Source code management

COMP8440: FOSSD
Lecture 3

Early approaches

- Simple tools
    - diff, patch and tar
    - Patches sent by email
    - Each developer maintains their own tree
    - Distribution by ftp and usenet

Patches

- ## Basic tool of code exchange
  - several formats available – unidiff now the norm
  - contains short context for each change
  - main tools: diff, patch, diffstat

```
--- a/source3/rpc_server/srv_svcctl_nt.c
+++ b/source3/rpc_server/srv_svcctl_nt.c
@@ -466,9 +466,7 @@ WERROR _svcctl_EnumServicesStatusW(pipes_struct *p,
                }

                blob = ndr_push_blob(ndr);
-               if (blob.length >= r->in.offered) {
-                       memcpy(r->out.service, blob.data, r->in.offered);
-               }
+               memcpy(r->out.service, blob.data, r->in.offered);
        }
```

Sending patches

- Common rules
    - use diff -up, exclude generated files
    - include diffstat output
    - include an explanation of your patch
    - [PATCH] at start of subject
    - use inline or plain text encoding for patch
    - break up your patches on logical boundaries
    - use a patch series if needed
    - check you've followed the project coding style
    - be sure you are sending to the right place
    - be patient, and follow up if need be
    - Add Signed-off-by (for some projects)

Let's look at some examples on the kernel list ...

First generation SCM systems

- RCS and SCCS
    - Manages files individually
    - Only one person editing at a time
    - No merge capability
    - Provides development history
    - Key data is who, what and when

The rise of CVS

- Concurrent Versions System
  - Built on top of RCS
  - Allowed for parallel development
  - Included merge and conflict resolution
    - based on diff/patch
- Hugely popular in the FOSS world
  - Dominated FOSS development from 1991 to 2005
  - Still very widely used, but less so each year
- Many limitations
  - Poor rename and directory support
  - Contacts centralised server for most operations
  - Poor branch merging support

Let's see a demo ...

Centralised vs Distributed

- Where is the project hosted?
  - CVS hosts in a central fashion
  - Each developer has a 'checkout'
  - Most project meta-data is only stored on the central server
- Distributed version control
  - All project history is locally available to all developers
  - Most systems aim for easy branching/merging

Subversion

- 'CVS done right'
  - Attempt to re-invent centralised source code control
  - Fixes many of the limitations in CVS
  - Adds project-wide revisions
  - Widely chosen to replace CVS in FOSS projects from 2001 onwards
  - Still very widely used
- Centralised design
  - Use of non-distributed design has been criticised
  - Distributed add-ons available (svk), but not widely used

Distributed Systems

- ## Early systems
  - Code Co-Op (windows based) in 1997
  - GNU Arch (aka TLA or Tom Lord's Arch) in 2001
- ## Bitkeeper
  - Adopted by Linux kernel in 2002
  - Unusual licensing model
  - Huge impact on speed of kernel development
- ## Newer systems
  - Lots of new systems starting in 2003
  - bazaar, darcs, mecurial, git, monotone
  - git has gained widest following

Bitkeeper and git

- ## Controversial choice
  - Linux kernel adopted bitkeeper in 2002
  - Led to acceleration of kernel development
  - Proprietary tool, with freeware client
  - License terms included a strong non-compete clause
- ## Move to git
  - Makers of bitkeeper disapproved of free client
  - Free bitkeeper withdrawn in 2005
  - Replaced by new system 'git' in June 2005

SCM Interfaces

- Command line dominates
  - Most FOSS users use command line interfaces
  - Tools aim to produce a very fast workflow
  - Most SCM tools also offer GUI or editor interfaces
- Web Interfaces
  - Many SCM tools provide web interfaces
  - Mostly used to browse development history
  - cvsweb, gitweb both very popular
  - Custom web interfaces are often built
- Interfaces with other systems
  - Tools to integrate with bug tracking systems
  - Integration with build management and build farms

Build Farms

- Integrating SCM with testing
  - Automating testing can help find bugs faster
  - Especially important for portability
- A build farm
  - Wide range of machines/OSes
  - Automatically run regressions tests on commit
  - Usually results are available publicly
  - Build/test failures may be reported by email
- Examples
  - Tinderbox
  - Samba build farm
  - Build-bot

Public SCM Hosting

- ## Canned hosting
  - Many/most FOSS projects used a canned hosting solution
  - Canned project hosting can make project maintainence much easier
  - usually less flexible than running your own
- ## DVCS public hosting
  - DVCS workflow created a demand for easier hosting
  - Many sites have sprung up for all the DVCS systems
  - See for example
    - Git: repo.oz.cz, github.com
    - Hg: bitbucket.org, freehg.org
    - bzr: launchpad.net

SCM Compatibility

- Two SCMs, one project
  - Some projects offer multiple SCMs for the same code
  - Gateway tools offer interoperability
  - As a new developer, choose the SCM that most of the existing developers in the project use
- Conversion tools
  - Many newer SCMs offer automated conversion
  - Allows project history to be preserved
  - Often requires some manual tweaking
  - Best known general converter is 'tailor'