Source code management

COMP8440: FOSSD
Lecture 3

Early approaches

- Simple tools
  - diff – show lines that differ between two files
  - patch – modify file according to diff file
  - tar – create “tape archive”
    - Usually then compressed with gzip or bzip2
  - Patches sent by email
  - Each developer maintains their own tree
  - Distribution of tar files 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
  - don't send html encoded email!
  - 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
  - Designed to work on a single machine
  - Only one person editing any file 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 used by some projects, but fewer every year
- Many limitations
  - Poor rename and directory support
  - Contacts centralised server for most operations
  - Poor branch merging support

Centralised vs Distributed

- Where is the project hosted?
  - CVS requires a central server
  - 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
    - Handles renames much better
    - Branching support is better but still not perfect
  - Adds project-wide revisions
  - Widely chosen to replace CVS in FOSS projects from 2001 onwards until 2005 or so
  - Still quite 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
    - Very sophisticated but also very hard to use
- Bitkeeper
  - Concepts from SCCS, extended to support distributed branching and merging
  - 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, mercurial, 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
  - Distributed by design, but server not free
    - FOSS projects all used a server hosted by BitMover
- Move to git
  - Makers of BitKeeper disapproved of free client
  - Free access to BitKeeper withdrawn in 2005
  - Replaced by new system 'git' in June 2005

SCM Interfaces

- Command line dominates
  - Most FOSS developers use command line interfaces
    - Complicated history can be hard to follow using CLI tools
  - 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, svnweb and gitweb are 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

Git concepts

- Git stores “objects”, each identified by hash
  - Hash is 160-bit SHA1 value, 40 hex digits
    - Can usually be abbreviated to 6–12 digits
  - File object (or “blob”): a single file
  - Tree object: a set of files and directories
  - Commit object: a reference to a tree, plus metadata
    - commit message
    - author and committer, with dates
    - one or more parent commits
  - Tag object: reference to another object, with optional cryptographic signature
- A branch is just a label for a commit
  - git maintains a record of the current branch
- Working directory
  - May differ from most recent commit on current branch

Git basic commands

- “git” command has many subcommands
  - git init – initialize a new, empty repository
  - git add – add files to a commit
  - git rm – remove files
  - git commit – commit changes
  - git branch – create a new branch
  - git checkout – check out a branch
  - git log – display commits on branch
  - git show – display an object (commit, blob, etc.)
  - git pull – bring in changes from another repository
  - git push – push changes to another repository
  - git merge – merge two (or more) branches together
  - git help <command> – get man page for subcommand
  - git diff – show differences in working tree
  - git config – examine and change git's configuration
  - git rebase – redo local commits on top of upstream

More git commands

- GUI tools:
  - gitk – useful for examining history and commits
  - git gui – useful for creating commits
- To work on an existing project, clone its repo:
  - git clone <URL>
  - URL is typically either https://… or git://…
- Create patches from your commits:
  - git format-patch
  - Creates one patch file for each commit
- Or send commits as emails directly:
  - git send-email --to=<addr> …
  - Requires configuration (SMTP server, etc.)
- “Bisection” for tracking down buggy commits
  - git bisect – pick commits for testing

Continuous Integration

- Integrates SCM with building and testing of code
- Requires good test suite for project
- Typically build/test is triggered by new commits
- Jenkins: https://jenkins.io/

Public SCM Hosting

- Canned hosting
  - Many/most FOSS projects use a canned hosting solution
  - Canned project hosting can make project maintenance 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: github.com, repo.oz.cz, bitbucket.org
    - Bazaar: 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'
- Builtin converters
  - some SCM tools have builtin access to other tools
  - “git svn” is paricularly useful