Version Control

Ken Bloom

Linux User Group of Davis

March 1, 2005

You've probably heard of version control systems like CVS being used to develop software. Real briefly, a version control system is generally thought of as a centralized repository where developers put source code. They check out code from the project, work on it in a working directory on their own machine, and then commit the changes back to the repository when they are done. The oldest systems allowed only one user to edit a file at a time – if they checked it out to make changes, it was locked and nobody else could check it out to make changes. Now, concurrent editing is a standard feature.

1 Version Control Systems

- CVS
- BitKeeper
- Arch
- Subversion
- SVK

The first 3 should really be called "Source Code Management Systems" because their file system structures are pretty rigid, suited mostly for source code. The last two have a more flexible filesystem structure so they may more generally be called "Version Control Systems".

2 CVS

2.1 History

- started in 1986 as a bunch of shell scripts for RCS
- algorithms remain as a basis for the C program

2.2 Features

- Concurrent editing
- Directory tree storage
- Source code stored in a centralized repository
- Branches and tags
- Plain text repository format

2.3 Disadvantages

- Bad support for binaries or symlinks.
- Can't move, copy, or delete files in the repository
- No atomic commits.

3 BitKeeper

3.1 Features

- Distributed source code management.
- Concept of changesets that is bigger than commits.
- Some GUI features built in.

3.2 Disadvantages

- Not free as in speech.
- Free as in beer to anyone willing to use Open Logging.

Linus uses this for kernel development.

4 Arch

4.1 Features

- Changesets
- Distributed source code management.
- No particular preference for a given filesystem or protocol.

4.2 Disadvantages

- Very rigid repository structure.
- I think it's really difficult to learn
- Many more commands needed to do things that Subversion does automatically.

5 Subversion

- Learn from the net's experience of 10 years with CVS.
- Works a lot like CVS, with most obvious deficiencies gone.
- Most flexible.
- Also called by its command-name `svn`.

5.1 Features

- Centralized repository.
- Move, copy, delete files.
- Constant space and time copies
- Binary files
- Symlinks in client 1.1 or later
- No explicit branching and tagging features.

Think of it as a versioned file system with version on one axis and the directory tree on the other.

Subversion handles symlinks! It features a cool symlink emulation feature on Windows.

We can see from the suggested repository layout how Subversion manages branching and tagging. Constant space copies make this possible.

6 SVN Repository Layout

- project-name
  - trunk
  - branches
    - branch-1
    - branch-2
  - tags
    - tag-1
    - tag-2

7 SVK

- Originally called SubversionKeeper.
- Decentralized features of arch and BitKeeper.
- Simpler repository structure than arch.
- Use with Subversion, CVS, Perforce servers without special server-side support
- Horrible documentation.

8 Using Subversion

8.1 Subversion commands

Creating a repository:	`svnadmin create <path>`
Importing files:	`svn import <path> <URL>`
Checking out a directory:	`svn checkout <URL>`
Adding a file	`svn add <path>`
Committing changes	`svn commit`
Updating working directory	`svn update`
Merge from one branch into another	`svn merge`
Make branches and tags, copy files	`svn cp`
Use for backups:	`svnadmin dump`

`svnadmin create`: two repository formats to choose from: bdb (uses Berkeley DB), and fsfs. bdb can't be used over NFS, so you'll want to choose fsfs if you're setting up your subversion repository that way.

These are the commands I used for my SVN/SVK demonstration:

```
svnadmin create --fs-type=fsfs THEREPO
export SVN=file://$(pwd)/THEREPO
cp -a /etc/X11 .
svn mkdir $SVN/trunk
svn import X11 $SVN/trunk
svn stat X11
rm -rf X11
svn ls $SVN
svn ls $SVN/trunk
svn co $SVN/trunk X11
cd X11
# make some edits to files
svn diff
svn stat
svn revert #one file
svn commit #the rest
svn rm xkb
ls #doesn't disappear
svn commit
ls #disappears
svn mv Xresources MovingThisSomewhere
ls #occurs immediately
svn commit
```

8.2 svk commands

Most of the Subversion commands are svk commands too!

Mirror of a remote repository	`svk mirror <URL> <depotpath>`
Synchronize that mirror	`svk sync <depotpath>`

Merge specific commits	`svk cmerge -c <revisions> \`
	`   <srcpath> <destpath>`

Star-merge knows what's been merged already	`svk smerge`

Generate a patch file by adding the `-p` option to `smerge` or `cmerge`.

More steps in the demonstration

```
svk depotmap -i #setup svk for the first time
svk ls //
svk mirror $SVN //theX11repo
svk ls //
svk ls //theX11repo
svk sync //theX11repo #import the whole version history
svk rm //theX11repo
svk mirror $SVN //theX11repo
svnlook youngest THEREPO
```

```
svk sync -s 5 //theX11repo #import version history starting at version 5
svk cp //theX11repo/trunk //localbranch
svk co //localbranch
svk rm Xsession.d #this is an easy edit.
svk rm fluxbox
svk smerge //localbranch //theX11repo/trunk
svn ls $SVN/trunk
#smerge propagated changes back to the original repository
```

9 Backing up

This script is called from part of a larger script that generates a backup.iso and then actually burns it.

```
#!/bin/bash
export R=/home/bloom_svn
LAST=$(svnlook youngest $R)
if [ -e /cdrom/subversion_latest ]; then
    FIRST=$( cat /cdrom/subversion_latest )
    FIRST=$(($FIRST + 1))
else
   FIRST=0
fi
svnadmin dump $R --deltas --incremental \
   -r${FIRST}:${LAST} > $1
echo $LAST > $2
```

10 At $HOME in Subversion

I keep my whole home directory (except for mail) in Subversion.

10.1 Reasons for my setup

- Keep computers in sync
- Easy incremental backups of my documents
- Version history
- Joey Hess says: distributed backups

10.2 Features of my setup

- Three (nonoverlapping) sets of dotfiles: `.hide`, `.home-plus`, `home-base`.
- Partial Checkouts

10.3 Tools (and kludges) for my setup

- `~/bin/recursive`
- `.svnfix`
- jpilot-backup

Joey Hess used to use CVS modules to create different checkouts of his system. CVS would automatically recursively check those files in and out. The most analagous feature on Subversion is the `svn:externals` property. But that doesn't have recursive commits.

I don't use `svn:externals` anyway, but I like recursive commits – I check out anything I need into the top level of my home directory.

I let `~/bin/recursive` traverse into these directories and commit and update them.

10.3.1 /bin/recursive

```
#!/bin/sh
echo =======
echo '* '~
cd $HOME
svn $1
for x in * .home-plus .hide; do
   if [ -e $x ] ; then
      echo =======
      echo '* '$x
      cd $x && svn $1
      cd $HOME
   fi
done
```

10.3.2 .svnfix

Lives in `.home-plus` and `.hide`

- Links things into their appropriate places
- fixes some permissions.
  - .ssh/authorized_keys
  - ssh secret keys
  - .fetchmailrc
- Copied from Joey Hess' svn repository at `http://svn.kitenet.net/trunk/`

10.3.3 jpilot-backup

- I use jpilot as my PalmPilot desktop software.
- jpilot's default backup behavior is to rename directories a lot (too much to keep things backed up in Subversion)
- Solution: the jpilot-backup plugin.
- check the PersistentArchive checkbox
- sync once to get all of the databases in place.
- keep .jpilot/Backup/MainArchive in Subversion.

11 Where do things live in my home directory

11.1 The root

- `bin/` – I absolutely need this otherwise some things just don't work
- `hide/` – rename this to .hide after checking it out
- `hide-insecure/` – contains an SSH private key with a password
- `home-base/` – move everything here into the root of the home directory after checking out
- `home-plus/`
- `parts/`
- `research/` – this has been here since before I moved everything to Subversion.

I decided to leave research where it was, rather than moving it into `parts/` because I still had several working directories checked out, pointing to it.

It's probably quite insecure to have .hide kept in the same repository, even though I only check it out over ssh or locally.

The best way for you to determine which dotfiles should go where is to review your own usage habits on your computers, and determine which files you would like to have move around with your profile, which ones you would like to have only on computers with larger capacities, and which ones are truly secret.

Each of the .directories is liable to have its own collection of included/excluded files, for example browser cache files aren't included, and lots of files that change really frequently aren't included.

12 Resources

- `http://subversion.tigris.org/` – Subversion
- `http://svnbook.red-bean.com/` – Subversion Book
- `http://svk.elixus.org/` – SVK
- `http://www.kitenet.net/~joey/cvshome.html`
  – The original article "At $HOME in CVS"
- `http://www.kitenet.net/~joey/svnhome.html`
  – Joey's new article posted after I agreed to do this talk.
- `http://better-scm.berlios.de/comparison/comparison.html`
  – A comparison of source code management systems