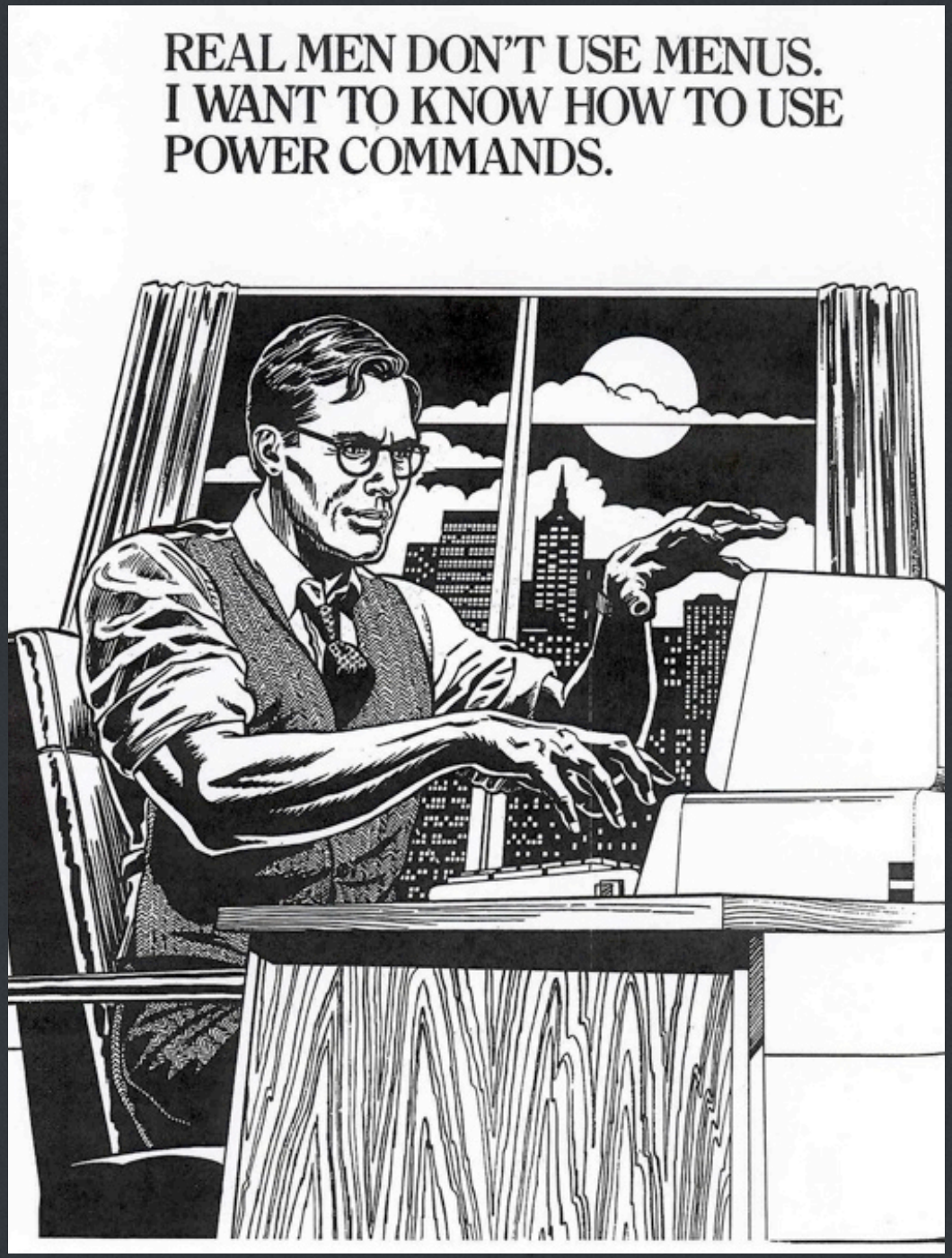
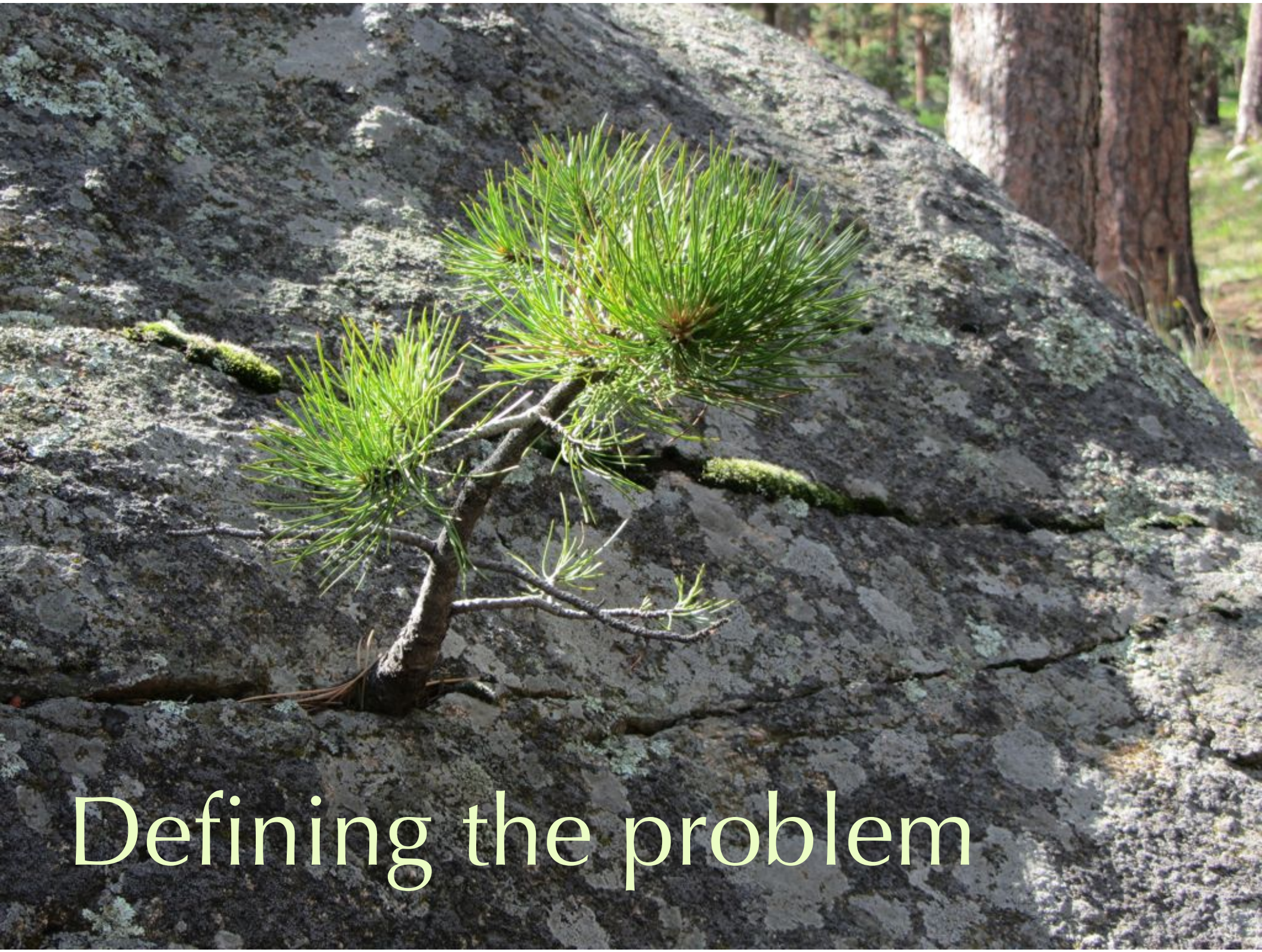
Managing Data with Version Control

Zack Williams
zdw@artisancomputer.com
http://artisancomputer.com

This is not a tutorial




Defining the problem

WTB: Business Tools

- Documenting work performed
- Accounting
- Organization
- Multiple users
- Backup, data integrity, etc.

What are my options?

	Hosted Solutions	File Sync	Version Control
Data Types	Limited	Any	Any
History	Maybe	None, or Limited	Full
Offline Access	None, or Limited	Yes	Yes
Sync Method	May exist	Automatic	User initiated
Conflict Resolution	Last change wins	Duplicates or Last change wins	Manual

What is a Version Control System (VCS)?

- A way to keep changes made to multiple copies of a set of files consistent
- Files are kept in Repositories (repos)
- Changes are logged, older versions retained
- Manual reconciliation of conflicts

Basic Concepts

- Get a copy of a repository
- Make changes to files, add and remove
- Describe and commit changes
- Put the changes back into the repo

Distributed VCS

- Creates a local copy of the remote repo
- Local changes can be pushed to the remote repo
- Changes in the remote repo can be retrieved
- Communication is SSH wrapped

Problems VCS solves

- Data is always available, no network required
- Changes eventually make it to everyone
- Data loss is unlikely
  - Everyone has entire repo history
  - Bad changes can be rolled back
  - Data is checksummed on disk

What should I use?

- One of the big two:
  - Git (`git`)
  - Mercurial (hg)

- For the most part, they're functionally identical

GUI or CLI?

- Both can do the basics
- CLI is more flexible, scriptable, more docs available
- GUI is better for browsing history
- Repos are just files on disk - use the tools you like

GUIs

- `git` - GitX, GitBox, Tower, SourceTree, GitHub for Mac, and more.
- hg - SourceTree, MacHG
- File status in Xcode, Vim, Textmate
- GUI diff tools
  - Filemerge, Kaleidoscope, etc.

Hosting

- Commercial Providers
  - GitHub, Bitbucket, etc.
- Self hosting
  - Nearly any server with SSH
  - Access Controls via `gitolite`, `mercurial-server`

Limitations

- Designed to work with code
  - Plain text works great
  - Other formats can be opaque to diff tools
- Large binary files are problematic

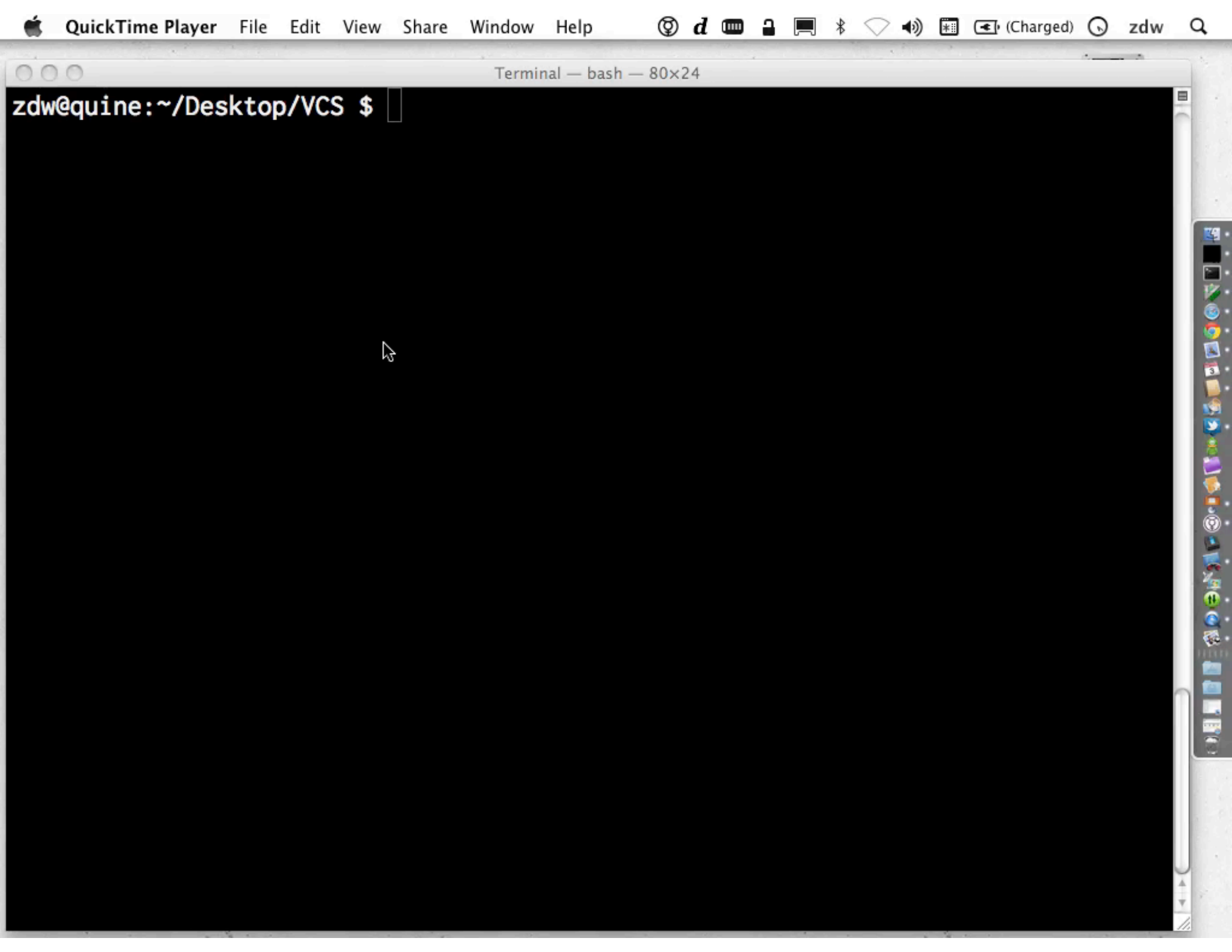
What should I keep in a repo?

- Things created by people
- Tools inside of tools!
- Unique or hard to reproduce data
- Not for ephemera, as a messaging system, etc.


Applying the tools to
the problem...

Documentation

- You will make notes
  - They will be important
  - They will change over time
- Use text for everything
- Including accounting data


QuickTime Player File Edit View Share Window Help
(Charged)
zdw
Terminal — bash — 80×24
zdw@quine:~/Desktop/VCS $

What else?

- Documentation driven design
- Automation
- Local software likes local files

Learn your environment

- Most people use a very small fraction of the tools available
- Build on the shoulders of giants

Questions?

Repo for this Preso:

https://github.com/zdw/mtc2011