

Version Control with Subversion and Xcode

Author: Mark Szymczyk

Last Update: June 21, 2006

This article shows you how to place your source code files under version control using Subversion and Xcode. By using version control you will be able to keep track of all the changes you make to your files and go back to earlier versions if you need to. The article focuses on local repositories stored on your Mac, but nearly all of the information is relevant if you want other people to be able to access your repositories.

Introduction

Subversion is a version control system, keeping track of the changes made to a file and who made the changes. While the use of version control is not limited to source code files (I could use version control to keep track of the changes I make to the articles and book chapters I write), the primary users of version control are software developers.

Version control is especially helpful on large projects with multiple developers. Each developer can add code to a file, and the version control system records the code each developer added and who added the code. Even if you're working on a project by yourself, version control can help you. If you've ever mistakenly saved a file and wished you could go back to the way the file was before you saved it, you'll appreciate version control. With version control you can go back to an older version of a file.

Many of you are familiar with CVS, which is a popular version control system. Subversion has several advantages over CVS.

- Subversion can handle the renaming of files.
- Subversion can track the changes made to directories.
- Subversion handles atomic commits. Atomic commits allow you to commit changes in multiple files while making sure all the changes get committed.

Xcode supports Subversion, and you can perform most version control tasks in Xcode. But there are four tasks you must perform before you can use Subversion in Xcode.

- Install Subversion.
- Create a repository, which is where Subversion stores the files you place under version control.
- Add your Xcode project to the repository.
- Check your project's files out of the repository.

On Mac OS X you must launch the Terminal application to enter the commands to create a repository, add a project to a repository, and check files out of the repository.

Installing Subversion

Subversion does not come with the Xcode Tools so you must download Subversion and install it. The Subversion website contains installers for many operating systems, including Mac OS X. The Coding Monkeys website also has a Mac OS X Subversion installer. I installed Subversion using the Coding Monkeys installer, and the installation was easy and error-free. I assume the installer on the Subversion site works just as well.

When you install Subversion, the installer installs it in the `/usr/local/bin` directory. The problem is that the Terminal application looks for programs in the `/usr/bin` directory by default. If you type in a Subversion command, you will get an error message saying the command was not found. You have to set the `PATH` environment variable to point to `/usr/local/bin`.

How you set the `PATH` variable depends on what Unix shell you're running in Terminal. On Mac OS X there are two common shells: `tcsh` and `bash`. The Terminal window's title bar tells you what shell you're using. If you're using `tcsh`, type the following command to set the `PATH` environment variable:

```
setenv PATH /usr/local/bin
```

If you're using `bash`, type the following command to set the `PATH` environment variable:

```
export PATH=$PATH:/usr/local/bin
```

Creating a Repository

There are two types of repositories you can create in Subversion: local and remote. A *local repository* resides on your computer and will be used only by you. A *remote repository* allows other people to access the repository and check out the files in the repository from their computers. This article covers local repositories. Apple has an article on their developer site that shows you how to setup remote Subversion repositories.

Use the `svnadmin create` command to create a Subversion repository. Enter the path to the directory where you want to place the repository. The following command creates a repository in the folder `MyRepository` on the hard drive used as your startup disk:

```
svnadmin create /MyRepository
```

You can create one repository per project or place multiple projects in a repository. The way Subversion updates version numbers may affect your decision on how many projects to place in one repository, especially if you're used to the way CVS updates version numbers. In CVS when you add a file to the repository, that file has version number 1. Making a change to the file increases the version number to 2. Future changes increase the version number to 3, 4, 5, and so on. Subversion works differently. It maintains one set of version numbers for the entire repository instead of giving each file its own version number.

Subversion updates the version number every time a file gets committed to the repository. When you add your first project to the repository, the files in the project will have version number 1. If you add a file to the repository, that file will have version number 2. If you make changes to the newly added file, the new version will be at version 3. If you add a second project to the repository, the second project's files will have version number 4. Every time the repository changes, Subversion updates the version number.

If you have a lot of projects in one repository, the version numbers can start to get high. You may not want to deal with version number 39742 of a file. Having one project per repository makes keeping track of version numbers easier. Having one repository for all your projects makes maintaining the repository easier. The number of projects you place in one repository is up to you.

Adding a Project to the Repository

After creating your repository, you must add a project to the repository so you can place the project's files under version control. Adding a Xcode project to a Subversion repository requires the following steps:

- 1) Create a project in Xcode if you haven't already done so.
- 2) Add folders with the names `branches`, `tags`, and `trunk` to the project folder.
- 3) Move the project files to the `trunk` folder.
- 4) Run the `svn import` command.

All Subversion requires is an initial Xcode project. After importing the project, you can add files to your Xcode project and add those files to the Subversion repository from Xcode. If you have an existing Xcode project you've been working on that you want to place under version control, that's great too. You can import the project and all its files into Subversion.

Adding the `branches`, `tags`, and `trunk` folders is not a mandatory step to add a project to a Subversion repository. But doing so will make your life easier if you need to branch your code in the future. Suppose you have a Mac OS X application, and you decide to create Linux and Windows versions. You could use the same Subversion repository, but have three branches: one for the Mac version of your code, one for the Linux version, and one for the Windows version. I recommend adding the `branches`, `tags`, and `trunk` folders just to be safe.

If you decide to add the `branches`, `tags`, and `trunk` folders to your project folder, you must move the files in the project folder to the `trunk` folder. You should move the project's build folder out of the project folder when adding the project to the repository. The build folder contains files Xcode creates when it builds your project, such as object and executable files. The contents of the build folder can become large as you work on your project, and the build folder's contents are files you don't want to place under version control because the files' contents are going to change every time you build your project.

Run the `svn import` command to add your project to the repository. The `svn import` command takes the following form:

```
svn import [ProjectFolderName] [Repository URL] [-m Message]
```

For a local repository, the repository URL starts with `file://` and contains the path to the repository. Suppose you have a project called `MyProject`. It resides in a folder called `MyProject` inside a `Code` directory on your startup disk. In this case you would have to navigate to the `Code` directory and run the `svn import` command by typing the following in the Terminal:

```
cd /Code
svn import MyProject file:///MyRepository/MyProject -m "Importing
project"
```

If you want to store more than one project in your repository, you should tack on a `MyProject` subdirectory to the repository URL like I did in the example. If you don't supply the `MyProject` subdirectory, Subversion imports the project in the repository's root. Only one project can reside in the root. If you import a second project without supplying a subdirectory, you will get an error saying the file pointed to by the repository URL already exists. Tacking on the `MyProject` subdirectory to the repository URL lets you have multiple projects in one repository.

Checking Files Out of the Repository

After importing your project, you must check the project's files out of the repository. To check files out of the repository,

- 1) Create a folder where you want the working copy to reside. You can do this in the Finder or from the Terminal with the `mkdir` command.
- 2) In the Terminal application, navigate to the folder you created in Step 1.
- 3) Run the `svn checkout` command.

The `svn checkout` command takes the following form:

```
svn checkout [Repository URL] [Folder Name]
```

To continue the example from the previous section, suppose you want the working copies of the files in the `MyProject` project in a folder called `WorkingCopy` in the `MyProject` folder. Type the following commands to check out the files in the `MyProject` project:

```
cd /Code/MyProject/WorkingCopy
svn checkout file:///MyRepository MyProject
```

Notice that I supplied the repository URL without any subdirectories when running the `svn checkout` command. Your `MyProject` folder should have the following structure after running `svn checkout`:

```
MyProject
  branches (No longer needed)
  tags (No longer needed)
  trunk (Contains original versions of project files)
  WorkingCopy
    MyProject
      branches
      tags
      trunk (Contains files under version control)
```

All the files that were imported will now be checked out in the `MyProject/WorkingCopy/MyProject/trunk` folder. The `WorkingCopy` folder contains the files under version control. You can remove the `branches`, `tags`, and `trunk` folders in the original `MyProject` folder. They are no longer needed because you created the `WorkingCopy` folder.

Turning on Version Control in Xcode

Before you can use Subversion with Xcode, you must turn on version control in Xcode. Xcode does not turn on version control when you create a project. To turn on version control for your project, open it and perform the following steps:

- 1) Select the project file from the Groups and Files list.
- 2) Click the Info button to open the project file's information panel.
- 3) Click the General tab.
- 4) Choose Subversion from the SCM System pop-up menu.

- 5) Select the Enable SCM checkbox.
- 6) Click the Edit button next to the SCM System pop-up menu.
- 7) Make sure the Subversion tool path is `/usr/local/bin/svn`.

Now that you have version control turned on for your project, you can go about the normal business of development: writing code, compiling it, testing it, and debugging it.

Seeing Which Files Aren't Up To Date

Initially the files in your project match the files in the repository. As you work on your project, editing source files and adding files to the project, some files no longer match. These files are the ones you must update in the repository. To see the list of files that don't match the saved version in the repository, select SCM from the Groups and Files list. The project window lists the files that don't match. The left column has a one-character code representing the file's version control status. Table 1 lists the possible codes.

Table 1 Version Control Status Codes

Code	Description
?	The file is not in the repository. When you add a file to your project, it won't be in the repository.
-	The file is in a folder that's not in the repository. You must add the folder to the repository from the command line.
M	You've modified the file. Choose <code>SCM > Commit Changes</code> to save your changes to the repository.
A	To be added. The file will be added to the repository the next time you choose <code>SCM > Commit Changes</code> .
R	To be removed. The file will be removed from the repository the next time you choose <code>SCM > Commit Changes</code> .
C	The changes you made to the file may conflict with the changes the latest version made. You would get this code if another programmer modified a file while you checked that file out.
U	The version of the file you're using is older than the latest version in the repository.

Unless you're running an older version of Xcode, you should see a small button in the lower right corner of the project window. Clicking that button shows the SCM detail view, which you can see in Figure 1. The detail view provides a transcript of all the files that are not up to date.

```

cd /OpenGL 2D Book Projects/LevelEditor/WorkingCopy/LevelEditor/trunk/
/usr/local/bin/svn status --show-updates
?      build
M      1 LevelEditor.xcodeproj/markszym.mode1
M      1 LevelEditor.xcodeproj/markszym.pbxuser
M      1 LevelEditor.xcodeproj/project.pbxproj
Status against revision: 1

cd /OpenGL 2D Book Projects/LevelEditor/WorkingCopy/LevelEditor/trunk/
LevelEditor.xcodeproj/
/usr/local/bin/svn info markszym.pbxuser project.pbxproj
Path: markszym.pbxuser
Name: markszym.pbxuser
URL: file:///Users/markszym/MarkSubversionRepository/trunk/LevelEditor.xcodeproj/
markszym.pbxuser
Repository Root: file:///Users/markszym/MarkSubversionRepository
Repository UUID: 6baa069e-e514-0410-8699-c6b2a477c59b
Revision: 1
Node Kind: file
Schedule: normal
Last Changed Author: markszym
Last Changed Rev: 1
Last Changed Date: 2006-05-29 00:53:32 -0400 (Mon, 29 May 2006)

```

Figure 1

Xcode's SCM detail view.

Committing Changes You Made

You've made changes to one of your source code files. It could be new code, a bug fix, a speed boost, or taking some ugly code that somehow works and cleaning it up. You tested the changes you made, and everything works perfectly. At this point you want to send the changes you made to the file to the repository, creating a new version of the file in the repository.

When you send your changes to the repository, Xcode prompts you for a message where you describe the changes you made. Before sending the changes, you may want to look at the changes you made to the file so you can type a meaningful message when prompted by Xcode. To see the changes you made to a file, choose **SCM > Compare with > Latest**. One annoying bug in FileMerge, the program that compares the files, is it occasionally treats the entire file as one difference. When FileMerge treats the entire file as one difference, you can't tell the differences between your version of the file and the version stored in the repository.

When you're ready to send your changes to the repository, choose **SCM > Commit Changes**. Xcode prompts you to type in a message describing the changes you've made. After typing the message, click the **Commit** button to send the changes. Now there's a new version of the file in the Subversion repository.

Discarding Changes

Suppose you made some changes in a file and found out the changes didn't work. How do you go back? Choose **SCM > Discard Changes**. Xcode erases all the changes you made and takes you back to the last version you sent to the repository. Discarding changes is the solution when you save a file and wish you hadn't.

Adding Files to the Repository

If you create an Xcode project and check the project's files out, you can add files to the project and add them to the Subversion repository from Xcode. To add a file to the repository, perform the following steps:

- 1) Select the file you want to add from the project window.
- 2) Choose **SCM > Add to Repository**. The file you want to add has the status **A**, which means it's ready to be added to the repository.
- 3) Choose **SCM > Commit Changes** to add the file to the repository.

Removing Files from the Repository

Although it's more common to add files to a project than to remove them, you can remove files from a Subversion repository from Xcode. To remove a file from the repository, perform the following steps:

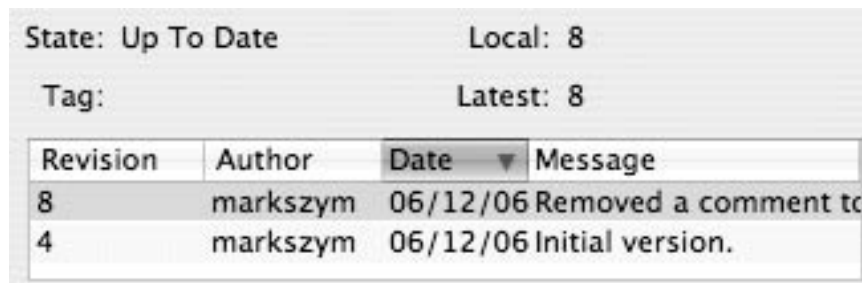
- 1) Select the file you want to remove from the Groups and Files list.
- 2) Press the Delete key.
- 3) An alert opens, asking if you want to remove the file from disk or just the reference to the file from the project. If you want to remove the file from the repository, you should remove the file from disk. If you remove just the reference, you will have to run the `svn delete` command from the Terminal to remove the file from the repository.
- 4) If you told Xcode to remove the file from disk, a second alert opens, asking if you want to remove the file from the SCM repository. Click the Remove button.
- 5) Select SCM from the Groups and Files list. The file you want to remove should have an R next to it saying it's ready to be removed.
- 6) Select SCM > Commit Changes to remove the file from the repository.

Seeing a File's SCM Information

After saving several versions of a file, the file's SCM information panel becomes more useful. Selecting a version from the panel fills the bottom area of the panel with information about that version of the file, including.

- The file name.
- The version number.
- Who checked the version into the repository.
- When the person checked the version into the repository.
- A message describing the changes made in this version.

In Subversion the version number represents the version of the repository, not the version of the file. If you look at Figure 2, you will see the file has version numbers 4 and 8. Where are versions 1-3 and 5-7? Those versions are for other files in my Subversion repository. Subversion updates the version number every time you make a change to the repository, such as importing a project, adding a file, or creating a new version of a file. When I added the source code file in Figure 2 to my repository, it was the fourth change I made to the repository. When I modified the file and committed the changes, it was the eighth change I made to the repository.



The screenshot shows the SCM information panel in Xcode. At the top, it displays 'State: Up To Date' and 'Local: 8'. Below that, it shows 'Tag:' and 'Latest: 8'. A table below lists the revision history:

Revision	Author	Date	Message
8	markszym	06/12/06	Removed a comment to
4	markszym	06/12/06	Initial version.

Figure 2

SCM version info.

Comparing Two Versions of a File

If you select two versions of the file, the Compare and Diff buttons become enabled. Clicking the Compare button shows the differences between the two versions in the FileMerge application. Clicking the Diff buttons opens a window in Xcode. In the window are the two versions of the file you're comparing side by side. Seeing the differences using the Diff button is more difficult. A changed line of code is indented a few spaces in the right hand version of the file, which isn't as noticeable as FileMerge's highlighting of the differences.

Reverting to an Old Version of a File

If you select an old version of the file, the Update button becomes enabled. Clicking the Update button sets the current version of the file to the selected version.

If you select the latest version of a file, and you've made changes to the file recently, the text of the Update button changes to Discard. Clicking the Discard button takes you back to the last version you saved to the repository, erasing the changes you made.

Viewing Annotations

When multiple people make multiple changes to a file, it's nice to know who changed what in the file. Annotations perform this vital task, telling you the following information for each line in a source code file:

- The version of the file that modified the line of code.
- Who modified the line.
- When that person modified the line.

To view the annotations, select the file from the project window and choose `SCM > Get Annotations for > Latest`. To see annotations for an older version of the file, choose `SCM > Get Annotations for > Revision`. A dialog appears with a list of all the versions for the file. Select the version you want and click the Annotate button.

Conclusion

Subversion is a large topic, large enough to write an entire book about it. Several people have already done so. There is no way a single article can cover everything about Subversion. To learn more about Subversion, go to the Subversion website. There is an online book you can read that should answer any questions not covered in this article. It also provides links to Subversion GUI programs so you can avoid using the command line.