Software Archery:
hitting the bull's-eye with GNU Arch

Cameron Patrick

Perth Linux Users' Group
March 2005

Free software revision control

- Keeping your code organised
- Allowing easy access to past versions
- Coordinating changes between developers
- Allows working in different ‘branches’
  - **and merging between them**

- So many revision control systems to choose from these days.

CVS

- Oldest and most popular revision control system for free software.
- Technical problems: doesn’t track renames, symbolic links, file permissions, has issues with binary files.
- Social problems: only core developers can use CVS, everyone else is left on their own. Branching and merging is tricky.

Simple revision control

- `cp -r myproject myproject.backup`

- Every time you make a major change to your software, take a backup copy and name it consistently: e.g. v1, v2, v3, v4, ...
- This is the mental model used by Subversion: every time you commit, the repository-wide version number increases.
- Fine for a centralised system, but what if you want to send your changes to someone else?

Simple revision control II

- `diff -urN` *`myproject.orig myproject`*

- Keep a bunch of diffs lying around for each interesting change you make, and you can build the latest version from an older version and some patches.
- This is the mental model used by Arch.

Changesets

- Diff and patch have some limitations: e.g. they don’t handle file renames or binary files.
- Arch uses a ‘changeset’ instead: a bunch of patch files and some meta-data to handle things that patch can’t, all wrapped up in a tar.gz file.

Renames and logical file IDs

- When a file is added to an Arch project, it's given a logical file identity.
- Inside a changeset we record a list of file names and their corresponding IDs.
- This means that if you have a file 'foo.c' but I've renamed it to 'silly_foo.c', we can still exchange Arch changesets.

Turning it into a revision control system

- Tar up the initial state of your project and keep it in a directory called `base-0`.
- Put changesets in numbered directories: `patch-1`, `patch-2`, `patch-3`, etc.
- We're now most of the way to having an Arch archive.

Aside: signed archives

- Every changeset is stored in a separate file and never changes once it's been written.
- So it's trivial to store a GPG signature next to each changeset. (A *signed* Arch archive.)
- This means that if your server is compromised, you will be able to easily see if your code has been tampered with.
  - c.f. break-in at freedesktop.org not long ago

Distributed revision control: the Arch namespace

- Give each changeset a globally unique name
  - e.g.: cp@chem.com.au--2005/hibernate--debian--0--patch-5
- Keep track of which changesets have already been applied in each tree
- Now we can automatically merge changes made by someone else, in a completely different Arch archive
  - apply the changesets that we don't already have

Arch development: history and future

- Arch was original implemented as a shell script known as `larch`. Don't use it!
- Rewritten in C as `tla` (Tom Lord's Arch)
  - tla 1.x is the production version
  - tla 2.0 is an ambitious rewrite, just begun
- Forked by Canonical (the folks behind Ubuntu) as bazaar or `baz`
  - Making more aggressive user interface changes and testing some of the new ideas for tla 2.0

Some reasons why clever merging is cool

- Hacking on someone else's project
  - making your own changes in your own archive
  - catching up with their changes as necessary
  - and they can easily merge some or all of your changes
- Working off-line on a laptop
  - committing to an archive stored on your laptop
  - push changes back to your main archive when you're back at home

Sharing your code with the world

- No special “arch server” is needed.
  - Arch doesn’t even need to be installed on the server machine.
  - Your choice of SSH/SFTP, HTTP / WebDAV, or straight ftp.
- Usual combination is SFTP for committing and HTTP for read access
- ‘tla archive-mirror’ command to copy new revisions from your archive to a remote server

Getting started with arch

- Download Bazaar from http://bazaar.canonical.com/
  - Debian/Ubuntu packages are available
- Then head over to wiki.gnuarch.org
  - read the “quick introduction” and/or “learning Arch for CVS users”
- Maybe join the gnu-arch-users mailing list

Demonstration

"Arch sucks, what else is there?"

- Lots of free distributed revision control systems to choose from.
  - Arch is the most popular, but that doesn't necessarily mean it's the best for you.

- Darcs
- Monotone
- Quilt (not really a revision control system)
- Subversion (not really distributed)
- Bazaar-NG (still in very early development)

Darcs (David's Advanced Revision Control System)

- Amazingly simple to learn and become productive with
  - Though some operations which are simple in arch and cvs are harder in Darcs
- Scalability issues with
  - large projects
  - lots of branches
- One to watch: rapidly gaining momentum and catching up to Arch

Questions?