Fossil: New Ideas In Version Control

D. Richard Hipp
2009-09-30
What Is Fossil?

- Distributed version control
- Distributed bugs tracking
- Distributed wiki
- Built-in web interface
- “Autosync” mode
- Self-contained
- HTTP for all network traffic
- CGI-enabled
- Embedded Documentation
- Robust & Reliable
What Is Fossil?

- Distributed version control
- Distributed bugs tracking
- Distributed wiki
- Built-in web interface
- “Autosync” mode

- Self-contained
- HTTP for all network traffic
- CGI-enabled
- Embedded Documentation
- Robust & Reliable

Common  Unusual  Unique
Fundamental Concepts

- A “repository” is a bag of “artifacts”
- Artifacts identified by SHA1 hash
- Artifacts are unordered
Fundamental Concepts

- Sync by sharing artifacts
- Sync mechanism has no knowledge of versions, wiki, or tickets
- HTTP used for sync transport
Fundamental Concepts

- After sync, repositories have the same set of artifacts
- Delta and zlib compression minimizes bandwidth
- Shunned and private files excluded from sync
Classes of Artifacts

- Manifest
  - List of files
  - Parent check-in
  - Check-in comment
- Wiki page edit
- Ticket change
- Cluster
- Control
- General content
Repository Implementation

- Artifacts stores as BLOBs in an SQLite database
  - Delta compression
  - Z-lib compression
- Cross-reference and summary data stored in auxiliary tables of the same database
- “fossil rebuild” scans artifacts to rebuild auxiliary tables
fossil new *filename*

• Create a new repository

fossil clone *url* *filename*

• Make a copy of an existing repository
• Ex URL: http://userid:*password*@hostname:port/path
• Ex URL: file:///path
fossil open \textit{filename}

- Open a local source tree

fossil info
fossil changes
fossil status
fossil extra
fossil ls

- Information about the local source tree
fossil push [url]
fossil pull [url]
fossil sync [url]

- Synchronize repositories

fossil update [version]
fossil merge version

- Synchronize local source tree

fossil undo

- Back out prior update or merge
fossil commit

- Create new version from local tree
- --private flag
- --branch flag
fossil server [filename]

- Starts an HTTP server on the repository given
- --port flag
- Works on both unix and windows

fossil ui [filename]

- Automatically finds an open TCP port
- Automatically launches web browser
Web interface supports:

- Timelines of changes
- File browsing
- diffs and “annotate”
- wiki & ticket viewing and editing
- Editing check-in comments and display colors
- User management
- “Shunning” inappropriate content
- Appearance (CSS, headers, footers, etc)
Autosync mode

- Pull before update
- Pull before commit
- Push after commit
- Helps prevent needless forking and branching
- Enabled by default
Self-contained

- Single binary: fossil or fossil.exe
  - Client & server code
  - Diff & merge logic
  - built-in web server
- Download one file and put on your PATH
- No other required software (zero, nada, nil)
- Chroot ready
- Optional: GPG, custom diff programs
HTTP Data Transport

- Remote repositories specified by URL
- Works from behind restrictive firewalls
- Full support for proxies
- Deploy on economical shared host account
- Bandwidth efficient
  - Suitable for use over a dial-up connection
  - Typical check-in generates ~5KB of traffic
The actual 2-line CGI script that runs the canonical self-hosting fossil repository:

```
#!/usr/bin/fossil
repository: /fossil/fossil.fossil
```
Simple Wiki Formatting Rules

- Blank line for paragraph break
- “*” for bullets
- “1.” for enumerations
- Indented line for indented paragraph
- Hyperlinks in [...] 
- Safe subset of HTML for advanced markup
- `<verbatim>...</verbatim> 
- `<nowiki>..</nowiki>`
Embedded Documentation

http://baseurl/doc/version/filepath

- The fossil website is implemented this way
- `version` can be any version prefix, branch name, “tip”, or “ckout”
  - “ckout” allows viewing website before check-in
- MIME-Type from `filepath` suffix
- The “.wiki” suffix renders using wiki rules
Robust & Reliable

- Extensive use of MD5 and SHA1 checksums
  - Each artifact identified by SHA1
  - Control artifacts contain an MD5 checksum
  - Entire content of a check-in verified by MD5
  - Sync messages checked by MD5
- Recoverability checked prior to SQLite transaction commit
- No content has ever been lost from a fossil repository
Additional Noteworthy Features

`fossil help [commandname]`

- Built-in help

- Run commands against all repositories

```
fossil all push
fossil all pull
fossil all sync
fossil all rebuild
```
Before travel: fossil all pull

After return: fossil all push

autosync
Self-hosting since 2007-07-21

- http://www.fossil-scm.org/
- 1356 check-ins
- 310 files in the source tree
- 5366 artifacts
- 161 MB of content in a 9.2 MB repository
  - 17:1 compression ratio
- 4.8 MB network traffic to clone

As of 2009-09-26 21:00 UTC
Complete SQLite Source History

- http://www.sqlite.org/src
- 6863 check-ins over 9.25 years
- 923 files in the source tree
- 29252 artifacts
- 1.3 GB of content in a 22 MB repository
  - CVS required ~320 MB
  - 56:1 compression ratio
- 13.8 MB network traffic to clone

As of 2009-09-26 21:00 UTC
Review

- Distributed version control
- Distributed bugs tracking
- Distributed wiki
- Built-in web interface
- “Autosync” mode

- Self-contained
- HTTP for all network traffic
- CGI-enabled
- Embedded Documentation
- Robust & Reliable

Common

Unusual

Unique
Summary

- Pushing the state of the art in distributed version control
- Stable and ready to use
- Questions?
- Live Demo?

http://www.fossil-scm.org/