that logs the actions you do in solving the homework. This is like your lab notebook `lab4.txt`, except it's for the homework instead of the lab.
2. Import `coreutils` 8.0 into a new Git or SVN repository of your own.
3. Install your change into a branch of the repository, by applying `patch` to your patch in `lab4.diff`. This should affect at least the two files affected by the original patch. This patch should work without having to fix things by hand afterwards.
4. Verify that the changed version works, as above.
5. Use Git's or SVN's own version-comparison commands to generate the backed-out patch into a file `hw4.patch`.
6. Use a text editor to make the ChangeLog entries have the same format as the original patch.
7. From within the `coreutils-8.0` directory, use `diff` to verify that the command `patch -p0 <hw4.patch`, when applied to a vanilla `coreutils` 8.0 distribution, generates your modified version (not counting the ChangeLog changes).

Submit

Submit the following files.

- The files `lab4.txt` and `lab4.diff` as described in the lab.
- The files `hw4.txt` and `hw4.patch` as described in the homework.

All files should be ASCII text files, with no carriage returns, and with no more than 100 columns per line. The shell command:

```
expand lab4.txt lab4.diff hw4.patch | awk '/\r/ || 100 < length'
```

should output nothing.

$Id: assign4.html,v 1.12 2010/04/12 18:22:21 eggert Exp $